University of Strathclyde

A Governed Real-Time Architecture for Cycle-Level Digital Representation, Quality Monitoring and Decision Support in Cyclic Manufacturing: Injection Moulding as a Case Study

Supervised by

Prof. Yi Qin

Submitted by

Manaf Al-Ahmad

A thesis submitted to the Department of Design, Manufacturing and Engineering Management for the degree of Doctor of Philosophy.

2026

Author's Declaration

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree. During its preparation, Grammarly was used to assist with proofreading, spelling, grammar, punctuation, and language clarity, while GitHub was used to support software development, source-code management, and version control. The author retained full responsibility for the research methodology, experimental work, analysis, interpretation of results, and final content of the thesis.

Signed

Manaf Al-Ahmad

Date

18/08/2026

Copyright Statement

The copyright of this thesis belongs to the author under the terms of the United Kingdom Copyright Acts as qualified by the University of Strathclyde Regulation 3.50. Due acknowledgement must always be made of the use of any material contained in, or derived from, this thesis.

Abstract

Cyclic manufacturing processes exhibit repeated operating patterns, yet small variations within and between cycles can affect product quality and undermine the reliability of offline prediction models. This thesis proposes, implements and evaluates a governed real-time architecture for cycle-level digital representation, quality monitoring and decision support, using injection moulding as the experimental case study.

The production cycle is treated as the traceable unit connecting acquired sensor signals, digital representations, predictive outputs, released decisions, subsequent quality confirmation and model-management actions. Two predictive representations were evaluated: an event-anchored feature vector aligned to the global maximum cavity pressure, and a full-curve representation retaining temporal process information. These were integrated into an architecture comprising a fixed Champion model, an updateable Challenger, drift-aware monitoring, a conservative final decision policy, human oversight, controlled retraining conditions and persistent evidence records.

Evaluation used 903 historically labelled cycles and two governed runtime runs of 850 cycles each, representing stable and retraining-enabled operation. The best offline configurations achieved accuracy of 0.966, F1-score of 0.963 and ROC-AUC of 0.994. Runtime evaluation examined classification performance, model disagreement, drift, processing latency, decision-loss sensitivity and a recorded retraining event. These findings emphasise traceability and governance alongside predictive accuracy.

The principal contribution is an implemented and evaluated governed real-time architecture that integrates cycle representation, quality prediction, runtime monitoring, traceable decision release and controlled model updating within a single operational framework, rather than treating these functions as isolated problems. This integration preserves explicit boundaries between prediction, decision release and model-update authority, without claiming autonomous control of machine actuators. Although the architecture provides a basis for cycle-level decision support in other cyclic manufacturing processes, transfer requires process-specific adaptation and independent validation.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Professor Yi Qin, for his guidance, patience, encouragement, and continuous support throughout my PhD journey. His knowledge, experience, and high standards have greatly shaped this research and my development as a researcher. I am especially grateful for the trust he placed in me, the opportunities he gave me, and the time he dedicated to discussing ideas, reviewing my work, and guiding me through the difficult stages of this study. His support helped me to think more clearly, work more independently, and continue with confidence when the research path was challenging. The lessons I have learned from him will remain with me beyond this thesis, not only in my future academic and professional work, but also in the way I approach challenges, responsibilities, and lifelong learning.

I would also like to thank the Department of Design, Manufacturing and Engineering Management at the University of Strathclyde for providing the research environment, facilities, and support needed to complete this work. I am grateful to the academic, technical, and administrative staff for their help and support during my studies.

I would like to extend my thanks to my colleagues and collaborators who shared their knowledge, experience, and advice throughout this research. Their discussions and support during experimental work, data analysis, system development, and thesis writing were greatly appreciated. I am also thankful to the friends I met during my time in Glasgow, whose kindness and companionship made this journey easier and more meaningful.

My deepest thanks go to my family for their unconditional love, patience, and support. Their belief in me has been a constant source of strength, especially during the most difficult moments of this journey. Although this PhD required time, effort, and sacrifice, their encouragement helped me to continue and complete this work.

Finally, I would like to thank everyone who supported me, inspired me, or contributed in any way to my academic and personal development during these years. This thesis would not have been possible without the guidance, support, and kindness of many people around me.

List of Publications

Conference publications

Al-Ahmad, M., Yang, S., Tian, Y., Christoforidis, G., Gao, J. and Qin, Y., 2023. Upgrading conventional production lines through implementing an industry 4.0 strategy. In *Advances in Manufacturing Technology XXXVI: Proceedings of the 20th International Conference on Manufacturing Research, Incorporating the 37th National Conference on Manufacturing Research, 6th–8th September 2023, Aberystwyth University, UK* (pp. 126-131). 1 Oliver's Yard, 55 City Road, London, EC1Y 1SP: SAGE Publications.

Al-Ahmad, M., Yang, S. and Qin, Y., 2024, August. Integrating machine learning with machine parameters to predict plastic part quality in injection moulding. In *MATEC Web of Conferences* (Vol. 401).

Yang, S., Al-Ahmad, M. and Qin, Y., 2024. A cloud infrastructure for enabling provision of the Material Testing as a Service (MTaaS). In *MATEC Web of Conferences* (Vol. 401, p. 10011). EDP Sciences.

Manuscripts under review or in preparation

Al-Ahmad, M., Qin, Y., Yang, S., Mekras, N., Smith, R., Charitidis, C.A. and Trompeta, K., An Event-Anchored Machine Learning based New Approach for Robust Real-Time Quality Prediction in Cyclic Manufacturing. *Available at SSRN 6309298*.

Al-Ahmad, M., Qin, Y., Yang, S., Mekras, N., Smith, R., Charitidis, C. and Trompeta, K. (in preparation) ‘A governed Champion–Challenger architecture for cycle-level quality decision support and model adaptation in cyclic manufacturing: an injection moulding case study’.

Table of Contents

Chapter 1 Introduction 1

1.1 Background and Motivation 2

1.2 Research aims and objectives 3

1.2.1 Aims 3

1.2.2 Objectives 4

1.3 Thesis Contributions 4

1.4 Overview of Methodology 7

1.5 Thesis structure 10

Chapter 2 Literature and Technology Review 13

2.1 Chapter overview and literature review approach 14

2.2 Cyclic manufacturing, real-time monitoring, and the research context 14

2.3 Injection moulding as the case-study process 16

2.4 Defects, cycle-level behaviour, and quality formation 18

2.5 Sensing, in-mould monitoring, and data acquisition for cyclic manufacturing 21

2.6 Industrial connectivity and monitoring architectures 24

2.7 Machine learning for quality prediction in manufacturing 28

2.8 Evaluation metrics, operational performance, and decision cost 32

2.9 Drift, concept shift, and model reliability in production 36

2.10 Model governance, controlled updating, and human-in-the-loop operation 39

2.11 Research gaps, research questions and chapter synthesis 42

Chapter 3 Experimental Setup and Data Collection 47

3.1 Manufacturing trials: cell configuration and case-study process 48

3.1.1 Experimental workflow and system overview 48

3.1.2 Production cell and machine configuration (case study) 50

3.1.3 Tooling and material context 51

3.1.4 Baseline operating window and planned variability 52

3.1.5 Link to the monitoring framework 53

3.2 Sensor system design and installation 54
3.3 Data Acquisition and Signal Capture Framework 56
3.4 Computational and Software Environment 57
3.5 Experimental Data Handling and Record Preparation 58
3.6 Handling outliers, extreme conditions, and data integrity 62
3.7 Governed real-time system methodology and architecture 67
3.7.1 System scope and operational definition 67
3.7.2 Methodology-level architecture 69
3.7.3 Operational reliability and performance considerations 70
3.7.4 Summary 71
Chapter 4 Cycle-Level Data Processing and Dual-Representation Feature Engineering 72
4.1 Chapter Overview 73
4.2 Feature Compilation Pipeline 74
4.2.1 Cycle segmentation and buffering 75
4.2.2 Event-anchored feature extraction at MP1 peak 75
4.2.3 Full-curve representation for Champion inference 76
4.2.4 Dataset assembly and governed storage 77
4.2.5 Interactive visual selection tool 77
4.3 Experimental Procedure and Trial Observations 77
4.3.1 First Trial 79
4.3.2 Second Trial 81
4.3.3 Third Trial 84
4.4 Features Selection and Description 86
4.5 Data Quality and Integrity 87
4.6 Summary 90
Chapter 5 Statistical Analysis of Process Variables and Their Relationship with Part Quality 92
5.1 Objective of the Statistical Analysis 93
5.2 Data Preparation and Feature Description 97

5.3 Correlation and Preliminary Data Exploration 99
5.4 Logistic Regression Modelling 101
5.4.1 Logistic regression between machine variables and quality 102
5.4.2 Model Fit and Performance Evaluation 107
5.5 Summary of Findings and Implications for Machine Learning 113
Chapter 6 Machine Learning for Cycle-Level Quality Classification in Cyclic Manufacturing 115
6.1 Chapter purpose and research question 116
6.2 Supervised learning for cyclic manufacturing quality 118
6.3 Dataset and case study description 121
6.4 Cycle representations and the two modelling routes 124
6.4.1 Why a cycle representation is needed 124
6.4.2 Event-anchored peak-based representation 127
6.4.3 Full curve representation and resampling 131
6.4.4 Engineering trade-off, information versus cost 133
6.5 Models considered and why they fit the study 135
6.5.1 Models for peak anchored features 136
6.5.2 Models for full curve inputs 139
6.6 Training and validation protocol 143
6.7 Results for peak anchored models 150
6.8 Results for full curve models 155
6.9 Direct comparison, peak anchored versus full curve 160
6.10 Discussion and implications for the governed real-time architecture 163
Chapter 7 Governed Real-Time Architecture for Cycle-Level Monitoring and Decision Support in Cyclic Manufacturing 167
7.1 Chapter purpose and scope 168
7.2 Operational Architecture of the Governed Real-Time System 172
7.3 Real-time data acquisition and cycle execution 178
7.4 Online Cycle Representation and Feature Construction 184
7.5 Champion, Challenger, and final decision policy 189

7.6 Run-Level Persistence, Traceability, and Evidence Structure 195
7.7 Live monitoring and operator-facing decision support 200
7.8 Drift awareness, human oversight, and governed retraining 205
7.9 Deployment Robustness, Safeguards, and Runtime Stability 210
7.10 Chapter Summary and Transition to Results Evaluation 215
Chapter 8 Live Evaluation and Discussion of the Governed Cycle-Level Architecture 219
8.1 Chapter purpose and evaluation scope 220
8.2 Stable Governed Live Examination Stage 223
8.3 Retrain-stage run performance under governed operation 235
8.4 Champion–Challenger comparison and disagreement behaviour 243
8.5 Final policy behaviour under true labels 247
8.6 Drift behaviour across the live run 253
8.7 Recorded retraining outcome and its governed meaning 256
8.8 Timing and real-time feasibility 260
8.9 Cost-sensitive decision trade-offs 266
8.10 Overall interpretation of the live evidence 274
8.11 Chapter summary 277
Chapter 9 Conclusions and Future Work 280
9.1 Concluding scope and response to the thesis aim 281
9.2 Key findings and response to the research questions and objectives 282
9.3 Contributions to knowledge and manufacturing practice 286
9.4 Generalisability, limitations and future research 289
9.5 Final concluding statement 293
References I
Appendixes I
Appendix A. Experimental Hardware, Sensor Installation, and Data-Acquisition Details I
A.1 Hydraulic pressure sensor installation I

A.2 Screw position sensor installation I
A.3 Nozzle temperature sensor installation II
A.4 Cavity pressure and temperature sensor installation II
A.5 Heating-water temperature and flow sensor installation III
A.6 ComoNeo acquisition platform and sensor connection overview III
A.7 Mould sensor connection and multi-channel acquisition IV
A.8 Conditioning hardware and connector layout IV
A.9 Sensor channel mapping V
Appendix B: Governed Real-Time System Interface and Runtime Evidence Views VI
B.1 Live cycle view and event-anchored alignment VII
B.2 Cycle-level monitoring and prediction table VIII
B.3 Run stability across the live evaluation window IX
B.4 Drift monitoring and distributional change IX
B.5 Governed retraining and evidence gates X
B.6 Final operational metrics and cost report XI
B.7 Final operational quality-evaluation view XIII
B.8 Champion–Challenger disagreement monitoring XIV
B.9 Rolling discrimination and model monitoring XV
B.10 Runtime latency and throughput XVI
Appendix C: Key Software Implementation Excerpts XVII
C.1 Event-anchor detection XVII
C.2 Synchronised event-state extraction XVIII
C.3 Five-fold cross-validation and out-of-fold evaluation XIX
C.4 Full-curve Champion representation XX
C.5 Challenger inference and stored decision threshold XXI
C.6 Final decision policy XXI
C.7 Drift calculation XXIII
C.8 Retraining admission conditions XXIV
C.9 Governed Challenger retraining XXV
C.10 Cycle-level persistence and traceability XXV

List of Figures

Figure 1. Conceptual governed real-time architecture for cycle-level digital representation, quality monitoring, and decision support using OPC UA acquisition and ML-based classification. 8
Figure 2. Conceptual governed workflow for cycle-level injection moulding quality monitoring and decision support. 9
Figure 3. End-to-end monitoring architecture linking in-mould and machine-side sensing, ComoNeo acquisition, OPC UA communication, machine-learning analysis, and dashboard-based decision support in the injection moulding case study. 10
Figure 4. Injection moulding case-study cell showing the machine, mould, sensor arrangement, data acquisition unit, and process-data flow. 50
Figure 5. Pressure and temperature operating boundaries are considered for the sensor system during the case-study trials. 55
Figure 6. Experimental data-preparation dashboard showing cycle curves, model outputs, and label-related fields. 60
Figure 7. Disturbed but valid cycle versus corrupted/invalid record 65
Figure 8. Data flow from sensor acquisition to cycle-level feature compilation. 75
Figure 9. Cycle-by-cycle variation of key process variables during the first trial, showing baseline operation, packing-pressure reduction phases, and recovery after restoration of nominal settings. 81
Figure 10. Cycle-by-cycle variation of key process variables during the second trial, showing staged disturbances in heating-water temperature, nozzle temperature, and screw position, followed by restoration to nominal conditions. 83
Figure 11. Cycle-by-cycle variation of key process variables during the third trial, highlighting machine interruption, later increases in injection speed and hydraulic pressure, and unstable sensor behaviour during flow-rate assessment. 85
Figure 12. Residual diagnostic plots for the MP1 regression model, showing non-normality, heteroscedasticity, skewness, and possible time-dependent drift. 95
Figure 13. ROC curve for the held-out test subset of 271 cycles, indicating the discrimination performance of the fitted classification model (AUC = 0.856). 108
Figure 14. Precision-recall curve for the held-out test subset, illustrating the trade-off between detecting defective cycles and avoiding false alarms (AP = 0.866). 109
Figure 15. Confusion matrix for the test set at threshold 0.50. 109
Figure 16. Confusion matrix on the test set at the selected operating threshold of 0.30. 110
Figure 17. Threshold sweep on the test set. 111
Figure 18. Reliability diagram for the test set. 111
Figure 19. Cycle-level supervised classification framework for manufacturing quality prediction. Structured representations of manufacturing cycles are evaluated by a supervised classifier that estimates the probability of a defect. A decision threshold then

converts this probability into a predicted quality class, illustrating the trade-off between defect detection and false alarms. 120
Figure 20. Example good and defective injection moulding cycles across the six monitored process variables. 124
Figure 21. Comparison of the peak-anchored and full-curve cycle representations evaluated using the same cycles, labels, train/test protocol, and metrics. 126
Figure 22. Peak event detection and synchronised sampling at the MP1 anchor. 130
Figure 23. Illustration of resampling from variable-length cycle data to a fixed-length representation of length L. 133
Figure 24. ROC curves for peak anchored models. 152
Figure 25. Precision-recall curves for peak anchored models. 152
Figure 26. Confusion matrix for Gradient Boosting. 153
Figure 27. Confusion matrix for XGBoost 154
Figure 28. ROC curves for full curve models. 156
Figure 29. Precision recall curves for full curve models. 157
Figure 30. Confusion matrix for full curve XGBoost 158
Figure 31. Confusion matrix for full curve Gradient Boosting. 159
Figure 32. Side-by-side ROC and precision-recall comparison of the best peak-anchored and full-curve models. 162
Figure 33. Conceptual bridge from validated machine-learning models to integration within the governed real-time architecture. 166
Figure 34. Conceptual transition from offline cycle-level model validation to the governed real-time architecture for manufacturing quality monitoring and decision support. 169
Figure 35. Operational structure of the governed real-time platform, showing the path from process-data acquisition to cycle-level governed decision and evidence storage. 173
Figure 36. Real-time data acquisition and cycle execution path, showing live streaming, temporary buffering, boundary confirmation, snapshot release, and transfer to the analytical layer. 180
Figure 37. Online cycle representation and event-anchored feature construction, showing the accepted cycle snapshot, MP1 event location, and aligned variables used to form the compact live feature vector 187
Figure 38. Predictive decision structure of the governed real-time platform, showing the fixed Champion path, the lighter Challenger path, and the locked Final policy layer. 191
Figure 39. Run-level persistence and evidence structure of the governed real-time platform, showing cycle records, event logs, correction records, retraining records, and configuration snapshots within one governed run. 197
Figure 40. Live monitoring view of the governed real-time platform, showing runtime visibility of process behaviour, prediction state, and timing performance. 202
Figure 41. Drift awareness and governed retraining pathway within the governed real-time platform, showing monitored change, human review, gated retraining, and controlled model update. 206

Figure 42. Main runtime safeguards of the governed real-time platform, showing cycle validation, fixed reference protection, gated adaptation, and preserved policy control. . 212
Figure 43. Overall logic of the governed real-time platform, showing the complete path from cycle-level process-data capture to governed runtime evaluation across the two operational stages. 216
Figure 44. Representative baseline-cycle alignment at the MP peak, showing the anchored event, the synchronised process state, and the Champion and Challenger scores at the same reference point. 224
Figure 45. Baseline confusion matrices for the Champion, the Challenger, and the Final output. 226
Figure 46. Baseline ROC curves for the Champion, the Challenger, and the Final output. 228
Figure 47. Baseline precision-recall curves for the Champion, the Challenger, and the Final output. 229
Figure 48. Baseline Champion score distribution relative to the operating threshold. 230
Figure 49. Baseline Champion calibration curve under live true-label evaluation 231
Figure 50. Baseline latency distribution 233
Figure 51. Retrain-stage confusion matrices for the Champion, the Challenger, and the Final output. 237
Figure 52. Retrain-stage ROC and precision-recall behaviour for the Champion, the Challenger, and the Final output. 239
Figure 53. Retrain-stage latency distribution. 241
Figure 54. Baseline Champion–Challenger comparison under true labels. 244
Figure 55. Retrain-stage Champion–Challenger comparison under true labels. 246
Figure 56. Baseline operating effect of the Final policy, showing the shift from the Challenger's raw output to the deployed Final output. 249
Figure 57. Retrain-stage operating effect of the Final policy, showing the shift from the Challenger's raw output to the deployed Final output. 250
Figure 58. Cross-run comparison of the final policy effect on false negatives, false positives, and false-alarm burden. 252
Figure 59. Drift timeline across rolling windows, showing mean PSI (blue) and mean KS (red) against CycleID, with the retraining event marked. 254
Figure 60. Retraining event positioned within the drift timeline, showing the temporal relationship between monitored distributional change and the recorded retraining cycle. 258
Figure 61. Baseline processing-time distribution and tail behaviour. 263
Figure 62. Retrain-stage processing-time distribution and tail behaviour. 265
Figure 63. Baseline normalised relative decision-loss sensitivity across assumed FN:FP penalty ratios. 269
Figure 64.Retrain-stage normalised relative decision-loss sensitivity across assumed FN: FP penalty ratios. 271
Figure 65. Cross-run comparison of processing-weighted false-positive burden (FP × mean processing time, ms) for the Champion, Challenger, and Final output 272

Figure A. 1. Kistler 4262A hydraulic pressure sensor used for machine-side pressure measurement. I
Figure A. 2. Screw position sensor installation on the injection moulding machine. II
Figure A. 3. Nozzle temperature sensing arrangement used to monitor melt temperature before mould entry. II
Figure A. 4. Kistler 6188A cavity pressure–temperature sensor used for in-mould process monitoring. III
Figure A. 5. Heating-water inlet temperature and flow sensor installation. III
Figure A. 6. ComoNeo system and sensor connection overview. IV
Figure A. 7. Pressure–temperature sensor installation on the mould and connection to the ComoNeo monitoring system. IV
Figure A. 8. Position of conditioning and connector hardware in the mould. V
Figure A. 9. Pressure–temperature sensor installation on the mould and connection to the ComoNeo monitoring system. Labels indicate: 1) sensors, 2) cavities, 3) Kistler display screen, 4) ComoNeo unit, and 5) multi-channel connection. VI

Figure B. 1. Cycle-to-digital-representation alignment view showing the detected MP peak, the aligned process state, and the Champion and Challenger scores for the final validated runtime evaluation. VII
Figure B. 2. Cycle-level monitoring table showing MP-peak-aligned features, Champion and Challenger predictions, truth labels, final governed labels, agreement state, and probability difference for the final validated run. VIII
Figure B. 3. Process stability view showing MP peak variation across the complete 850-cycle live evaluation window. IX
Figure B. 4. Drift monitoring view showing feature-level distributional change using PSI, KS, and JS-divergence indicators during the governed runtime evaluation. X
Figure B. 5. Governed retraining evidence view showing the drift timeline, retraining marker, evidence-gate status, and recorded retraining context at CycleID 745. XI
Figure B. 6. Run-level metrics and cost report for the retrain-stage run, showing Champion, Challenger, and Final performance together with cost-related evidence. XII
Figure B. 7. Final operational quality-evaluation view showing the confusion matrix and precision-recall behaviour for the governed Final output across all 850 truth-linked runtime cycles. XIII
Figure B. 8. Champion–Challenger monitoring view showing predicted defect probabilities, disagreement points, and probability-difference behaviour across the late-stage run window. XIV
Figure B. 9. Rolling ROC-AUC monitoring view showing Champion and Challenger discrimination behaviour over the live evaluation window. XV
Figure B. 10. Runtime latency and throughput view showing mean latency components for Champion inference, Challenger inference, total model latency, and end-to-end processing time. XVI

List of Tables

Table 1. Relationship between identified research gaps, thesis contributions, and supporting evidence. 45

Table 2. Tooling, material, machine, and monitored-process context for the injection moulding case study 51

Table 3. Baseline injection moulding settings used for reference operation. 53

Table 4. Sensor channels and measurement purpose in the experimental data collection framework 55

Table 5. Cycle representation summary comparing the Champion full-curve route and Challenger event-anchored route. 61

Table 6. Data issues encountered and handling decisions 64

Table 7. Live governance rules and recorded outcomes 65

Table 8. Distinction between a monitoring dashboard, a monitoring system, and the governed real-time architecture proposed in this thesis. 68

Table 9. Methodology-level system layers, inputs, outputs, and stored evidence. 70

Table 10. Class distribution and train–test partition of the 903-cycle dataset used for model development and held-out evaluation. 94

Table 11. Cycle-level feature definitions used for the statistical analysis, showing summary measures, signal source, and inclusion status. 98

Table 12. Logistic regression coefficients for defect probability using MP1 peak-anchored predictors. 105

Table 13. Binned calibration results for the test set. 112

Table 14. Test-set performance at threshold 0.50 and at the selected operating threshold. 112

Table 15. sensor channels and their role in the cycle. 122

Table 16. Structure of the event anchored feature vector. 129

Table 17. Peak-anchored route models and their role 138

Table 18. Models evaluated for the full-curve route and their roles. 142

Table 19. Training and validation settings by model family. 149

Table 20. Peak anchored model performance summary 150

Table 21. Full curve model performance summary 155

Table 22. Direct comparison of the best peak-anchored and full curve models 161

Table 23. Role of the two operational stages used to evaluate the governed real-time architecture. 171

Table 24. Main layers of the governed real-time platform and the operational function of each layer. 176

Table 25. Main stages of cycle execution within the governed real-time platform, from live signal acquisition to validated cycle snapshot release. 181

Table 26. Comparison of the two online cycle representations used in the governed real-time platform: full-curve reference representation and event-anchored operational representation. 188

Table 27. Functional roles of the Champion, Challenger, and Final layers within the governed decision structure of the real-time platform. 194
Table 28. Main persistent evidence components stored by the governed real-time platform and the interpretive purpose of each component. 199
Table 29. Main monitored outputs of the governed real-time platform and the operational value of each during runtime execution. 204
Table 30. Main elements of the governed retraining pathway in the governed real-time platform, including drift awareness, human review, gate conditions, and controlled Challenger updating. 209
Table 31. Main safeguards used to preserve runtime stability, decision control, and operational reliability in the governed real-time platform. 214
Table 32. Main implementation outcomes established in Chapter 7 and their role in supporting the measured evaluation presented in Chapter 8. 217
Table 33. Baseline run integrity summary, showing accepted cycles, truth-linked cycles, and class distribution. 223
Table 34. Baseline confusion-matrix counts and derived classification metrics (accuracy, precision, recall, and F1) for the Champion, the Challenger, and the Final output. 226
Table 35. Baseline threshold-independent discrimination results, showing ROC AUC and PR AUC for the Champion, the Challenger, and the Final output 229
Table 36. Baseline score and calibration summary, including threshold, ECE, Brier score, prevalence, and selected calibration-bin values. 231
Table 37. Baseline normalised relative decision-loss sensitivity for the Champion, Challenger, and Final output across the evaluated FN: FP penalty ratios. 232
Table 38. Baseline timing summary, including mean processing time, p95 processing time, and mean model latency for the Champion and the Challenger. 233
Table 39. Retrain-stage run integrity summary, showing accepted cycles, truth-linked cycles, class distribution, human corrections, and the recorded retraining event. 236
Table 40. Retrain-stage confusion-matrix counts and derived classification metrics (accuracy, precision, recall, and F1) for the Champion, the Challenger, and the Final output. 238
Table 41. Retrain-stage threshold-independent discrimination summary. 239
Table 42. Retrain-stage normalised relative decision-loss sensitivity comparison for the Champion, the Challenger, and the Final output. 240
Table 43. Retrain-stage timing summary, including mean processing time, p95 processing time, and mean model latency for the Champion and the Challenger. 242
Table 44. Baseline Champion–Challenger comparison, showing classification metrics, FP/FN counts, discrimination metrics, processing-weighted false-positive burden, and mean model latency. 245
Table 45. Retrain-stage Champion–Challenger comparison, showing classification metrics, FP/FN counts, processing-weighted false-positive burden, and mean model latency. 246
Table 46. Baseline comparison of the Challenger and Final output under true labels, showing the policy-induced change in FP, FN, and derived classification metrics. 248

TABLE 47. RETRAIN-STAGE COMPARISON OF THE CHALLENGER AND FINAL OUTPUT UNDER TRUE LABELS, SHOWING THE POLICY-INDUCED CHANGE IN FP, FN, AND DERIVED CLASSIFICATION METRICS. 250
TABLE 48. CROSS-RUN SUMMARY OF FINAL POLICY IMPACT RELATIVE TO THE CHALLENGER, SHOWING CHANGES IN FP, FN, PRECISION, RECALL, F1, AND PROCESSING-WEIGHTED FALSE-POSITIVE BURDEN. 252
TABLE 49. SUMMARY OF PSI AND KS DRIFT BEHAVIOUR ACROSS THE LIVE RUN, SHOWING EARLY-, MIDDLE-, AND LATE-STAGE DRIFT CHARACTERISTICS AND THE RELATIONSHIP BETWEEN THE MAIN PSI EXCURSION AND THE RETRAINING EVENT. 255
TABLE 50. RECORDED RETRAINING OUTCOME, SHOWING CORRECTION COUNT, RETRAINING COUNT, RETRAINING CYCLE, AND RUN INTEGRITY CONTEXT. 257
TABLE 51. GOVERNED INTERPRETATION OF THE RECORDED RETRAINING OUTCOME, DISTINGUISHING OPERATIONAL SUCCESS, ARCHITECTURAL CONTINUITY, DRIFT CONTEXT, AND LIVE FEASIBILITY. 260
TABLE 52. BASELINE TIMING SUMMARY, SHOWING MEAN PROCESSING TIME, P95 PROCESSING TIME, AND MEAN MODEL LATENCY FOR CHAMPION AND CHALLENGER. 261
TABLE 53. RETRAIN-STAGE TIMING SUMMARY, SHOWING MEAN PROCESSING TIME, P95 PROCESSING TIME, AND MEAN MODEL LATENCY FOR CHAMPION AND CHALLENGER. 264
TABLE 54. BASELINE NORMALISED RELATIVE DECISION-LOSS SENSITIVITY FOR CHAMPION, CHALLENGER, AND FINAL OUTPUT ACROSS ASSUMED FN:FP PENALTY RATIOS 268
TABLE 55. RETRAIN-STAGE NORMALISED RELATIVE DECISION-LOSS SENSITIVITY FOR CHAMPION, CHALLENGER, AND FINAL OUTPUT ACROSS ASSUMED FN: FP PENALTY RATIOS. 270
TABLE 57. CROSS-RUN CHANGE IN PROCESSING-WEIGHTED FALSE-POSITIVE BURDEN FOR CHAMPION, CHALLENGER, AND FINAL OUTPUT 272
TABLE 58. SUMMARY OF THE PRINCIPAL FINDINGS OF THE RESEARCH AND THEIR SUPPORTING EVIDENCE. 285

List of Equations

Equation 1. Logit transformation of defect probability 103
Equation 2. Defect probability from the linear predictor 103
Equation 3. Linear predictor for the MP1-anchored logistic regression model 103
Equation 4. General form of the logistic regression model 107
Equation 5. Predicted defect probability for cycle i 119
Equation 6. Binary classification decision rule 119
Equation 7. Full-curve input representation 132
Equation 8. Input sizes and dimensionality ratio of the cycle representations 134
Equation 9. Standard classification metrics 146
Equation 10. Rates used to construct the ROC curve 146
Equation 11. Live process state vector 174
Equation 12. Completed cycle window 174
Equation 13. Dual cycle-representation mappings 175
Equation 14. Champion and Challenger probability outputs 175
Equation 15. General final-decision policy 177
Equation 16. Active cycle buffer 179
Equation 17. Machine-confirmed cycle-completion gate 180
Equation 18. Formation of the completed cycle record 180
Equation 19. Formation of the cycle snapshot 181
Equation 20. Cycle-execution state sequence 182
Equation 21. MP1 peak-event index 185
Equation 22. Process-variable alignment at the MP1 event 186
Equation 23. Event-anchored feature vector 186
Equation 24. Conservative disagreement-resolution policy 192
Equation 25. Thresholding of the model probability outputs 193
Equation 26. Normalised decision loss under a relative error-cost ratio 194
Equation 27. Committed operational cycle record 196
Equation 28. Complete traceability record 198
Equation 29. Live monitoring state 201
Equation 30. Run-level processing-latency sequence 202
Equation 31. Ordered drift-monitoring state sequence 205
Equation 32. Set of human-reviewed corrections 207
Equation 33. General retraining-admission gate 207
Equation 34. Conjunctive retraining-eligibility gate 207
Equation 35. Governed Challenger model transition 208
Equation 36. Accepted-cycle validity condition 211
Equation 37. Runtime pacing constraint for reliable signal capture 213

Acronyms and Nomenclature

A. Acronyms

Acronym	Full term	Meaning in this thesis
ABS	Acrylonitrile Butadiene Styrene	Polymer material used in the injection moulding case study
AEI	Advanced Engineering Informatics	Journal associated with the related paper output
AI	Artificial Intelligence	A general term used in the industrial monitoring/analytics context
ANN	Artificial Neural Network	Neural network model used in the predictive comparison
AP	Average Precision	Summary metric for precision-recall performance
AUC	Area Under the Curve	Threshold-independent discrimination metric
BiLSTM	Bidirectional Long Short-Term Memory	Sequence-learning neural-network architecture
CNN	Convolutional Neural Network	Neural-network architecture for pattern extraction
CNN-1D	One-dimensional Convolutional Neural Network	CNN variant applied to one-dimensional sequence data
DAQ	Data Acquisition	Hardware/software system used to capture and process signals
ECE	Expected Calibration Error	Calibration metric used in live evaluation
FN	False Negative	Defective part incorrectly classified as good
FP	False Positive	Good part incorrectly classified as defective
F1	F1-score	Harmonic mean of precision and recall
GB	Gradient Boosting	Tree-based boosting model family
IIoT	Industrial Internet of Things	Industrial connectivity context used in the literature review
IoT	Internet of Things	Connected sensor / monitoring architecture context
ISA-88	International standard for batch control	Referenced in the industrial architecture discussion
ISA-95	International standard for enterprise-control integration	Referenced in the industrial architecture discussion
KS	Kolmogorov–Smirnov statistic	Drift / distribution-comparison metric
LabVIEW	Laboratory Virtual Instrument Engineering Workbench	DAQ / industrial data-acquisition environment referenced in the review
ML	Machine Learning	Data-driven modelling framework used throughout the thesis
MLP	Multilayer Perceptron	Feed-forward neural network model
MQTT	Message Queuing Telemetry Transport	Lightweight industrial messaging protocol
NI	National Instruments	Vendor/platform referenced in DAQ discussion
OOF	Out-of-fold	Cross-validation predictions generated on held-out folds
OPC UA	Open Platform Communications Unified Architecture	Industrial protocol used for live communication and integration

Acronym	Full term	Meaning in this thesis
PCA	Principal Component Analysis	Unsupervised dimensionality-reduction method referenced in the review
PLC	Programmable Logic Controller	Industrial control hardware referenced in the connectivity discussion
PR	Precision-Recall	Evaluation framework emphasising defect-class behaviour
PR AUC	Area Under the Precision-Recall Curve	Threshold-independent summary for precision-recall behaviour
ReLU	Rectified Linear Unit	Neural-network activation function
ROC	Receiver Operating Characteristic	Threshold-based discrimination curve
ROC AUC	Area Under the ROC Curve	Threshold-independent ranking metric
SCADA	Supervisory Control and Data Acquisition	Industrial monitoring/control context referenced in the review
SOTA	State-of-the-art	Best available methods in the literature
SVM	Support Vector Machine	Supervised a nonlinear classification model
TCN	Temporal Convolutional Network	Sequence-model architecture for full-curve learning
TN	True Negative	Good part correctly classified as good
TP	True Positive	Defective part correctly classified as defective
XGBoost	Extreme Gradient Boosting	Optimised boosted-tree model used in predictive comparison

B. Nomenclature

Symbol	Meaning	Unit/note
$\alpha_{i,j}$	Aligned value of the j-th process variable at the MP1 anchor event for cycle i	Event-aligned feature value
Argmax fallback	Direct maximum-index fallback when peak detection fails	Robustness mechanism
$B_i(t)$	Temporary in-cycle buffer for cycle i up to time t	Runtime buffer state
B_{valid}	Validity condition for class balance and representation adequacy in retraining	Governance condition
Baseline run	Stable governed live run without retraining	Evaluation stage
Brier score	Probability calibration error metric	Calibration metric
C_h	Full set of reviewed human corrections	Governance/evidence set
c_j	Individual reviewed human correction j	Governance/evidence item
C_i	Completed and accepted the cycle record for cycle i	Cycle-level governed record
Challenger	Adaptable candidate model evaluated against the Champion	Model role

Symbol	Meaning	Unit/note
Champion	Fixed reference model used during governed operation	Model role
Cycle_Done	Cycle-completion signal used for cycle-boundary detection	Machine-state signal
Cycle_ID(t)	Cycle identifier state at time t	Runtime state
$\mathbf{d_i}$	Drift-monitoring state at cycle i	Monitoring state
D	Full drift-monitoring sequence across the run	Monitoring record
detect_mp_peak (mp,t)	Operational peak-detection function used to locate the MP peak	Peak-finding method
disagree_defect	Locked operational rule under which disagreement implies a defect	Final policy mode
Drift_snapshot	Stored record of current drift status and evidence context	Monitoring artifact
ECE	Expected Calibration Error	Probability calibration metric
event-anchored representation	Representation built around the MP peak and aligned values	Challenger-oriented route
event log	Stored runtime log of skips, drift changes, retrain attempts, and exceptions	Traceability artifact
final label	Final deployed cycle label after applying the locked policy	Decision output
final_reason	Logged reason explaining the final deployed decision	Traceability field
full-curve representation	Representation using pre-processed full-time series channels	Champion-oriented route
$\mathbf{G_i}$	Cycle-completion boundary condition/gate for cycle *i*	Runtime gate
$\mathbf{G_{retrain}}$	Retraining gate decision condition, with 1 = allow update and 0 = block	Governance decision
$\mathbf{G_{run}}$	Governed run context and identity	Governance field
$\mathbf{H_{valid}}$	Validity condition requiring human-reviewed correction evidence	Governance condition
HP(t)	Hydraulic pressure signal	bar
HWT(t)	Hot-water or heating-water temperature signal	°C
I(.)	Indicator function used in gate-condition evaluation	Mathematical operator
$\mathbf{K_{mp1,i}}$	Index at which the selected MP1 event occurs for cycle *i*	Alignment reference
L	Stage-level monitored timing set	Timing/monitoring set
$\mathbf{M_i}$	Live monitoring state at cycle *i*	Monitoring state
$\mathbf{M_{champion}}$ **(.)**	Predictive mapping function of the fixed Champion model	Model function
$\mathbf{M_{challenger}}$ **(.)**	Predictive mapping function of the operational Challenger model	Model function

Symbol	Meaning	Unit/note
$Mnew_{challenger}$ (.)	Updated Challenger mapping function after governed retraining	Updated model function
MP	Generic mould cavity pressure notation	Generic pressure symbol
MP1(t)	Principal cavity pressure signal at sensor/channel 1	bar
MP_peak	Maximum cavity-pressure value within the completed cycle, used as the event anchor.	Event anchor for compact representation
MT	Generic mould cavity temperature notation	Generic temperature symbol
MT1(t)	Cavity temperature signal at sensor/channel 1	°C
Mould_close	Mould-close state signal	Context signal
Mould_open	Mould-open state signal	Context signal
$\eta_{extimators}$	Number of trees in an ensemble model	Hyperparameter
N	Total number of accepted cycles in the operational stage	Run size
NT(t)	Nozzle temperature signal	°C
P95	95th percentile	Used for latency and processing-time assessment
Peak_index	Sample index corresponding to the detected MP peak	Alignment reference
Φ_{ea}(.)	Mapping function generating the event-anchored representation.	Representation function
Φfc(.)	Mapping function generating the full-curve representation.	Representation function
Π(.)	Locked operational policy resolution function	Final decision policy
$P_{champion,i}$	Probability output of the Champion model for cycle *i*	Model score
$P_{challenger,i}$	Probability output of the Challenger model for cycle *i*	Model score
Quality=0	Good part	Binary class label
Quality=1	Defective part	Binary class label
q_i	Acquisition and execution state for cycle *i*	Runtime state
R_i	Committed run-level execution record for cycle *i*	Traceability record
R_{valid}	Validity condition for safety and governance regime checks	Governance condition
run_id	Unique identifier used to separate runs and preserve traceability	Governance field
S_i	Frozen cycle snapshot associated with cycle i	Accepted cycle snapshot
SP(t)	Screw position signal	mm
$t_{end,i}$	Confirmed end time of cycle *i*	Time stamp
$t_{peak,i}$	Event time or relative position of the MP1 peak within cycle *i*	Time/event location

Symbol	Meaning	Unit/note
$t_{proc,i}$	Processing or decision latency associated with cycle *i*	Time/latency
$t_{start,i}$	Confirmed start time of cycle *i*	Time stamp
$\mathbf{T_i}$	Complete traceable evidence record for cycle *i*	Evidence object
Time(t)	Cycle timing signal	Time variable
$\mathbf{V_i}$	Acceptance validity condition for cycle i	Runtime validity gate
$\mathbf{V_{j,i}(t)}$	The j-th process variable in cycle i at time t	Generic process-variable signal
x(t)	Live machine state vector observed at time t	State vector
$\mathbf{y}_{champion,i}$	Discrete class decision of the Champion model for cycle i after thresholding	Binary decision
$\mathbf{y}_{challenger,i}$	Discrete class decision of the Challenger model for cycle i after thresholding	Binary decision
$\mathbf{y}_{final,i}$	Final deployed operational decision for cycle i	Binary deployed decision
$\mathbf{z}_{ea,i}$	Event-anchored feature representation vector for cycle i	Compact feature vector
$\mathbf{z}_{fc,i}$	Full-curve feature representation vector for cycle i	Full-curve feature vector
zscore_max	Maximum absolute z-score across selected event-anchored features	Drift metric
$\mathbf{\Delta t}_{detect,\,max}$	Maximum timing interval for reliable cycle-boundary detection	Timing constraint
$\mathbf{\Delta t}_{exec}$	Effective runtime execution pacing	Timing parameter
$\mathbf{\Theta_c}$	Active decision threshold for the Champion	Threshold parameter
$\mathbf{\Theta_h}$	Active decision threshold for the Challenger	Threshold parameter
$\mathbf{\Lambda}$	L2 regularisation term	Hyperparameter

Chapter 1
Introduction

1.1 Background and Motivation

Modern manufacturing operates under increasing pressure to deliver consistent product quality while improving productivity, reducing waste, and responding to changing production conditions (Kumar et al., 2026). Achieving this consistency remains challenging because manufacturing processes are influenced by interacting process variables and operating conditions that may vary during production. Consequently, apparently repeated manufacturing operations do not necessarily produce identical process behaviour or identical product quality. This creates a need for methods that can detect quality-related changes during production and provide timely information before conventional end-of-process inspection is completed.

Many manufacturing processes are cyclic, with the same sequence of operations repeated to produce successive parts. Each cycle generates a time-dependent history of process signals, while the resulting product quality may still vary from one cycle to another (Cheng et al., 2025). This makes the production cycle a natural unit for linking process behaviour to the corresponding quality outcome. However, important quality-related changes may occur within the cycle and remain unnoticed until later inspection, leading to delayed detection, additional scrap, and unnecessary production cost (Shin et al., 2025).

This challenge has increased interest in digitalised manufacturing approaches that support earlier quality assessment during production rather than relying only on final inspection. However, practical manufacturing use requires more than collecting and displaying process data (Latsou et al., 2024). Process measurements must be connected to cycle-level interpretation, quality prediction, decision support, and later quality evidence. The system must also remain aware that production behaviour can change over time. Models developed from historical data may therefore become less representative of later operating conditions (Shen et al., 2025), creating a need for continued monitoring and controlled model management during runtime operation.

Machine learning offers strong potential for identifying relationships between measured process variables and product quality (Muaz et al., 2025). However, limited model interpretability can reduce transparency and trust when artificial intelligence is used to support manufacturing decisions (Puthanveettil Madathil et al., 2025). Successful offline prediction also does not by itself demonstrate suitability for manufacturing operation (Rajashekarappa et al., 2026). During

runtime use, predictions must be generated within the available cycle time, linked to the correct production cycle, stored with sufficient evidence for later evaluation, and monitored as operating conditions change. If model adaptation is allowed, the conditions under which an update is accepted must also be controlled (Mateo-Casalí, Boza and Fraile, 2026b). The research problem therefore extends beyond predictive accuracy to include runtime reliability, traceability, decision governance, and controlled adaptation.

These conditions motivate the need for a governed real-time architecture for cyclic manufacturing. The practical problem is not only how to predict product quality, but how to connect each production cycle with its process evidence, predictive output, released quality decision, later quality confirmation, and any subsequent model-management action. This requires a structure in which prediction, monitoring, decision-making, and model adaptation remain traceable and are controlled through explicit rules rather than unrestricted automatic updating. The motivation of this research is therefore to support reliable cycle-level quality decision support under realistic operating conditions, including process variability, disturbances, noise, drift, and non-ideal cycles.

Injection moulding is used as the experimental case study because it provides a suitable cyclic manufacturing environment in which repeated production cycles can be clearly identified, measured, and linked to individual quality outcomes. Its process behaviour is influenced by interacting pressure, temperature, timing, material, and machine conditions, while relatively small variations may affect the resulting part quality (Zhao et al., 2024). These characteristics make injection moulding suitable for investigating the wider research problem of how cycle-level process information can be transformed into timely and traceable quality decision support. Although the experimental validation is process-specific, the underlying research problem is relevant to other cyclic manufacturing processes in which repeated cycles can be monitored and associated with product quality.

1.2 Research aims and objectives

1.2.1 Aims

This research aims to propose, implement, and evaluate a governed real-time architecture for cycle-level quality monitoring and decision support in cyclic manufacturing. The architecture integrates cycle-level sensing and

representation, real-time quality prediction, drift-aware monitoring, controlled model adaptation, and traceable operational evidence.

1.2.2 Objectives

To achieve the research aim, four objectives are defined. These objectives follow the progression of the research from experimental evidence and predictive modelling to system implementation and operational evaluation.

1. To establish a cycle-level experimental and analytical basis for quality monitoring by acquiring and organising synchronised multi-sensor process data and examining the relationships between measured process behaviour and part quality.

2. To develop and compare event-anchored and full-curve representations for cycle-level quality prediction, and to evaluate suitable machine-learning models in terms of predictive performance and representation requirements.

3. To design and implement a governed real-time architecture that integrates cycle handling, complementary Champion–Challenger model roles, an explicit Final decision policy, drift-aware monitoring, human correction, controlled model updating, and traceable evidence storage.

4. To evaluate the proposed architecture under runtime operating conditions in terms of predictive behaviour, decision-policy performance, timing and robustness, drift awareness, controlled model adaptation, and its potential transferability to other cyclic manufacturing processes.

1.3 Thesis Contributions

The research presented in this thesis contributes a governed operational architecture for cycle-level quality monitoring and decision support in cyclic manufacturing. Rather than treating quality prediction as an isolated offline modelling task, the architecture integrates cycle-level sensing, complementary

process representations, predictive modelling, explicit decision logic, drift-aware monitoring, controlled model adaptation, and traceable operational evidence within one connected framework. Injection moulding is used as the experimental case study, while the underlying architectural and governance principles are formulated with potential relevance to other cyclic manufacturing processes that share comparable cycle-based monitoring and quality-assessment requirements.

The first contribution is the development of a cycle-level governed real-time architecture for cyclic manufacturing. The proposed framework treats the production cycle as the central unit linking sensing, representation, prediction, decision-making, and operational evidence. This aligns the analytical system with the natural structure of cyclic processes, where quality is associated with the behaviour of individual production cycles rather than with static process settings alone.

The second contribution is the design and implementation of a real-time cycle-level sensing and data-handling framework. The framework captures, synchronises, organises, and validates multi-sensor process data so that each accepted production cycle can be represented consistently for monitoring, prediction, and later quality evaluation. The contribution extends beyond signal acquisition to include cycle-boundary handling, event timing, data integrity, and the preservation of cycle identity throughout runtime operation.

The third contribution is the investigation and comparison of complementary cycle representations for quality prediction. The research evaluates both event-anchored and full-curve representations to determine how different levels of cycle information affect predictive performance and deployment practicality. The event-anchored approach provides a compact representation based on a physically meaningful point within the cycle, while the full-curve route retains the wider time-dependent signal behaviour. Their comparison provides evidence for selecting representations according to both predictive requirements and operational use in cyclic manufacturing.

The fourth contribution is the development and evaluation of machine-learning models for cycle-level quality prediction using the two complementary representations. The modelling stage establishes the predictive basis for the later real-time architecture by evaluating not only classification performance, but also the relationship between predictive capability, input representation, and deployment requirements. This connects offline model development with the

operational roles subsequently assigned to the Champion and Challenger within the real-time system.

The fifth contribution is the incorporation of drift-aware monitoring and governed model management into the real-time architecture. The proposed system does not assume that a deployed model remains valid indefinitely. Instead, it monitors changing operating conditions and model behaviour while keeping adaptation separate from drift detection itself. Any model update is therefore treated as a controlled action that requires defined evidence and governance conditions rather than as an automatic response to detected change.

The sixth contribution is the implementation of a governed Champion–Challenger adaptation strategy. A fixed Champion provides a stable reference, while an updateable Challenger provides the controlled adaptation path. Retraining is permitted only when defined evidence and governance conditions are satisfied, and human-reviewed correction is incorporated as part of the supervision and approval process. This separates model adaptation from unrestricted automatic retraining and preserves traceability of model-management actions during operation.

The seventh contribution is the evaluation of the architecture beyond predictive accuracy alone. The assessment considers runtime behaviour, decision-policy performance, robustness under disturbed operating conditions, drift awareness, processing time, controlled retraining, and traceable evidence generation. This provides an operational evaluation of the complete architecture rather than limiting assessment to offline model performance.

Taken together, these contributions define the thesis as more than a study of quality prediction in injection moulding. The broader contribution lies in the development and evaluation of architectural and governance principles for cycle-level quality monitoring, decision support, and controlled model management in cyclic manufacturing environments.

To make the relationship between the research gaps and the thesis contributions explicit, Table 1 maps each identified gap to the corresponding contribution and to the chapters where the supporting evidence is presented. This mapping clarifies that the thesis is not only an offline machine-learning study, but an integrated contribution to cycle-level representation, real-time deployment, governance, traceability, and controlled adaptation in cyclic manufacturing.

1.4 Overview of Methodology

The research follows a structured methodology that moves from process understanding to evaluation of a governed real-time architecture. The purpose of the methodology is to connect the physical manufacturing process, the captured cycle data, the cycle-level digital representations, the predictive models, and the governed runtime system within one research pathway.

The first stage of the methodology is to define the case-study process and the measurement context. Injection moulding is used to provide the experimental setting, process variables, sensor signals, cycle records, and quality outcomes needed to develop and test the proposed approach. This stage establishes the physical and data-acquisition basis for the later analysis.

The second stage is the preparation of cycle-level data. Raw process signals are organised into cycle records so that each production cycle can be treated as a unit of analysis. This includes attention to cycle timing, signal synchronisation, data integrity, and the link between measured process behaviour and part-quality labels.

The third stage is the construction and examination of cycle representations. The research considers both event-anchored representations and fuller curve-based representations. These representations are used to examine how different forms of cycle information affect interpretability, predictive modelling, and suitability for live deployment.

The fourth stage is statistical and machine-learning analysis. Statistical analysis is used to examine relationships between measured process variables and part quality. Machine-learning models are then developed and evaluated for cycle-level quality classification, with attention to both predictive performance and the practical requirements of later real-time use.

The fifth stage is the implementation of the governed real-time architecture. The selected data-handling, representation, and prediction logic is integrated into a real-time monitoring framework that supports cycle handling, predictive inference, visual monitoring, governed evidence storage, and traceable runtime decision support.

The sixth stage introduces governed operation. This includes monitoring for changing process behaviour, comparing model routes, preserving a stable reference model, controlling any updateable model path, and using human

correction where appropriate. Model adaptation is therefore treated as a controlled and evidence-supported process rather than as unrestricted automatic retraining.

The final stage is the evaluation of the complete framework under realistic operating conditions. The evaluation considers predictive performance, runtime behaviour, robustness under disturbed conditions, drift-related behaviour, governed retraining, and practical suitability for industrial deployment.

Overall, the methodology provides a connected path from experimental process observation to evaluation of a governed real-time architecture. It links data acquisition, cycle-level digital representation, statistical analysis, machine learning, real-time implementation, and governed evaluation in a way that supports the thesis aim without treating the work as only an offline modelling exercise.

The overall architecture, end-to-end monitoring structure, and governed cycle-level workflow developed in this research are summarised in Figures 1–3.

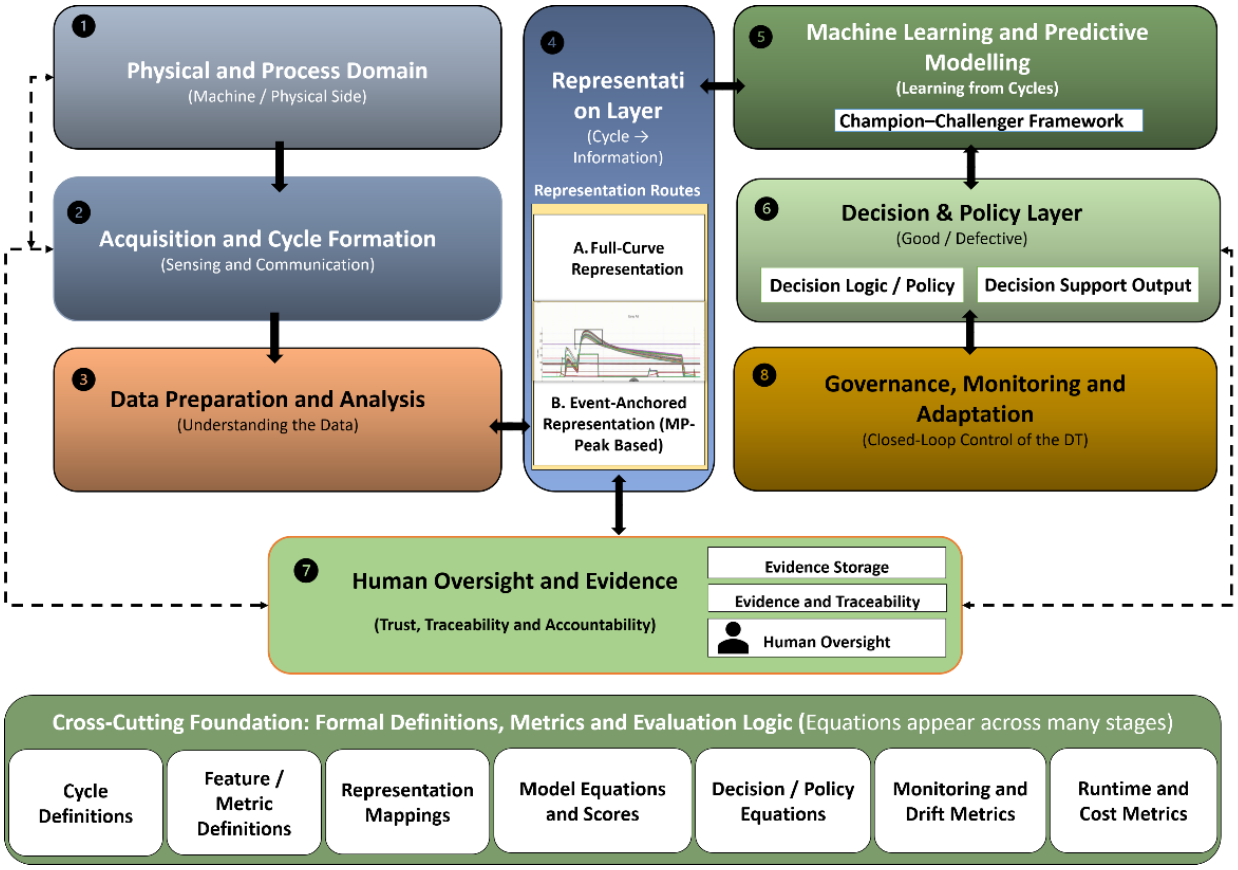


Figure 1. Conceptual governed real-time architecture for cycle-level digital representation, quality monitoring, and decision support using OPC UA acquisition and ML-based classification.

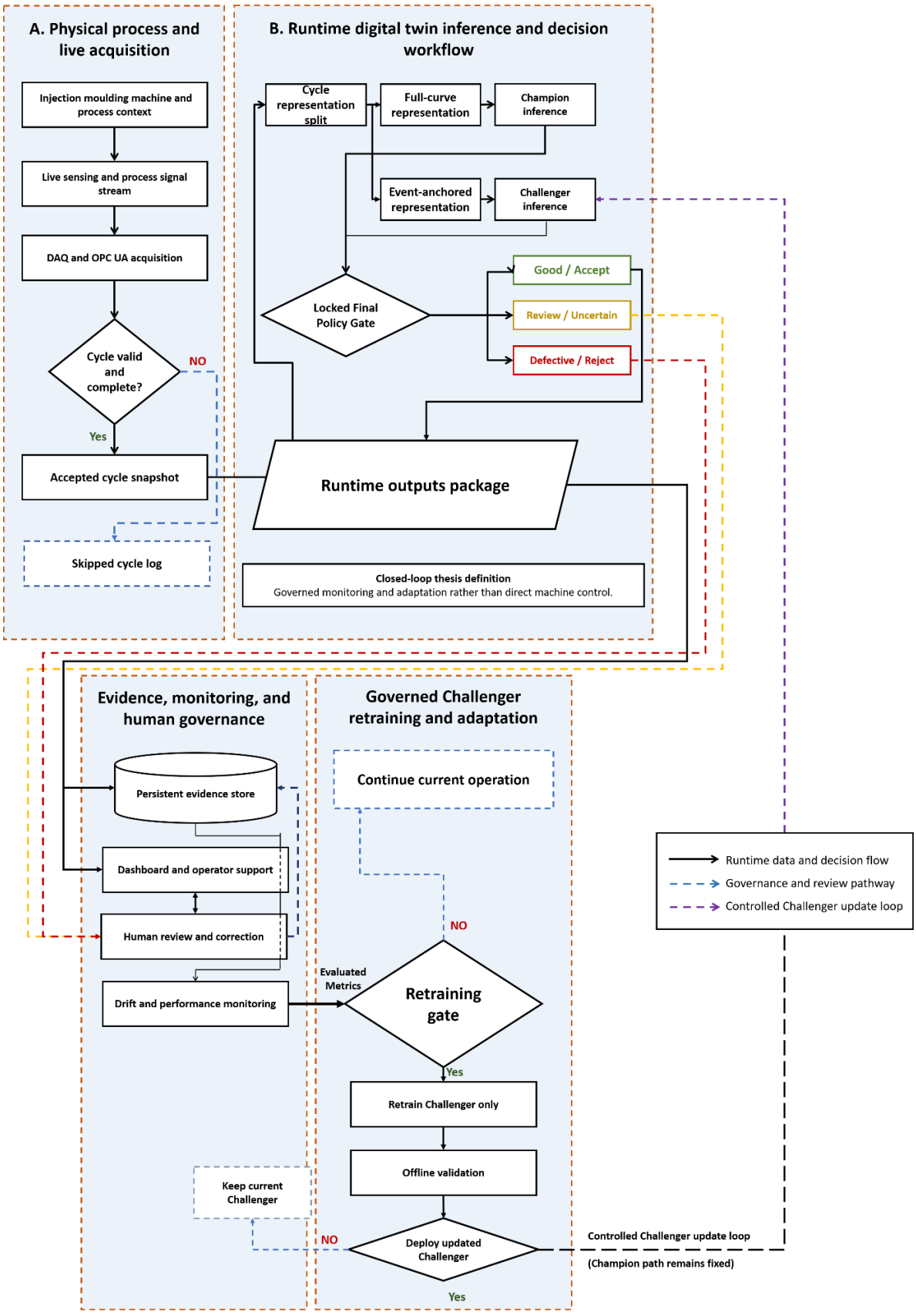


Figure 2. Conceptual governed workflow for cycle-level injection moulding quality monitoring and decision support.

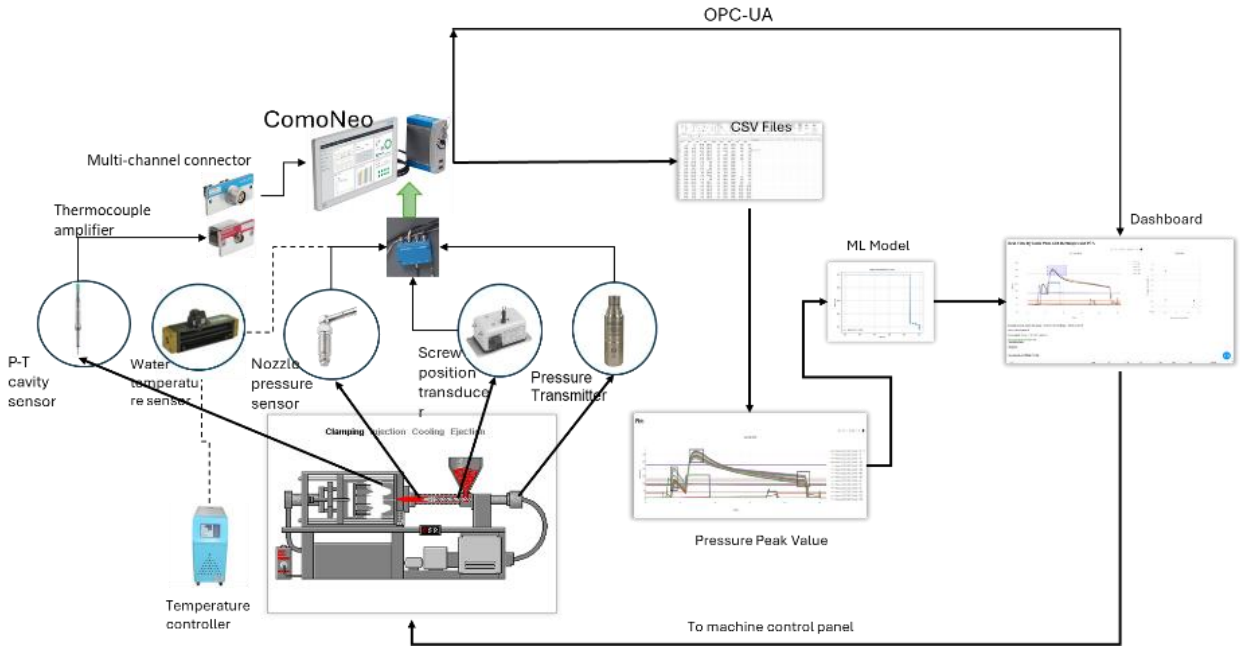


Figure 3. End-to-end monitoring architecture linking in-mould and machine-side sensing, ComoNeo acquisition, OPC UA communication, machine-learning analysis, and dashboard-based decision support in the injection moulding case study.

1.5 Thesis structure

This thesis is organised into nine chapters that move from the general problem of quality monitoring in cyclic manufacturing to the design, implementation, and evaluation of a governed real-time architecture demonstrated through injection moulding. The structure is intended to preserve a clear distinction between the broader methodological contribution of the research and the specific industrial case study used for validation. In this way, the thesis first establishes the scientific and manufacturing context, then develops the experimental and analytical foundation of the work, and finally presents the real-time implementation, evaluation, and overall conclusions of the proposed framework.

Chapter 1 introduces the research background, motivation, aim, objectives, thesis contributions, and overall methodology. It defines the central problem addressed in the thesis, namely, how cycle-level sensing, machine learning, and real-time monitoring can be integrated into a practical and governed architecture for cycle-level quality monitoring and decision support in cyclic manufacturing. It also clarifies that injection moulding is used as the experimental case study through which the proposed framework is developed and evaluated, while the broader

design logic is intended to remain relevant to cyclic manufacturing processes more generally.

Chapter 2 reviews the literature and technological background relevant to the research. It examines cyclic manufacturing as a class of processes in which repeated production cycles generate time-dependent signals that can be linked to quality outcomes. The chapter then reviews digital manufacturing concepts and architectures, sensing, data acquisition, industrial connectivity, machine learning, evaluation, drift, and governed deployment, before identifying the main research gap addressed by the thesis.

Chapter 3 presents the experimental setup and data collection environment used in the research. It describes the injection moulding machine, mould, material system, selected sensors, and the associated data acquisition arrangement. The emphasis in this chapter is on the physical case-study environment and on how process signals were captured from the manufacturing system. In this way, the chapter establishes the practical foundation on which the later analytical and real-time implementation stages are built.

Chapter 4 explains how the raw process signals are organised and transformed into structured cycle-level datasets suitable for analysis and modelling. It describes the preparation of the data, the handling of cycle information, and the development of the two main representations used in the thesis, namely the event-anchored representation and the fuller curve-based representation. The chapter also explains how quality labels are linked to cycle records so that subsequent statistical analysis and machine learning stages can be performed on a structured basis.

Chapter 5 presents the statistical analysis of the measured process variables and their relationship to part quality. This chapter provides process-level interpretation before the machine learning stage and examines how the measured variables relate to part-quality outcomes from an analytical perspective. The findings of this chapter help motivate the later selection of variables, representations, and modelling strategies, while also clarifying the limits of purely statistical interpretation.

Chapter 6 develops the machine learning models for cycle-level quality classification. It describes the preparation of data for modelling, the training and comparison of candidate models, and the offline evaluation of predictive performance. This chapter establishes the predictive modelling basis of the work

and provides the evidence needed to justify the models later selected for integration into the governed real-time environment.

Chapter 7 presents the governed real-time architecture implemented in this research. It explains how cycle-level process data are handled during operation, how the two digital representations are constructed, and how predictions, monitoring outputs, and governed runtime functions are generated in real time. It also introduces the controlled operational structure of the system, including the Champion–Challenger model roles, the explicit Final decision policy, drift awareness, traceable decision logic, evidence storage, human oversight, and controlled model adaptation.

Chapter 8 presents the results and discussion of the thesis from both predictive and deployment perspectives. It evaluates not only classification performance, but also robustness under disturbed conditions, behaviour during runtime operation, drift monitoring outcomes, governed retraining behaviour, decision-policy performance, and computational responsiveness. The chapter also discusses the practical significance of the results, the strengths and limitations of the proposed framework, and its relevance to manufacturing settings beyond the specific case study.

Chapter 9 concludes the thesis by summarising the main findings and contributions, discussing the limitations of the work, and outlining directions for future research. It also reflects on the wider implications of the research for governed real-time quality monitoring and decision-support architectures in cyclic manufacturing and comments on how the proposed approach may be adapted for processes beyond injection moulding.

Overall, the thesis is structured to provide a logical progression from problem definition and literature foundation, through experimental development and predictive modelling, to real-time implementation and practical evaluation. This organisation also helps maintain a clear distinction between the general architectural contribution of the research and the injection moulding case study used to validate it.

Chapter 2
Literature and Technology Review

2.1 Chapter overview and literature review approach

This chapter reviews the literature and technologies that provide the foundation for the research presented in this thesis. The review begins with the wider context of cyclic manufacturing and cycle-level quality monitoring before considering injection moulding as the selected case-study process. It then examines process sensing and data acquisition, industrial connectivity and monitoring architectures, machine-learning methods for manufacturing quality prediction, model evaluation, changing process conditions, concept drift, model reliability, controlled updating and human involvement in model-management decisions. The chapter concludes by synthesising these areas to identify the research gaps that motivate the methodology and system developed in the later chapters.

The literature review follows a structured thematic approach rather than a formal systematic-review protocol. The literature was organised around the main technical themes arising from the research problem and the development of the proposed manufacturing architecture. Relevant peer-reviewed research was examined to establish current approaches, reported capabilities and limitations within each theme, while foundational studies were retained where they were necessary to explain established concepts or methods. Particular attention was given to literature related to manufacturing quality prediction, cyclic process monitoring, sensor-based process representation, machine learning, model reliability and adaptation. The purpose of the review was therefore not simply to describe previous studies individually, but to compare the available approaches, identify limitations in their integration and operational use, and progressively establish the research gaps addressed by this thesis.

2.2 Cyclic manufacturing, real-time monitoring, and the research context

Manufacturing systems are increasingly expected to maintain high quality, low waste, and stable operation under changing production conditions. This requirement is difficult to achieve because process behaviour is shaped by coupled interactions between material response, machine dynamics, sensing quality, environmental variation, and control behaviour. These interactions are especially important in cyclic manufacturing, where the same sequence of operations is repeated and each cycle generates time-dependent signals such as

pressure, temperature, force, position, flow, current, vibration, or acoustic response (Windmann, 2022). Although cyclic processes are repetitive in structure, individual cycles are not necessarily identical in behaviour. Quality variation may develop gradually across successive cycles or arise from short-lived disturbances within a single cycle. Consequently, monitoring approaches based mainly on aggregate or average process values may obscure time-localised behaviour that is relevant to part quality, supporting the need for cycle-resolved analysis.

This makes the production cycle a useful unit for monitoring, interpretation, and decision-making. In cyclic manufacturing, quality is influenced not only by global process settings but also by events occurring at particular points within a cycle, such as a peak response, a transition between phases, a delay in actuation, or a brief signal disturbance. A cycle-level view therefore provides a useful analytical basis because it allows each cycle to be captured, compared, and linked to a corresponding quality outcome (Cerquitelli et al., 2021).

A significant issue in the literature is that much of the reported work remains centred on offline analysis under controlled conditions. Strong predictive performance during model development does not by itself demonstrate suitability for sustained manufacturing operation, where deployment introduces additional challenges associated with integration into live production, dynamic shop-floor data, data quality, model drift, and operational reliability (Heymann et al., 2023; Rajashekarappa et al., 2026). This creates a gap between predictive accuracy demonstrated during model development and the wider requirements of operational manufacturing use.

A related limitation is that existing studies often address separate parts of the monitoring problem. Some focus on sensing technologies, others on feature extraction, classifier comparison, dashboard development, or industrial communication. Fewer studies demonstrate how sensing, data transfer, analytics and monitoring functions can be integrated within one runtime structure for reliable manufacturing process monitoring and decision support (Zhang et al., 2024). In practical manufacturing, this integration is important because a model that performs accurately offline may still have limited value if the surrounding system cannot preserve cycle integrity, manage timing variation, handle disturbed data, or maintain traceable decision records.

These requirements become particularly important when machine learning is used. Manufacturing signals often contain nonlinear and coupled relationships that may not be captured adequately by manually defined deterministic thresholds or rule-based logic. However, high offline accuracy does not guarantee reliable operational performance. Production data may change over time, sensors may drift, and labels may be delayed, uncertain, or corrected. Practical deployment therefore requires not only predictive capability but also structured model deployment, monitoring, governance, and continued iteration, including defined human–system interactions (Safdar et al., 2024).

Cyclic manufacturing provides a suitable setting for such an architecture because each cycle forms a natural unit of measurement, prediction, and decision. The repeated structure allows process behaviour to be analysed at the cycle level while retaining the time-dependent information required for diagnosis and quality prediction (Kozjek, Kralj and Butala, 2017). This thesis therefore takes cyclic manufacturing as the general research scope and develops a governed real-time architecture around this cycle-centred logic. Injection moulding is used as the experimental case study because it provides short, repeated cycles, interacting process effects, and quality-sensitive behaviour that can be observed through measured process signals.

Taken together, the literature suggests that the key research need is not simply the development of another accurate offline classifier. A broader need exists for an operational manufacturing architecture that connects cycle-level sensing, digital representation, prediction, runtime monitoring, decision governance, controlled adaptation, and persistent evidence within a coherent structure. This chapter therefore reviews the literature from this perspective, moving from cyclic manufacturing and the injection moulding case study to sensing, data acquisition, industrial connectivity, machine learning, evaluation, drift, and model governance.

2.3 Injection moulding as the case-study process

Injection moulding is used in this thesis as the experimental case study through which the proposed governed real-time architecture is developed and evaluated. It provides a suitable and demanding example of cyclic manufacturing because production is organised into short, repeated cycles that can be individually identified and associated with corresponding part-quality outcomes. At the same

time, process behaviour is governed by interacting thermal, mechanical, material, and timing-related conditions, while relatively small process variations can affect final part quality. These characteristics make injection moulding particularly suitable for investigating cycle-level quality monitoring because important quality-related information may be concentrated at specific periods or events within the production cycle rather than being represented adequately by average process values alone (Gim and Turng, 2023). Although injection moulding is used for experimental validation, the underlying problem of linking repeated cycle behaviour to quality outcomes is relevant to cyclic manufacturing processes more generally.

At a process level, injection moulding transforms polymer material into finished parts through a repeated sequence of melting, filling, packing, cooling, and ejection. The polymer is first heated to a molten state in the barrel, then injected into the mould cavity under pressure, packed to compensate for shrinkage, cooled until sufficiently stable, and finally ejected before the next cycle begins (Abd Aziz et al., 2015). Although this sequence is well established, the resulting part quality depends on how thermal, flow, and mechanical conditions evolve within and across these stages rather than on the nominal sequence alone.

This makes injection moulding more than a simple thermal process or a simple mechanical process. Thermally, the process depends on heat input, phase change, and controlled heat removal. Melting occurs through barrel heating together with shear heating from screw rotation, while cooling is governed by heat transfer through the mould and its cooling channels. These thermal transitions influence shrinkage, crystallisation, residual stress, and dimensional stability (Fernandes et al., 2018). Mechanically, the process depends on pressure generation, force transmission, screw displacement, and clamping behaviour. During filling and packing, the machine must deliver the melt with sufficient and stable pressure while the mould remains securely closed. Any instability in these mechanical conditions can contribute directly to quality variation and defect formation (Narowski and Wilczyński, 2024).

The flow behaviour of the molten polymer adds further complexity. The melt behaves as a non-Newtonian shear-thinning fluid, so its viscosity changes with both shear rate and temperature. As it passes through runners, gates, and cavity sections, it experiences changing pressure gradients and thermal conditions. This means that flow behaviour is sensitive not only to machine settings but also to local geometry, thermal balance, and material condition. Variations in these

factors can influence cavity filling, pressure transmission, weld-line formation, air entrapment, and final dimensional accuracy (Ozdemir, Uluer and Guldas, 2004; Kurt et al., 2009).

Material behaviour is equally important. Melt viscosity, crystallinity, thermal stability, molecular orientation, moisture content, and filler content can all influence how the polymer responds during filling, packing, and cooling. Problems in material preparation or conditioning may not appear immediately, but can later become visible as warpage, sink marks, silver streaks, or other quality defects (Chaciński and Sutowski, 2021; Mourya et al., 2023). For this reason, injection moulding should not be interpreted as a fixed machine recipe alone. It is better understood as a coupled cyclic process in which machine action, thermal history, flow response, and material behaviour interact within each cycle.

From the perspective of this thesis, the main importance of injection moulding is not only that it is industrially significant, but that it provides a strong validation setting for cycle-level monitoring and decision support. Each cycle is short, repeated, and rich in time-dependent information. At the same time, product quality is sensitive to subtle disturbances in pressure, temperature, timing, and material response. This makes injection moulding a suitable case study for evaluating real-time process monitoring, machine-learning-based quality prediction, and decision support using production-process data (Lin and Chen, 2024).

Injection moulding is treated here as a demanding case-study process within the broader scope of cyclic manufacturing. The purpose is not to limit the thesis to one process only, but to use a well-instrumented and defect-sensitive environment to develop and examine methods that may inform other cyclic manufacturing settings where repeated cycle behaviour can be measured and linked to quality outcomes.

2.4 Defects, cycle-level behaviour, and quality formation

Part quality in injection moulding is influenced by a wide range of defects that arise from coupled interactions between machine settings, material behaviour, mould design, and cycle dynamics. Relatively small changes in pressure history, temperature history, filling behaviour, or cooling balance may produce different

defect modes in parts that are otherwise made under nominally similar settings. This makes defect formation important not only as a product-quality issue, but also as a process-behaviour issue. From a cyclic manufacturing perspective, defects can be interpreted as cycle-level outcomes that reveal how the process behaved during key moments within a repeated production sequence (Zhang and Zhan, 2026).

Injection moulding is commonly described through critical phases including plasticating, filling, packing, cooling, and ejection, with each phase influencing the final quality of the moulded part (Gaspar-Cunha et al., 2025). Fernandes et al. (2018) described the injection-moulding cycle in terms of filling, packing, and cooling, with pressure maintained during packing to compensate for shrinkage and cooling continuing until the part is sufficiently rigid for ejection. Taken together, these accounts show that quality cannot be understood from a single signal or process stage alone. Effects generated earlier in the cycle may only become visible later during packing, cooling, or final part inspection.

This helps explain why defect diagnosis is difficult in practice. The relationship between process input and part outcome is often nonlinear and strongly coupled. Vukovic (2024) highlighted the difficulty of identifying which parameters most strongly influence quality when multiple inputs interact. Cao et al. (2025) similarly stressed that machine settings alone do not explain variation adequately, because mould design and material behaviour during filling and cooling also shape the outcome. Defect formation is therefore often linked to dynamic cycle behaviour rather than to one incorrect fixed setpoint (Chen, Chen and Huang, 2025).

Several recurring injection-moulding defects include short shots, sink marks, flash and burn-related defects, all of which are strongly influenced by interactions among process parameters (Köken, Firuzan and Yavuz, 2026). These defects are useful in research because they can often be linked to process physics. A short shot is commonly associated with insufficient filling capability and may result from low melt temperature or inadequate pressure transmission. Sink marks are generally linked to shrinkage and insufficient packing compensation, especially in thicker regions. Warpage is often associated with uneven cooling and residual stress gradients. Weld lines and flow marks are more closely related to flow-front behaviour and thermal imbalance during filling. Flash and burn marks tend to reflect mechanical closure, venting, or excessive local pressure, while silver streaks highlight the importance of material preparation and gas management

(Chaciński and Sutowski, 2021; López-Adrio and Álvarez, 2024). However, the same visible defect may arise from different combinations of causes depending on part geometry, polymer type, mould condition, and operating regime.

This point is important because it shows that defect categories should not be treated as simple one-cause one-effect outcomes. Chen and Turng (2005) described injection moulding through hierarchical control levels, including machine control, process control, and quality control. This hierarchy is useful because some defects are influenced more strongly by actuator or hardware behaviour, others by cavity-level thermal and flow behaviour, and others by the final condition of the part detected during inspection or use. The literature suggests that defects are better understood as the visible result of interactions across multiple layers of the process rather than as isolated responses to single variables. Zhao et al. (2020) provide a broad review of sensing, process-parameter optimisation, and control in intelligent injection moulding, demonstrating the importance of integrating information across these areas. However, as a review of enabling methods rather than an evaluation of a single sustained runtime architecture, the work does not itself demonstrate how cycle-level sensing, quality prediction, decision governance, and controlled model management can be integrated and evaluated during operation.

From a monitoring perspective, the main implication is that defects are often linked to specific moments or transitions within the cycle. The filling-to-packing transition, the pressure response during packing, and the thermal behaviour before ejection may all influence whether a part remains acceptable or becomes defective. This supports the use of cycle representations that focus on timing, shape, peaks, and transition behaviour rather than relying only on average values or fixed machine settings (Liew et al., 2022). It also explains why cycle-based monitoring is more informative than end-of-line inspection alone. End inspection may identify that a defect exists, but cycle-level signals may help explain when and how the conditions leading to that defect developed.

Control strategies in the literature also reinforce this interpretation. Injection-moulding defects including flow lines, burn marks, warpage, sink marks and short shots are associated with combinations of machine and process parameters, requiring process-parameter adjustment rather than attribution to a single cause (Mourya et al., 2023). This is consistent with the broader view that quality improvement depends on restoring stable cycle behaviour in which pressure, flow, and temperature evolve in a coordinated way (Huang, Nian and Lin, 2021).

The literature, therefore, supports a broader analytical claim relevant to this thesis. In cyclic manufacturing, defects are not only part-level outcomes. They are also evidence of how the repeated cycle behaved over time. This makes defect analysis important for cycle-level monitoring and quality prediction. If quality problems emerge from time-specific cycle behaviour, then an effective monitoring architecture must represent cycle evolution, link it to final outcomes, and detect when the process begins to move away from stable operation. Injection moulding provides a strong case study for this because defects such as short shots, flash, warpage, sink marks, and weld lines are clearly influenced by how pressure, temperature, flow, and cooling evolve within individual cycles. The same cycle-centred principle is also relevant beyond injection moulding, as cycle-synchronised sensor data from repetitive manufacturing operations can preserve process context and support the detection of changes in operating condition (Esmaili et al., 2025).

2.5 Sensing, in-mould monitoring, and data acquisition for cyclic manufacturing

Reliable real-time monitoring in manufacturing depends first on the ability to capture process behaviour in a consistent and time-resolved form. For cyclic processes, this requirement is especially important because quality is often influenced by short-lived events within repeated production cycles rather than by average conditions alone. Sensing and data acquisition, therefore, do more than provide background information. They determine whether cycle behaviour can be represented accurately enough for monitoring, prediction, and later decision-making (Le Goff et al., 2025; Masato, Kazmer and Panchal., 2023).

The literature shows that sensing in manufacturing can be considered at several levels. Machine-side variables, such as screw speed, injection pressure, hydraulic pressure, position, actuator state, and controller outputs, describe how the machine drives the cycle. In-mould variables, such as cavity pressure and cavity temperature, capture how the material responds inside the tool and are often more directly linked to part formation. Final part indicators, such as dimensions, weight, surface quality, or functional performance, reflect the ultimate production outcome. This layered view is useful because it shows that no single source of information is sufficient in all cases. Machine signals may indicate what the

system attempted to do, while in-mould signals may show what happened at the material level (Zheng, 2025).

Sensor type and placement strongly affect the usefulness of monitoring. In manufacturing, non-contact sensors such as optical, infrared, and ultrasonic devices are often used for detection, tracking, and inspection, while contact sensors such as pressure, temperature, and strain sensors are used where internal process conditions must be measured directly. In injection moulding, this distinction is particularly important because variables inside the mould cavity are often more informative for quality prediction than machine-side settings alone. As a result, in-mould sensing has gained attention as a means of observing filling, packing, and cooling behaviour more directly (Haberchad and Salih-Alj, 2025).

Among available measurements, cavity pressure is often treated as one of the most informative signals because it reflects the combined influence of flow resistance, packing effectiveness, and machine actuation. Tsai and Lan (2015) and Pramujati, Dubay and Samaan (2006) showed that pressure profiles are closely related to outcomes such as warpage, shrinkage, and incomplete filling. Cavity temperature also provides important information because thermal behaviour influences viscosity, solidification, residual stress, and final part stability. Other sensing approaches, including strain gauges, ultrasonic methods, fibre Bragg grating sensors, and multi-variable sensor designs, can provide richer process insight, but they also increase integration and calibration complexity (Ageyeva, Horváth and Kovács, 2019; Gordon et al., 2015; Kazmer et al., 2011). The literature therefore suggests that sensing design should not be judged only by how many variables are measured, but by whether the selected measurements provide stable and interpretable evidence of cycle behaviour.

This is where data acquisition becomes critical. Even when informative sensors are installed, a monitoring system may still perform poorly if signals are sampled too slowly, aligned incorrectly, or buffered unreliably. Accurate real-time acquisition is essential because many quality-relevant events occur over short periods within the cycle. Reliable real-time data acquisition is essential because it provides the process measurements required for monitoring melt behaviour and supporting adaptive quality control in injection moulding (Chang et al., 2025). Limitations in sampling rate, synchronisation, buffering, and communication can prevent a system from capturing the most informative parts of the cycle, which means that prediction quality may be constrained by the pipeline before model training even begins.

The literature shows several DAQ directions. Low-cost IoT-based systems are attractive because of their affordability, ease of deployment, and remote accessibility. Mgbemena and Okeagu (2023) used an ESP32-based DAQ system with infrared and ultrasonic sensors for predictive maintenance, while Babalola (2021) described a Raspberry Pi approach for basic acquisition and monitoring. These examples demonstrate that low-budget platforms can support useful monitoring when signal behaviour is relatively slow and occasional delay is acceptable. For fast cyclic processes, the temporal resolution of data acquisition becomes critical because rapid within-cycle events can otherwise be obscured. Massano et al. (2024) demonstrated dynamic injection-moulding measurements using a 0.05 s data cycle, showing the importance of sufficiently fast acquisition for capturing transient process behaviour.

Edge and embedded DAQ systems provide a stronger option for time-critical monitoring because sensing, preprocessing, and control logic are placed close to the machine. Arslan, Konuşkan and Lazoglu (2025) implemented cycle-level cavity-pressure monitoring using piezoelectric sensors, with data acquired at 5 kHz and processed through an edge-computing architecture for real-time visualisation and analysis. This type of architecture can reduce communication delay and support event-synchronised operation. However, reliable process monitoring still depends on measurement methods that can identify machine status and characterise transient melt-flow behaviour, because machine and melt conditions can fluctuate from shot to shot (Zhao et al., 2024).

Industrial injection-moulding studies have used complementary data-acquisition approaches for machine signals, in-mould measurements, and part-quality inspection. Aminabadi et al. (2022) acquired machine and peripheral data through OPC UA, while analogue machine and mould signals were recorded using HBM acquisition hardware and CATMAN software. LabVIEW was used for the automated part-weighing station. Proprietary systems such as Kistler ComoNeo have also been used for cycle-synchronised in-mould acquisition. Werner et al. (2020) used ComoNeo within a digital assistance system for continuous cycle-based acquisition and self-learning quality evaluation, and Loftis, Farahani and Pilla (2020) showed how such systems can support feature extraction and regression modelling from structured sensor streams. These studies illustrate an important point, acquisition becomes most valuable when it is linked directly to monitoring and decision support rather than treated only as a storage function.

Across the literature, several recurring requirements appear. First, signal quality depends not only on nominal sensor specification, but also on calibration, mounting, amplification, and local installation conditions. Second, synchronisation and cycle alignment are essential because comparisons between cycles require the same physical events to be represented consistently in time. Third, robustness under industrial behaviour remains a major challenge. Machine interruptions, communication loss, sensor glitches, and partial cycles are common in practice, and DAQ systems must manage these without corrupting later analysis. Finally, scalability depends on whether signals can be transferred and stored in a structured and traceable form, allowing later connection to monitoring software, predictive models, and evidence logging (Radlbauer, Moser and Wagner, 2025).

For this thesis, the literature leads to a clear analytical conclusion. Sensing and data acquisition should not be treated as preliminary technical details. They are core determinants of whether a real-time monitoring and decision-support architecture can operate reliably in cyclic manufacturing. If the system cannot capture cycle behaviour in a complete, aligned, and traceable form, then later prediction, drift monitoring, and governed decision-making will remain weak regardless of the sophistication of the model. Injection moulding provides a strong case study for this issue because its quality-relevant behaviour is concentrated in short, repeated, and highly sensitive cycles. At the same time, the same sensing and acquisition logic is relevant more broadly to cyclic manufacturing processes in which cycle-level behaviour must be observed before it can be analysed and used for quality-related decision support. Reliable acquisition alone is therefore insufficient; cycle-resolved data must also be transferred, structured, and retained without losing temporal alignment or traceability. These architectural requirements are examined in Section 2.6.

2.6 Industrial connectivity and monitoring architectures

Reliable monitoring in manufacturing depends not only on sensing and data acquisition, but also on how data is moved, structured, stored, and made available for decision-making. In connected production environments, this wider infrastructure is often described through Industrial Internet of Things architectures, industrial communication protocols, and supporting data services.

These elements are important because a monitoring system does not operate as a single sensor or model in isolation. It operates as a connected structure in which data must travel from the physical process to software layers that support interpretation, prediction, and action (Permin, Wohlgemuth and Keller, 2025).

The literature often presents IIoT architectures as a key part of smart manufacturing because they support continuous acquisition, remote accessibility, and integration between shop-floor devices and higher-level analytics. Charest, Finn and Dubay (2018) presented an approach for integrating machine learning at the controller level of an injection moulding machine. Their initial experimental evaluation demonstrated the potential of data-driven methods to improve process-controller performance. Typical building blocks include smart sensors, edge devices, gateways, cloud or local storage, and analytics services. Cheng et al. (2018) proposed a 5G-enabled Industrial Internet of Things (IIoT) architecture for smart manufacturing, emphasising the importance of high data-transmission rates, communication reliability, and low latency for real-time industrial data exchange. These communication requirements are particularly relevant to manufacturing monitoring systems, where the timely and reliable transmission of sensor information is necessary to support operational analysis and decision-making. This is an important point for manufacturing systems, because connectivity alone does not create industrial value unless the transferred data remains timely, complete, and usable for operational decisions.

A recurring weakness in the literature is that connected architectures are sometimes discussed mainly in terms of remote access and system flexibility, while timing and reliability receive less attention. Junker and Domann (2017) emphasised that connected monitoring can support more efficient control of complex production processes through automated data-driven adjustment. In practice, however, this benefit depends on more than connectivity. If data transfer is delayed, incomplete, or unstable, monitoring outputs may arrive too late or may generate unreliable alarms, especially in fast cyclic processes where important events occur in short time windows (Turkmanović et al., 2023). For this reason, industrial monitoring architecture should be judged not only by whether components are connected, but also by whether the communication path preserves timing behaviour suitable for the process being observed.

Industrial communication standards play a central role in this architecture. OPC Unified Architecture has become one of the most widely used standards because it supports secure, platform-independent communication across automation and

business layers. Virta et al. (2010) described OPC UA as supporting secure communication and service-oriented integration, while also aligning with wider manufacturing structures such as ISA-88 and ISA-95. In injection moulding, EUROMAP 77 defines a standardised OPC UA interface for exchanging data between injection moulding machines and Manufacturing Execution Systems (MES), supporting interoperability between equipment and software from different manufacturers (EUROMAP, 2020). In contrast, EUROMAP 63 specifies a file-based data exchange interface using ASCII request and response files for communication between injection moulding machines and host computer systems (EUROMAP, 2000). These developments show that industrial communication is not only a technical convenience. It is a necessary condition for building monitoring systems that can exchange structured data across machines, controllers, and higher-level software.

Alongside OPC UA, lighter communication approaches are also widely used. MQTT is commonly adopted in manufacturing IoT systems because its publish-subscribe architecture provides lightweight and efficient communication, with performance commonly assessed in terms of latency, scalability, and resource utilisation (Saha et al., 2024). MQTT can therefore be useful when many sensor nodes or frequent updates are involved. However, lightweight communication also raises an important system-design question. Different industrial signals tolerate different levels of delay and message loss. Some monitoring tasks may accept occasional missing samples, whereas cycle-based fault detection and quality decisions may require stricter completeness, ordering, and delivery assurance (Permin, Wohlgemuth and Keller., 2025). The literature, therefore, suggests that protocol selection should follow the requirements of the monitoring task rather than the popularity of the protocol itself.

This point is reinforced by studies that combine multiple tools rather than relying on a single protocol. Akin et al. (2021) described architectures in which machine data is published using MQTT, managed through OPC middleware such as Kepware, and routed through tools such as Apache NiFi towards analytics modules for quality monitoring and predictive maintenance. Kusić and Slapšak (2020) described a TCP/IP and MQTT-based injection moulding monitoring architecture in which sensor data was transferred to cloud storage and visualised through Grafana. These examples show that industrial monitoring systems are often assembled from several communication and software layers. Their practical success depends less on any one protocol in isolation and more on whether the

whole data path supports stable sampling, structured transfer, suitable storage, and useful access for later analysis.

Web services and databases form another important part of this infrastructure because they manage how information is stored, queried, and exchanged between system components. Web services and application programming interfaces can provide standardised interfaces through which industrial applications access heterogeneous manufacturing equipment and exchange operational data (Kim and Lee, 2025). Cheng et al. (2017) similarly described web services as loosely coupled and platform-independent applications that support data interchange across distributed systems. In manufacturing, this enables sensor and controller data to move beyond local capture and become available to dashboards, analytics modules, and supervisory tools. Lee, Ryu and Cho (2017) also showed that such service layers are often linked to control-oriented systems such as SCADA, which operate close to the shop floor and provide the bridge between raw process signals and wider monitoring or decision-support functions.

Storage design is equally important. As process data volume increases, database choice affects what forms of monitoring and analysis remain practical. Behnke et al. (2018) noted the growing importance of scalable data storage for real-time communication and analytics. For manufacturing monitoring, time-series databases provide a suitable structure for storing and retrieving continuously acquired process data and supporting subsequent monitoring and visualisation (Garcia et al., 2024). The analytical point here is that storage is not a passive detail. It shapes whether data remains queryable, reproducible, and auditable over long production periods. Without a consistent and structured data-management design, manufacturing data may become difficult to integrate, retrieve, and reuse reliably for subsequent analysis and decision support (Radlbauer, Moser and Wagner, 2025).

Traceability has become increasingly important in connected manufacturing because production data must be linked reliably to the corresponding workpiece and processing conditions to support quality assurance, later analysis, and evidence of manufacturing history (Reinhardt et al., 2024). Although such approaches are still developing, they highlight a broader industrial concern. Data handling is now part of system credibility. Unclear provenance, missing traceability, or unreliable storage can reduce confidence in monitoring outputs even when the prediction model itself appears technically strong (Kuhn and Franke, 2021).

Taken together, the literature suggests that industrial connectivity should not be treated as a background technical layer added after sensing and modelling. It is a core part of monitoring architecture because it determines whether process information can move reliably from the physical system to the analytical and decision layers. At the same time, existing work often focuses on interoperability and connectivity more than on cycle-level timing, data integrity under disturbance, and traceable runtime behaviour. This limitation is important for the present thesis. For manufacturing, cyber-physical production architectures integrate physical production resources with digital technologies and analytics to support smarter and more responsive operational decision-making (Kumar et al., 2026). With a reliable data pathway in place, attention shifts to how cycle-level measurements can be transformed into meaningful quality predictions. Section 2.7 examines the machine-learning methods used for this purpose and their limitations in manufacturing applications.

2.7 Machine learning for quality prediction in manufacturing

Machine learning has become increasingly important in manufacturing because it can capture complex relationships between process behaviour and product outcome that may be difficult to represent using manually specified deterministic thresholds or rule-based decision logic alone. In quality prediction applications, machine learning is commonly used to classify part condition, estimate defect probability, or identify process states associated with poor production performance. Recent injection-moulding studies have applied both conventional machine-learning and deep-learning methods for quality prediction, including Random Forest, XGBoost, CatBoost and sequence-based neural architectures, with strong performance reported on real production-cycle data (Punyangarm and Chotayakul, 2025). Much of the literature has focused on supervised learning because many industrial settings now maintain labelled quality records or inspection outcomes. In such cases, models are trained to associate measured process behaviour with acceptable and defective parts. Recent injection-moulding research has applied supervised machine-learning models directly to quality-related process data. Tayalati et al. (2025) trained Support Vector Machine and Random Forest models to predict volumetric shrinkage from injection-moulding process parameters, demonstrating the suitability of

supervised learning for modelling nonlinear relationships between process conditions and product quality.

Unsupervised learning remains relevant, although its role is usually different. Methods such as clustering and Principal Component Analysis are more often used for anomaly detection, feature reduction, or exploratory structure identification rather than direct quality classification. This distinction is important because it reflects a broader pattern in manufacturing research. Supervised methods are preferred when reliable quality labels are available, whereas unsupervised methods are more useful when the objective is to detect abnormal behaviour, reduce dimensionality, or identify hidden structure in the data before classification is attempted. In practical systems, both directions may be useful, particularly when labels are delayed, incomplete, or uncertain (Cohen and Ni, 2021). Label reliability also presents a distinct challenge for supervised industrial defect recognition. Liu et al. (2025) proposed a parallel hybrid penalty network to improve defect recognition under noisy training labels by combining sample selection, feature learning, and penalty-based learning. Although evaluated on industrial defect-recognition datasets rather than injection-moulding cycle data, the study demonstrates the importance of accounting for annotation uncertainty when developing robust quality-classification models.

The effectiveness of machine learning depends not only on the choice of algorithm but also on the nature of the dataset and how information is processed. Several studies have shown that high predictive performance can be achieved with moderate datasets, which challenges the assumption that large datasets are always required. Chen, Guo and Chang (2023), for example, reported precision above 90% using a relatively small collection of injection cycles. This is particularly relevant in injection moulding, where each cycle can generate dense temporal information from pressure, temperature, and machine-state signals. In such cases, the information richness within each cycle may compensate partly for the limited sample count.

Representation choice is therefore central to model performance. Bogedale et al. (2023) showed that high-resolution time-series data produced significantly better injection-moulding quality predictions than scalar process data alone. This suggests that data depth can be as important as data volume. At the same time, a full-cycle representation is not always the most practical option for deployment. One group of approaches uses complete time-series information, allowing the model to learn from the whole signal shape. Other uses reduced

representations based on selected features, summary statistics, or event-anchored values extracted at key moments in the cycle. Full-cycle methods can capture richer process behaviour, but they increase computational demand and may be more sensitive to missing samples, misalignment, or variable cycle length. Reduced representations are often more practical for real-time monitoring, but they may omit information outside the selected features or event points (Ding et al., 2025). The choice between these representations is therefore not only a modelling preference. It affects real-time feasibility, robustness, and interpretability.

Previous injection moulding studies demonstrate that physically meaningful characteristics extracted from process curves can provide useful alternatives to using complete time-series signals. Chen, Yang and Huang (2018) derived quality-related indices from pressure signals, including peak pressure, pressure gradient, viscosity, and energy indices. Ke and Huang (2020) subsequently used pressure-derived quality indices, including peak-pressure information, as inputs for machine-learning-based quality prediction. Gim and Rhee (2021) represented the cavity-pressure profile through process state points corresponding to physically meaningful stages of the moulding cycle, including the filling stage, filling-to-packing transition, maximum cavity pressure, end of packing, and end of cooling. Gülçür and Whiteside (2021) similarly demonstrated that process fingerprints, including peak, derivative, integral, and comparative quantities extracted from process signals, can provide compact physical proxies for quality prediction.

These studies establish an important foundation for the present research: neither the use of cavity pressure nor the extraction of peak- or event-related process information is new in itself. However, the reviewed approaches primarily construct individual pressure-derived indices, characteristic process-state points, or channel-specific process fingerprints. In the studies reviewed here, a single physical event in one reference signal is not used as a common temporal anchor for synchronously sampling the state of the other monitored process variables and forming one compact multivariable cycle representation. This distinction provides the basis for the event-anchored representation examined in this thesis. The highest cavity-pressure peak is used as the reference event, and the simultaneous state of the monitored process variables at that event is retained as a compact representation for quality prediction. Its value is subsequently

assessed through direct comparison with a full-curve representation derived from the same cycle-level data.

This issue is especially important in injection moulding, where many defects arise from coupled thermal, pressure, and timing effects rather than from one variable acting in isolation. Chen, Nguyen and Tai (2018) identified two persistent challenges in the process, determining suitable initial process parameters and making timely adjustments when defects occur. These challenges support the need for adaptive learning systems that can extract useful information from cycle data and remain informative under changing production conditions. From a broader manufacturing perspective, they also show why machine learning should be treated as part of an operational system rather than as a standalone classification exercise.

The wider manufacturing literature has increasingly linked machine learning with big-data infrastructure. Demchenko et al. (2013) identified volume, velocity, variety, value, and veracity as key characteristics of big data. These characteristics are relevant to industrial environments where heterogeneous sensor streams are continuously generated and require reliable data management. Akin et al. (2021) presented an industrial big-data architecture for automotive manufacturing, integrating real-time on-premises data acquisition, storage, and processing using Apache Kafka, Apache Spark, and Hadoop. Their work demonstrated the infrastructure requirements and implementation challenges associated with managing large-scale manufacturing data. This suggests that machine learning deployment is not only a modelling problem. It is also a data-management and system-integration problem, particularly when models must operate on live streams rather than static datasets.

Open-source tools have helped make industrial machine learning more accessible. Landset et al. (2015) described Hadoop as a framework for scalable storage and parallel processing, while also noting the limitations of MapReduce for iterative machine learning tasks. Related toolkits such as Mahout, MLlib, H2O, and SAMOA have supported model development across languages, including Java, Python, R, and Scala (Landset et al., 2015). These developments show that the software ecosystem has expanded considerably, but they also indicate that technical accessibility, workflow usability, and scalability remain practical barriers when moving from laboratory experimentation to industrial deployment.

A more important limitation is that much of the literature still focuses on offline predictive performance rather than runtime behaviour. Many studies demonstrate that a model can classify historical data effectively, but fewer show how performance changes when signals are noisy, cycle boundaries shift, labels are delayed, or operating regimes evolve over time (Prioli et al., 2025). In such situations, a model that appears strong offline may still be difficult to trust in real production. The key issue, therefore, is not only which algorithm achieves the highest offline score, but how machine learning can be embedded within a reliable production system. In cyclic manufacturing, this means that models must operate on cycle-based data structures, remain useful under changing operating conditions, and support decisions that can be explained and traced (Jourdan et al., 2023). Injection moulding provides a strong case study for this problem because it offers rich cycle-level information together with defect-sensitive process behaviour. At the same time, the same challenge extends more broadly to other cyclic manufacturing processes in which repeated signal patterns must be linked to quality outcomes. The literature therefore supports machine learning as an important enabling method, but also shows that prediction alone is insufficient without reliable data representation, evaluation under changing conditions, and governance during deployment (Heymann et al., 2023). Accordingly, industrial suitability cannot be judged from headline offline scores alone. Evaluation must also consider threshold behaviour, calibration, time-dependent performance, execution latency, and the operational consequences of different prediction errors, as discussed in Section 2.8.

2.8 Evaluation metrics, operational performance, and decision cost

Evaluating machine learning models for manufacturing quality prediction requires more than reporting a single accuracy value. In many production settings, defects are relatively rare compared with acceptable parts, and the practical cost of different errors is not the same. A model may therefore appear strong on accuracy while still producing unacceptable false approvals or excessive false rejects. For this reason, evaluation should account for both classification performance and operational consequences. He et al. (2023) highlighted the challenges associated with imbalanced manufacturing datasets, while Gerling et al. (2022) demonstrated that cost-based evaluation metrics can

reveal operational differences that conventional classification metrics may overlook.

A common starting point is the confusion matrix, which separates outcomes into true positives, true negatives, false positives, and false negatives. In manufacturing quality classification, false positives usually correspond to acceptable parts that are incorrectly rejected, increasing scrap or rework costs. False negatives correspond to defective parts that are incorrectly accepted, which can create higher downstream costs through warranty claims, customer complaints, safety concerns, or reputational damage. Because the impact of these two error types is often very different, the confusion matrix provides a more transparent view of model behaviour than accuracy alone (Sankhye and Hu, 2020). Precision, recall, and F1-score are derived directly from the confusion matrix and are often more informative under class imbalance (Fathy, Jaber and Brintrup, 2020). Precision shows how many predicted defects are truly defective, while recall indicates how many actual defects are successfully detected. F1-score provides a balanced summary when both types of error matter, but it may still hide whether performance is dominated by false positives or false negatives. For this reason, it is more informative when reported together with the confusion matrix rather than as an isolated result (Lobo, Sampaio and Novais, 2024).

Many quality-prediction models produce probabilities rather than fixed labels, which makes threshold behaviour important. Receiver operating characteristic curves are widely used to show how model performance changes across thresholds by plotting true positive rate against false positive rate. The area under the ROC curve is often used as a threshold-independent summary of ranking ability. However, ROC curves can appear favourable even in imbalanced problems because the false positive rate is normalised by the size of the negative class. When the number of acceptable parts is very large, many false positives may still produce a visually low false positive rate. For this reason, precision-recall curves are often more informative in manufacturing quality problems where defects are relatively rare. PR curves show the trade-off between precision and recall as the threshold changes, and PR-AUC is usually more sensitive to defect-class performance. In practical terms, PR-based evaluation is often more closely aligned with the operational question of how many predicted defects are meaningful and how many true defects are actually captured (Lee, Lee and Kim, 2020; Kafunah, Ali and Breslin, 2021; Köken, Firuzan and Yavuz, 2026).

Discrimination ability alone is not sufficient when probabilities are used in operational decisions. Calibration is also important because a model may rank parts correctly while still outputting probabilities that are systematically too high or too low. Reliability curves, or calibration plots, compare predicted probabilities with observed defect frequencies. A well-calibrated model supports more reliable threshold selection and stronger risk-based decision rules because a predicted probability should have a stable real-world meaning. Poor calibration can lead to inconsistent decisions across production periods, especially when thresholds are treated as confidence-based action points. For this reason, calibration assessment is increasingly relevant in manufacturing studies that aim to support deployment rather than offline comparison only (Bauer et al., 2026).

Another important issue is evaluation across time. Many studies still report results based on random train-test splits, but this can produce optimistic estimates when manufacturing data has a temporal structure. In real production, operating conditions change over time, and a model may later face conditions that were not represented during early training. Temporal validation is therefore more informative when the aim is to assess production readiness. Rolling performance tracking, for example, through rolling F1-score or similar windowed metrics, can reveal gradual degradation that would not be visible in one overall score. Phase-based evaluation provides another useful perspective by dividing the production run into periods such as baseline, disturbance, and recovery. This allows performance to be analysed under different operating regimes rather than averaged across the whole dataset. Such analysis can reveal differences in predictive performance across operating conditions. Kvaktun, Liu and Schiffers (2023) investigated this problem experimentally in injection moulding by introducing controlled material-related disturbances into initially stable production and evaluating the resulting prediction errors. Their findings demonstrated that predictive performance can deteriorate following changes in process conditions, supporting the need to evaluate model reliability across distinct operating periods.

Operational performance also depends on latency. In real-time manufacturing systems, a prediction that arrives too late may have little practical value even if its statistical quality is high. Latency can be measured as the time between signal availability and prediction output, and it is often summarised using statistics such as the mean and the 95th percentile. The mean reflects typical behaviour, while the 95th percentile provides insight into higher-delay cases that may matter

during heavy system load or communication instability. This distinction is important because average latency alone may hide long-tail delays caused by buffering problems, temporary disconnection, or increased computation demand. Reporting both typical and high-percentile latency gives a more realistic picture of operational readiness (Yin, 2025).

Manufacturing evaluation is also shaped by cost-sensitive decision-making. The most suitable model or decision threshold depends not only on statistical performance, but also on the relative consequence of false negatives and false positives. In classification problems, different errors may carry different penalties, so these two error types should not always be treated as equally important. Elkan (2001) describes this through the use of a cost matrix, where prediction outcomes are assigned different decision costs; these costs do not need to be strictly monetary and may also represent wasted time, operational burden, or the severity of an incorrect decision. In manufacturing quality prediction, this distinction is important because a false positive may lead to unnecessary rejection, inspection, or rework, while a false negative may allow a defective part to continue downstream. Sankhye and Hu (2020) similarly show that early quality prediction can reduce problems associated with rework, recalls, transportation, and production-plan disruption. Cost-sensitive evaluation helps explain why the model with the highest F1-score the best operational choice is not always. A model with a slightly lower F1-score may still be preferable if it produces fewer of the more harmful error types under the selected decision policy, supporting a closer alignment between prediction policy and manufacturing priorities (Gerling et al., 2022).

The literature also suggests that reporting should support traceability rather than only headline metrics. Confusion matrices, ROC and PR curves, calibration plots, latency summaries, and cost-based comparisons each capture different aspects of production readiness. When these are presented together, they provide stronger evidence of how a model behaves under realistic operating conditions. This is especially important in governed manufacturing systems, where decisions may need to be justified not only technically but also operationally and economically (Alkhatib et al., 2025).

Overall, a reliable evaluation for manufacturing quality prediction should combine class-based metrics, threshold-sensitive analysis, calibration assessment, time-aware performance tracking, latency reporting, and cost-sensitive interpretation. Taken together, these forms of evaluation provide a

stronger basis for judging whether a model is suitable for deployment rather than only for offline experimentation. They also prepare the ground for the next issue in the literature, how model reliability changes when the process itself drifts over time.

2.9 Drift, concept shift, and model reliability in production

Manufacturing processes rarely remain perfectly stable over long production periods. Even when nominal machine settings are held constant, gradual or sudden changes may still occur because of tool wear, machine ageing, sensor degradation, environmental variation, maintenance actions, and changes in raw material batches. These changes can alter the statistical properties of sensor signals and may also change the relationship between process variables and quality outcomes. In machine learning terms, this behaviour is commonly described as drift, and it is one of the main reasons why models that perform well in offline testing may lose reliability during real production operation (Loufakis et al., 2024).

Drift can take several forms. Sensor drift refers to changes in sensor output that do not reflect true process change, such as calibration shift, thermal drift, signal offset, or loss of sensitivity. Process drift refers to genuine change in the underlying behaviour of the process, for example, changes in heat transfer caused by fouling, changes in pressure response caused by valve wear, or changes in cycle timing caused by controller behaviour. Label drift may also occur when inspection practice changes, when operator judgement shifts, or when the definition of a defect changes because of customer requirements. In real industrial use, these forms of drift may occur together, which makes it unsafe to assume that a training dataset will remain representative of future production for long periods (Kvaktun, Liu and Schiffers, 2023; Jourdan, 2024).

Concept drift is more specific than a change in the distribution of measured process variables. It refers to a change in the relationship between the model inputs and the target outcome, so that a previously learned mapping between process behaviour and product quality may no longer remain valid. This distinction is important because a change in sensor or process-variable distributions does not necessarily mean that concept drift has occurred. Conversely, model performance may change even when the input distributions

appear relatively stable. Changes in feature distributions do not necessarily imply loss of predictive validity. Hinder, Vaquet and Hammer (2024) distinguish virtual drift, involving changes in the input-feature distribution, from real drift, involving changes in the conditional relationship between inputs and target outcomes. Neither type of drift necessarily implies a particular change in predictive error. In manufacturing, Bilen et al. (2025) investigated the detection and adaptation of drift in acoustic-signal-based quality prediction for micro-gear production, demonstrating the practical importance of maintaining model reliability under changing process conditions.

In this thesis, the implemented runtime monitoring is therefore described as drift awareness rather than direct concept-drift detection. Changes in selected process features are monitored relative to a reference condition, while later quality evidence and model performance provide the additional information required to determine whether an observed change has practical consequences for prediction. This distinction supports the governed approach adopted later in the thesis, where detected change alone does not automatically authorise model retraining.

The literature reports several drift detection approaches. One common direction is statistical monitoring of selected features or indicators using control-chart logic, z-score thresholds, or moving-window comparison between a reference baseline and recent data. These methods are attractive because they are relatively simple to compute and can operate continuously. Another direction uses distance measures between distributions, including divergence-based methods and two-sample tests, to determine whether recent data differs significantly from baseline behaviour. More advanced work uses embedding-based monitoring, where high-dimensional data is compressed into a lower-dimensional representation, and drift is detected in that latent space. In manufacturing, method choice is usually shaped by three practical needs, interpretability, computational efficiency, and robustness to noise and missing samples (Jourdan et al., 2023; Catti, Nikolakis and Alexopoulos., 2025).

A central challenge is that drift monitoring must distinguish meaningful change from normal cycle-to-cycle variation. Even in stable production, manufacturing signals are never perfectly identical. If drift thresholds are too sensitive, the system may generate repeated false warnings that reduce operator trust and create unnecessary intervention. If thresholds are too relaxed, early degradation may be missed. Accordingly, drift-monitoring methods must balance sensitivity

to genuine distributional change against the risk of false alarms, including the influence of sample size and monitoring-window design (Hinder, Vaquet and Hammer, 2024). In the present study, minimum evidence requirements and staged monitoring statuses, such as insufficient data, candidate drift, and confirmed drift, are adopted as implementation-level safeguards. These distinguish preliminary indications of change from stronger monitoring evidence without treating drift detection alone as authorisation for model retraining.

Drift detection is also closely related to model reliability over time. A model may appear strong when evaluated on data collected under one operating regime, yet perform poorly when the regime changes. This is especially relevant in cyclic manufacturing, where signal shape, timing, and transition behaviour may shift gradually across long runs or abruptly after intervention. Under such conditions, one overall score from an offline dataset may provide a misleading impression of stability. A more informative evaluation requires monitoring how performance changes across time, across phases of operation, and across different disturbance conditions. In this sense, drift is not only a data problem. It is also an evaluation problem because it determines whether reported performance remains meaningful once production moves away from the original training conditions (Zhu et al., 2024).

A common misunderstanding is that detected drift should automatically trigger retraining. In practice, this can be risky. If drift is caused by a temporary disturbance, sensor fault, unstable startup behaviour, or an abnormal transition period, retraining during that interval may cause the model to learn misleading patterns and reduce performance after the system returns to normal behaviour. For this reason, drift monitoring is often more useful as a trigger for investigation than as an immediate retraining command. When updating is allowed, drift detection can trigger the collection and labelling of a small batch of new samples before retraining, while the existing model remains active until the updated model is ready for deployment (Shen et al., 2025). This makes drift monitoring a necessary part of reliable deployment, but not a sufficient rule for model change by itself.

From the perspective of this thesis, the literature points to an important limitation in current manufacturing monitoring research. Many studies recognise that drift exists, but fewer show how drift is handled within a traceable operational framework that links detection, evaluation, decision logic, and controlled updating over long production periods. This gap matters because

model reliability in production depends not only on detecting that something has changed, but also on deciding what the system should do next and how that response should be justified (Jourdan, 2024; Catti et al., 2025).

Overall, drift and concept shift are central challenges in manufacturing monitoring systems because they weaken the assumption that a trained model will remain valid over time. Drift detection methods can provide useful early warning and support safer decision-making, but their practical value depends on threshold design, windowing strategy, staged status logic, and a clear link between detected change and controlled response. These issues lead directly to the next section, which examines how model governance, controlled updating, and human-in-the-loop operation can be used to maintain reliability and trust during long-term deployment.

2.10 Model governance, controlled updating, and human-in-the-loop operation

A major limitation in manufacturing machine learning is that many published studies still treat deployment as if it were the final step after model training. In practice, production use introduces a different problem. Once a model is placed in operation, its behaviour must be managed over time under changing conditions, imperfect labels, and variable process regimes. A model that performs well during offline testing may later become unreliable even if the original training procedure was technically sound. Deployment-focused manufacturing research therefore treats machine learning as a lifecycle-management problem as well as a prediction problem, requiring coordinated practices for model deployment, monitoring, governance, and iteration (Safdar et al., 2024).

One important governance issue is model updating. Retraining is often presented as a natural response when performance weakens, but the literature shows that updating can improve performance in some situations while making it worse in others. This risk appears when retraining is carried out on data collected during unstable operating conditions, when the number of corrected labels is too small to guide learning reliably, or when temporary disturbances are learned as if they were normal process behaviour. Retraining should depend on sufficient new evidence rather than simply on the availability of new data. In a manufacturing MLOps framework, Rungtalay and Kaitwanidvilai (2025) activate supervised model updating only after enough labelled failure samples have been collected,

while significant detected drift can trigger additional retraining. A practical deployment pattern is champion–challenger evaluation, supported by model versioning, continuous monitoring, documented approval processes, and rollback mechanisms, providing a controlled basis for comparing and updating deployed models (Chew et al., 2026).

This structure is especially important in manufacturing because process behaviour often changes across operating phases rather than in one simple direction. Stable production, disturbed operation, maintenance recovery, and material transitions may all appear within the same long run. If model updates are applied without a preserved baseline, it becomes difficult to determine whether performance changes reflect genuine process drift or uncontrolled model modification. Maintaining a stable production model while candidate models are evaluated provides a controlled basis for comparison and reduces the risk associated with model updates during deployment (Chew et al., 2026).

A further requirement is an explicit rule for combining outputs when multiple predictive models are available. Ferraro et al. (2026) demonstrate that manufacturing quality-control ensembles can use a defined late-fusion strategy to select and aggregate model predictions rather than combining outputs indiscriminately. Human-in-the-loop input is often proposed as a way to improve label quality and support safer updating. In manufacturing quality prediction, labels may be delayed, uncertain, or partly subjective. Some defects are not captured fully by simple dimensional or weight measurements, while visual inspection may vary across operators or shifts. Operators review and correction can therefore provide higher-confidence labels, especially for uncertain or high-impact cycles. However, this input is limited by workload and availability, which means that human correction cannot be assumed to exist at full scale for every cycle. Bauer et al. (2025) demonstrated that confidence-based selection of predictions for manual verification can reduce annotation effort while maintaining manufacturing quality-monitoring performance under distribution shift. Their findings support selective human review when model predictions are potentially unreliable, rather than requiring manual verification of every production sample.

Within a governed framework, human corrections become valuable not simply because they improve the dataset, but because they provide stronger evidence for later model improvement. Human-in-the-loop manufacturing systems can incorporate expert feedback into defect-recognition workflows and use that

feedback to support cognitive retraining and subsequent model refinement (Kotsiopoulos et al., 2024). Even then, corrected labels alone are not sufficient. Updating should still depend on conditions such as minimum evidence quantity, acceptable operating stability, and a clear record of what data was used and under what regime it was collected. This is important because a small number of corrected labels gathered during unstable operation may still be insufficient to support safe retraining. The governance problem is therefore not only how to collect better labels, but how to decide when those labels are trustworthy enough to influence model change.

Traceability is a further requirement that appears repeatedly in deployment-focused literature. In industrial settings, it is rarely enough to report that a model was retrained or that a decision was made. It must also be possible to explain what triggered the update, what evidence was available, what policy was applied, and what outcome followed. This is especially important where machine learning outputs influence quality decisions, maintenance actions, or later model replacement. Without traceability, performance changes are difficult to audit, and stakeholder trust becomes difficult to sustain. A governed system requires persistent records of model versions, decision outputs, and model updates so that manufacturing quality decisions can be reproduced, verified, and audited (Mateo-Casalí, Boza and Fraile, 2026a).

Taken together, the literature suggests that manufacturing machine learning should not be treated as a single static classifier deployed once and left unchanged. Practical systems require a stable reference, a controlled path for candidate improvement, explicit decision policies, selective use of human correction, and traceable evidence of how the model behaved and changed over time. These requirements are particularly important in cyclic manufacturing, where process behaviour evolves across long runs and where confidence in production decisions must be maintained under real operating conditions. Lin and Chen (2026) demonstrated a policy-governed manufacturing framework integrating sensor, MES, and ERP information with human review, traceable evidence, and auditable control decisions. However, their framework addresses closed-loop manufacturing control, whereas the present thesis focuses on governed cycle-level quality monitoring and decision support without autonomous adjustment of machine actuators. Governance is therefore treated as part of the real-time monitoring and decision-support architecture itself. It defines how predictions are resolved, how updates are permitted or blocked, how

human corrections are incorporated, and how evidence is preserved for later evaluation. Section 2.11 consolidates these unresolved integration challenges into the research gaps and research questions addressed by this thesis.

2.11 Research gaps, research questions and chapter synthesis

The reviewed literature shows clear progress in several areas relevant to modern manufacturing, including cycle-level sensing, in-mould monitoring, industrial data acquisition, connected communication architectures, machine learning for quality prediction, deployment-oriented evaluation, drift awareness, and human-supported model adaptation. These developments provide a strong technical foundation for real-time quality monitoring and decision support. However, the literature also shows that these capabilities are often developed as separate technical functions rather than as parts of a coherent operational manufacturing architecture. Based on the literature reviewed in Sections 2.1–2.10, four principal research gaps are identified.

Research Gap 1: Limited cycle-cenred integration of process evidence and quality prediction

Cyclic manufacturing naturally generates repeated production units in which process signals, timing information, predictions, quality outcomes, and later corrections can all be associated with the same physical cycle. However, the literature frequently treats acquisition, signal alignment, buffering, feature extraction, storage, and prediction as separate technical tasks rather than as parts of one cycle-centred analytical structure. This can become problematic when cycle segmentation is unstable, signals are misaligned, transient information is lost, or predictions cannot be linked reliably to the corresponding quality evidence. Accurate aggregation and synchronisation of production data are essential because asynchronous sensor streams can reduce data quality and limit the performance and applicability of data-driven manufacturing models (Schmetz and Kampker, 2025).

The literature also suggests that cyclic manufacturing requires a more explicit treatment of the production cycle as the primary unit of analysis. Many smart-manufacturing and connected-monitoring approaches operate at a broader

machine or system level, whereas quality in cyclic processes may depend strongly on how pressure, temperature, position, timing, flow, or other signals evolve within individual production cycles. A cycle-centred architecture is therefore required to preserve the relationship between measured process behaviour and the quality outcome associated with that same production instance (Cerquitelli et al., 2021; Gim and Turng, 2023). A gap remains in connecting cycle-level sensing, physically meaningful process representation, prediction, and quality evidence within one persistent analytical structure.

Research Gap 2: Limited transition from offline predictive performance to runtime manufacturing evaluation

A substantial proportion of manufacturing machine-learning research demonstrates classification or regression performance using historical or experimentally prepared datasets. These studies provide important evidence that process data can be used for quality prediction, but they give less attention to how predictive behaviour changes when models are incorporated into sustained runtime operation. Runtime use introduces additional requirements including cycle integrity, execution timing, signal variability, changing class distributions, delayed or corrected labels, disagreement between analytical routes, and changing operating conditions.

Consequently, predictive accuracy established during model development does not by itself demonstrate that the analytical system will remain reliable once deployed. Zhu et al. (2024) show that changing data distributions and concept drift in industrial manufacturing can limit the practical deployment of trained models and lead to performance degradation over time. A gap therefore remains between demonstrating that a model can predict quality offline and evaluating how predictive models, representations, and decision logic behave as part of an operational cycle-level manufacturing workflow.

Research Gap 3: Limited integration of predictive models with explicit manufacturing decision governance

The reviewed literature contains many examples of sensor systems, predictive models, monitoring platforms, dashboards, and digital manufacturing frameworks. However, manufacturing machine-learning applications remain highly fragmented, with most systems addressing limited and specialised

objectives rather than integrated operational functions (Ördek, Borgianni and Coatanea, 2024). Recent Quality 4.0 research has demonstrated end-to-end decision-support architectures that combine sensor-network data, historical data, and machine-learning models for in-process quality control (Tanane et al., 2025). However, the broader literature still provides limited treatment of how prediction outputs should be incorporated into an explicit governed quality-decision structure.

This issue becomes particularly important when more than one analytical representation or predictive model is available. Model output and manufacturing decision authority are not necessarily equivalent. A predictive model generates an analytical estimate, whereas the released quality decision may also need to reflect application-specific consequences, such as the different implications of accepting a defective part or unnecessarily rejecting a good part. Limited attention has therefore been given to architectures that maintain distinct predictive roles while using an explicit and stable policy to determine the final quality decision. A further gap concerns the traceability of this process, including the ability to identify which cycle, representation, model version, prediction, and policy state produced a particular operational decision.

Research Gap 4: Limited governance of drift-aware model adaptation and human-supported updating

The literature increasingly recognises that manufacturing data distributions and predictive relationships may change during operation. Drift monitoring, concept-shift detection, retraining, and human-in-the-loop correction have therefore become important research topics. However, these mechanisms are often considered independently. In particular, detecting statistical change does not necessarily establish that a deployed model should immediately be replaced, while the availability of new data does not guarantee that those data are sufficiently reliable, representative, or balanced for retraining.

Human correction can provide important supervisory evidence, and recent manufacturing research has begun to integrate targeted human verification with adaptive model updating. Bauer et al. (2025) select potentially unreliable predictions for manual review and use the verified or corrected samples for continual model adaptation under distribution shift. However, a broader gap remains in linking human evidence, data sufficiency, class representation, regime

stability, candidate-model validation, and update authority within one traceable governance mechanism. This creates a need to distinguish between evidence that conditions may have changed and authority to modify an operational predictive model.

Taken together, these four gaps show that the principal unresolved issue is not simply the development of another accurate quality-prediction model. The wider research needs concerns how cycle-level process evidence, complementary analytical representations, predictive models, decision authority, drift awareness, human-supported evidence, controlled adaptation, and persistent traceability can be organised within one governed manufacturing architecture. The thesis addresses this need using injection moulding as the experimental case study while formulating the underlying logic for cyclic manufacturing more generally. Table 1 summarises the relationship between the four literature gaps identified above, the corresponding thesis response, and the chapters in which the supporting evidence is presented.

Table 1. Relationship between identified research gaps, thesis contributions, and supporting evidence.

Research Gap	Thesis Contribution	Supporting Evidence
Gap 1: Limited cycle-centred integration of process evidence and quality prediction	Development of a cycle-centred approach that links process sensing, cycle representation, statistical and predictive analysis, quality evidence, and persistent cycle identity within the same analytical workflow.	Chapters 3–8
Gap 2: Limited transition from offline predictive performance to runtime manufacturing evaluation	Development and offline evaluation of complementary cycle-level predictive representations, followed by their integration and evaluation within the governed runtime architecture.	Chapters 5–8
Gap 3: Limited integration of predictive models with explicit manufacturing decision governance	Development of distinct Champion and Challenger predictive roles together with a separate, locked Final decision policy and traceable cycle-level decision records.	Chapters 6–8
Gap 4: Limited governance of drift-aware model adaptation and human-supported updating	Development of drift-aware monitoring and a governed Challenger adaptation mechanism in which human-supported evidence, data adequacy, class representation, regime suitability, and model-update conditions are considered before adaptation.	

Based on the four literature gaps identified above, the thesis addresses the following research questions:

RQ1. How can synchronised multi-sensor process evidence be organised at the cycle level and linked to part-quality outcomes to support reliable quality monitoring in cyclic manufacturing?

RQ2. To what extent can an event-anchored cycle representation provide quality-prediction performance comparable with a full-curve representation while requiring substantially less raw process information?

RQ3. How can complementary predictive models, explicit decision authority, drift awareness, human-supported evidence, controlled model adaptation, and traceable records be integrated within a governed real-time manufacturing architecture?

RQ4. How does the proposed architecture perform under runtime operating conditions in terms of predictive behaviour, decision-policy performance, timing and robustness, drift awareness, and controlled model adaptation?

In summary, this chapter has shown that reliable quality monitoring in cyclic manufacturing depends on more than sensing or predictive accuracy alone. The literature establishes the importance of cycle-resolved process representation, reliable data acquisition and connectivity, appropriate predictive modelling, deployment-oriented evaluation, drift awareness, and controlled model management. However, these capabilities are often investigated separately, leaving unresolved questions concerning their integration, runtime governance, traceability, and controlled adaptation. The four research gaps and corresponding research questions identified in this chapter therefore provide the basis for the research objectives and subsequent methodology of the thesis. The following chapters move from experimental data acquisition and cycle-level process analysis to statistical investigation, predictive modelling, governed runtime implementation, and operational evaluation.

Chapter 3
Experimental Setup and Data Collection

3.1 Manufacturing trials: cell configuration and case-study process

This chapter presents the experimental setup and data collection programme used to generate the cycle-level dataset for this research. It defines the physical case-study environment, including the injection moulding machine, mould, product, material system, sensor arrangement, and data acquisition pathway. The chapter provides the experimental foundation for the statistical analysis, machine-learning development, and real-time evaluation presented in later chapters.

Injection moulding was selected as the case-study process because it is a strongly cyclic manufacturing operation in which thermal, mechanical, and flow-related variations can produce measurable changes in cycle behaviour and part quality. This makes it suitable for collecting synchronised multi-sensor data under both stable and disturbed operating conditions. The chapter focuses on the experimental case-study environment, while the full governed real-time architecture and operational logic are developed later in Chapter 7.

The experimental programme was conducted through three injection moulding trials using the instrumented Battenfeld 60T production cell. During each trial, parts were produced under baseline operating conditions together with planned changes to selected process settings to provide a wider range of cycle behaviours and quality outcomes. Machine-side and in-mould signals were recorded throughout each cycle using the ComoNeo data-acquisition platform, including hydraulic pressure, screw position, nozzle temperature, cavity pressure, cavity temperature, and mould thermal measurements. Each completed cycle was assigned a cycle identity and linked to the corresponding manufactured part. The produced parts were subsequently inspected using dimensional, weight, thickness, and visual quality checks and classified as good or defective. Across the three trials, 903 complete labelled cycles were retained for the statistical and machine-learning analyses. The detailed sensor installations, connections, and acquisition hardware are provided in Appendix A.

3.1.1 Experimental workflow and system overview

The experimental workflow was organised as a connected sequence from physical production to cycle-level process records and corresponding quality outcomes. Parts were produced using the Battenfeld 60T injection moulding cell

under an initial reference operating condition and subsequent planned variations in selected process settings. The experimental cell comprised the injection moulding machine, mould and tooling, installed process sensors, signal-conditioning and connection hardware, the Kistler ComoNeo data-acquisition platform, and an external computer environment for data processing and monitoring. Each manufactured part was associated with its production cycle so that the measured process behaviour could later be linked to the corresponding inspection result.

Process behaviour was observed from three complementary parts of the manufacturing system. Machine-side measurements, including hydraulic pressure, screw position, and nozzle temperature, represented actuation, material delivery, and the thermal condition before mould entry. In-mould cavity pressure and cavity temperature measured the process response during part formation, while heating-water temperature and flow represented the thermal boundary conditions of the mould. These signals were routed through the required connection and conditioning hardware to the ComoNeo platform, where they were acquired as synchronised time-series data over each production cycle.

Two data routes were used from the same acquisition environment. For offline analysis, cycle data were exported as structured CSV records containing timestamped process measurements and cycle identifiers. For runtime access, ComoNeo was connected to the external computing environment through a local Ethernet/TCP/IP network, with OPC UA providing structured access to selected process variables and cycle-status information. ComoNeo acted as the data source, while the Python-based environment acted as the receiving client for data handling and subsequent monitoring functions. Python tools including Pandas, Plotly, and Dash were used during data organisation, inspection, visualisation, and development of the later monitoring environment.

The physical quality assessment formed the final part of the experimental chain. Produced parts were inspected using dimensional, weight, thickness, and visual checks and were subsequently classified as acceptable or defective. The cycle identifier therefore provided the link between the manufactured part, its multi-sensor process record, and its quality outcome. This workflow established the traceable experimental dataset required for the later statistical analysis, machine-learning development, and governed runtime evaluation. The individual machine and tooling elements are described in the following subsections, the sensing and acquisition arrangements are detailed in Sections 3.2 and 3.3, and the

corresponding hardware installation and connection evidence is provided in Appendix A.

3.1.2 Production cell and machine configuration (case study)

The experimental trials were conducted on a Battenfeld 60T injection moulding machine operating as a conventional production unit. The machine was selected because it represents a typical shop-floor asset that can be instrumented and connected to monitoring tools without major modification. This was important for the deployment focus of the research, since industrial monitoring systems often need to operate with existing machines, controller limitations, and restricted access to internal process signals.

The case-study product was a medical oximeter casing component. This part was selected because its geometry and functional features make it sensitive to thermal and pressure-related process variation. It also supports practical quality inspection, since defects such as incomplete filling, surface irregularities, and geometric inconsistencies can be identified after moulding. The production cell was therefore suitable for generating repeatable cycles while still allowing controlled variability to be introduced during the experimental trials.

Figure 4 shows the injection moulding case-study cell and the relationship between the machine, mould, sensor arrangement, data acquisition unit, and process-data flow.

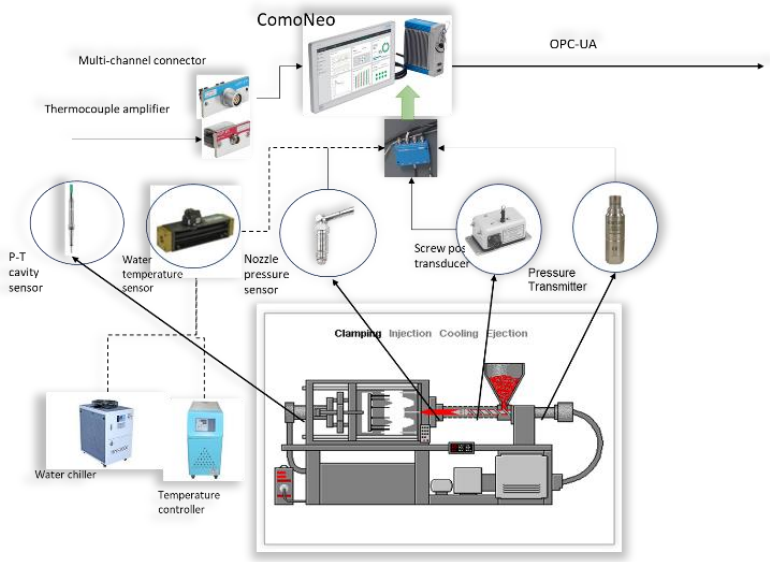


Figure 4. Injection moulding case-study cell showing the machine, mould, sensor arrangement, data acquisition unit, and process-data flow.

3.1.3 Tooling and material context

The mould system used in the trials included interchangeable inserts to support controlled variation while keeping the overall machine and tooling context consistent. Inserts were produced in P20 tool steel and also using 3D-printed DED stainless steel, which allowed rapid iteration and experimental flexibility. This approach supported two important needs. First, the same monitoring method could be tested across different tooling inserts without changing the overall data acquisition architecture. Second, the resulting cycles naturally contained a wider range of behaviours and outcomes, which is valuable when evaluating monitoring robustness rather than only optimised steady-state operation.

The polymer processed for the case-study parts was ABS (Magnum 3453), used with antimicrobial masterbatch additives. This material choice supported stable moulding while still exhibiting quality sensitivity to thermal and pressure changes, which is important when studying defect formation linked to process conditions. Mould and cooling-channel design were supported using Moldex3D, and this ensured that the tooling layout and cooling logic were documented and repeatable. Although the manufacturing method studied is injection moulding, the key point for the general framework is that the material, tooling, and cycle phases were documented clearly enough to interpret how variations in signals relate to variations in outcomes.

Table 2 summarises the main tooling, material, machine, and monitored process context used in the injection moulding case study.

Table 2. Tooling, material, machine, and monitored-process context for the injection moulding case study.

Category	Specification
Machine type	Conventional plastic injection moulding machine
Machine capacity	60-ton injection moulding machine
Machine manufacturer context	Battenfeld-based production environment
Machine controller context	UNILOG 2040 controller boards were reported for most machines in the facility
Monitored machine/process variables	Screw position, injection time, machine pressure, nozzle pressure, runner pressure, cavity pressure, and cavity temperature
Hydraulic pressure sensing	Kistler 4262A hydraulic pressure sensor

Category	Specification
Hydraulic pressure sensor range	Up to 350 bar
Nozzle sensing	Pressure–temperature sensor installed at the nozzle
Nozzle sensor range	Up to 3000 bar and operating temperature up to 350 °C
Screw-position sensing	Kistler P50 position sensor
Processed polymer	ABS (Magnum 3453)
Polymer additives	Antimicrobial masterbatch additives
Material selection rationale	The material was selected to support stable moulding while remaining sensitive to thermal and pressure variation in the process.

3.1.4 Baseline operating window and planned variability

A baseline operating window was established based on datasheets and preliminary trials. This window provided a stable starting point for collecting reference cycles before introducing controlled variations. In industrial settings, a baseline is normally defined as a period of operation where the process is believed to be stable and where acceptable parts are consistently produced. This baseline behaviour is important later because it provides a reference distribution for monitoring, comparison, and drift tracking.

The baseline settings were maintained within normal industrial ranges. Later phases deliberately applied deviations to generate both acceptable and defective outcomes for supervised evaluation. The planned settings and disturbances were selected to generate typical quality-relevant changes such as incomplete filling, increased warpage risk, flash tendency, and general stability loss. The trial design did not aim to produce only clean cycles. Instead, it was structured to include selected disturbances and non-ideal behaviour relevant to the case-study process. These conditions reproduce particular classes of thermal, pressure, filling, and sensing variation that may occur in practice, but they are not intended to statistically represent the full range, frequency, or distribution of variation encountered across industrial injection-moulding production. In addition, This design also allowed the monitoring framework to be tested against both gradual changes and sudden faults, rather than only against a single disturbance type.

Table 3 presents the baseline machine settings used as the reference operating window before controlled process variation was introduced.

Table 3. Baseline injection moulding settings used for reference operation.

Parameter	Value
Injection Pressure	75–85 MPa
Holding Pressure	55–65 MPa
Screw Speed	80–120 rpm
Barrel Temperatures	210–230–240 °C (zones 1–3)
Mould Temperature	35–45 °C
Cooling Time	10–14 s
Cycle Time	~25–30 s

The experimental programme was designed as a set of structured perturbation campaigns rather than as a fully factorial or independently replicated design of experiments. Selected process settings were varied in staged sequences around a stable baseline to create controlled departures in process behaviour and to generate a range of corresponding quality outcomes for monitoring and predictive evaluation. The purpose was therefore to evaluate cycle-level sensing, representation, prediction, and system behaviour under stable and disturbed conditions, rather than to estimate isolated causal effects or formal interaction effects for every process parameter. Because all factors were not independently randomised and replicated across all combinations, the results are interpreted in terms of the observed process responses and quality outcomes rather than as a complete factorial characterisation of injection-moulding parameter effects.

The detailed perturbation sequences, cycle ranges, parameter changes, observed quality outcomes, and recovery stages for the three trials are reported in Sections 4.3.1 to 4.3.3.

3.1.5 Link to the monitoring framework

The outcome of this section is a clearly defined cyclic production context that supports the monitoring framework described in later sections. Cycles are produced continuously in repeated phases and can be segmented consistently

using cycle state signals. Key thermal and mechanical signals can be acquired across the full cycle and aligned in time, allowing cycle-level records to be created for analysis and for real-time decisions. Baseline behaviour and disturbed behaviour can also be compared in a controlled way, which supports evaluation of model stability, drift monitoring, and later governance actions.

The next section details the sensor system design and installation. The focus is placed on the measured variables required to represent thermal and mechanical behaviour at the cycle level and to support later snapshot construction, feature extraction, and model evaluation.

3.2 Sensor system design and installation

A customised in-mould and machine-mounted sensor system was installed on the Battenfeld 60T injection moulding machine to capture the part formation during each cycle. The instrumentation was designed to observe the process from three complementary perspectives: machine-side actuation, in-cavity forming behaviour, and mould thermal boundary conditions. This structure was necessary because quality variation in injection moulding is rarely caused by a single variable, but by interactions among pressure, temperature, motion, timing, flow resistance, and heat transfer.

The selected signals included hydraulic pressure, screw position, nozzle temperature, cavity pressure, cavity temperature, and mould heating-water temperature/flow. Hydraulic pressure and screw position represented the machine-side delivery of energy and material into the mould. Nozzle temperature represented the melt thermal condition before mould entry. Cavity pressure and cavity temperature represented the actual forming environment inside the mould, where filling, packing, cooling, and solidification occur. Mould heating-water measurements represented the thermal boundary condition that influences cooling stability. Together, these signals supported cycle-level analysis, MP1 event anchoring, full-curve representation, and later real-time monitoring.

Cavity pressure and cavity temperature were treated as core process-response signals because they directly describe the in-mould conditions experienced by the polymer during filling, packing, and cooling. The cavity pressure peak was also used later as a physically meaningful anchor point for event-based feature extraction. The broader sensor set was therefore selected not only to monitor

stable operation, but also to capture deviations during non-ideal cycles and planned disturbances. Figure 5 illustrates the pressure and temperature operating boundaries considered when selecting and applying the sensor system for the case-study trials.

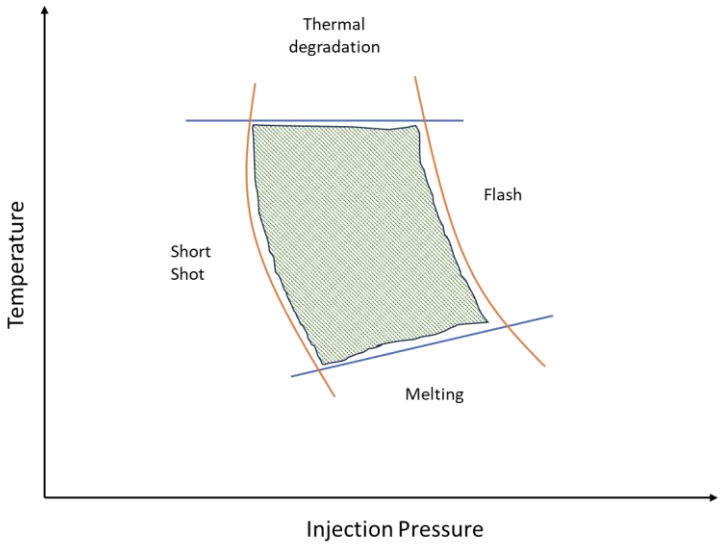


Figure 5. Pressure and temperature operating boundaries are considered for the sensor system during the case-study trials.

Table 4 summarises the monitored signals and their role in the experimental data collection framework. Detailed sensor specifications, installation photographs, mounting locations, and connection arrangements are provided in Appendix A.

Table 4. Sensor channels and measurement purpose in the experimental data collection framework.

Signal/measurement	Sensor or source	Measurement role in the thesis
Hydraulic pressure	Kistler 4262A hydraulic pressure sensor	Machine-side actuation and pressure delivery during injection and packing.
Screw position	Kistler P50 linear displacement sensor	Screw motion, shot delivery behaviour, and cycle progression.
Nozzle temperature	Nozzle-mounted temperature sensing arrangement	Melt thermal condition before mould entry.
Cavity pressure	Kistler 6188A cavity pressure–temperature sensor	In-mould pressure response, filling, packing, and MP1 event anchoring.
Cavity temperature	Kistler 6188A cavity pressure–temperature sensor	In-mould thermal response during filling, packing, and cooling.
Heating water temperature and flow	Smart flow Tracer VM sensor	Mould thermal boundary condition and cooling stability.

3.3 Data Acquisition and Signal Capture Framework

The data acquisition arrangement was designed to capture, synchronise, and store process signals from the experimental injection moulding setup in a form suitable for later analysis, modelling, and live monitoring. The focus in this section is the acquisition pathway from the installed sensors to structured cycle-level data. The wider real-time monitoring software, prediction flow, dashboard, and governed runtime behaviour are presented later in Chapter 7.

A Kistler ComoNeo data acquisition platform was used to record cycle-synchronised process signals during the trials. The platform received analogue signals from the machine-mounted and in-mould sensors, including hydraulic pressure, screw position, nozzle temperature, cavity pressure, cavity temperature, and mould thermal boundary measurements. These signals were recorded as full-time series curves across each injection cycle so that transient behaviour during filling, packing, and cooling was preserved for later full-curve analysis and event-anchored feature extraction.

The acquisition system supported both offline and online data routes. For offline dataset construction, cycle data were exported as CSV files containing timestamped channel readings and cycle identifiers. This allowed each produced part to be linked to its raw cycle record, extracted features, and quality label. For online operation, selected process variables were exposed through OPC UA and accessed by the Python-based monitoring environment. This enabled cycle-by-cycle ingestion without manual file transfer and provided the communication pathway later used by the governed real-time implementation. Signal integrity was supported through stable channel mapping, industrial cabling, signal conditioning, and repeatable connection arrangements. These measures were important because electrical noise, unstable grounding, incorrect mapping, or conditioning drift could create artificial variation that would later appear as modelling error. The detailed connectivity layout, conditioning hardware, channel mapping, and installation photographs are provided in Appendix A.

The OPC UA connection was used only for data access and monitoring rather than for machine control. The DAQ platform provided selected process variables and cycle-status information through OPC UA, while the external Python environment acted as the client for monitoring, buffering, and later decision support. No control commands were sent from the monitoring environment to modify machine operation. Before live use, the exposed nodes and update behaviour

were checked using an independent diagnostic client to confirm that the expected values were visible and updating correctly. A simulated signal source was also used during software development when machine access was unavailable; however, the analytical, predictive, and deployment results reported in this thesis are based on physical machine data.

Overall, the acquisition framework provided a consistent bridge between the physical injection moulding cell and the later analytical and governed real-time stages. It preserved full-cycle signal behaviour for offline modelling while also supporting structured live access for real-time monitoring and governed decision support.

3.4 Computational and Software Environment

A Windows-based computational environment supported the experimental, analytical, and real-time implementation stages. The verified implementation workstation featured an 11th-generation Intel Core i7-11370H processor running at 3.30 GHz, with four physical cores, eight logical processors, and about 16 GB of system memory. The available graphics hardware was Intel Iris Xe Graphics. The deployed Champion and Challenger models were configured for CPU-based execution, and no dedicated GPU acceleration was required for the governed real-time inference stage.

Python provided the principal computational environment for data preparation, feature processing, machine-learning development, evaluation and real-time implementation. NumPy and Pandas were used for numerical and structured data handling, while SciPy supported signal-processing operations, including resampling of the full-curve representation. Scikit-learn was used for conventional machine-learning development, cross-validation and performance evaluation, while XGBoost provided the boosted-tree framework used by the deployed Champion and Challenger models. TensorFlow/Keras was additionally used for the comparative neural-network models evaluated during the offline machine-learning stage. Model artefacts used by the implemented system were serialised and loaded using Joblib.

Statistical analysis was supported by Minitab together with the Python-based analytical environment. Minitab was used for the regression-based statistical analysis reported in Chapter 5, while Python supported subsequent data

processing, classification evaluation, visualisation and machine-learning analysis. Matplotlib and Plotly were used for analytical and presentation graphics, while Dash provided the interactive monitoring and visualisation interface used in the implemented real-time system.

Communication between the data-acquisition environment and the Python-based monitoring system was implemented using OPC UA over the local Ethernet/TCP/IP network. The Python implementation used the asyncua library for OPC UA communication, including the live reader and the simulation environment used during development and verification. The Kistler ComoNeo platform provided the acquisition-side process data, while the Python environment performed cycle handling, feature construction, inference, monitoring and evidence generation.

Persistent runtime information was stored using SQLite, including cycle records, predictions, monitoring events and retraining-related evidence. Completed full-cycle signal records were retained in compressed NumPy .npz format, while CSV and JSON files were used for training datasets, evaluation outputs, model metadata, configuration records and exported evidence. Trained model artefacts were retained in Joblib format. This combination provided a consistent data path from experimental acquisition through offline analysis and model development to governed cycle-level real-time monitoring and decision support.

Saved runtime evidence records Python 3.13.5 together with NumPy 2.2.6, Pandas 2.3.1, SciPy 1.16.0, scikit-learn 1.7.1, XGBoost 3.0.4 and Matplotlib 3.10.3 for the validated implementation environment. The implemented software stack also used Dash, Plotly, Joblib and asyncua for the user interface, model persistence and industrial communication functions.

3.5 Experimental Data Handling and Record Preparation

The signals acquired from the experimental setup were not used directly as independent sample points. Instead, they had to be reorganised into cycle-level records that preserved the relationship between the physical production cycle, the measured process behaviour, and the inspected quality outcome. This preparation stage was therefore an essential link between the experimental acquisition system described in the previous sections and the statistical,

machine-learning, and live-evaluation stages developed later in the thesis. Its purpose was to ensure that each accepted cycle could be treated as a coherent analytical unit rather than as a loose collection of sensor readings.

To achieve this, the exported multi-sensor signals were handled in a Python-based environment that supported file organisation, signal inspection, preliminary visualisation, and structured record preparation. The use of tools such as Pandas, Plotly, and Dash made it possible to inspect cycle behaviour, check signal consistency, and confirm that the measured variables retained the same meaning across the experimental trials. At this stage, the role of the software environment was mainly preparatory: it provided the practical means to organise the experimental records and verify their suitability for later modelling, rather than serving as the final real-time implementation itself.

Once the signals had been imported and checked, the main task was to preserve their cycle-related structure. Each production cycle needed to retain its timing information, channel identity, and relationship to the corresponding moulded part. This was important because the thesis does not evaluate quality prediction at the level of isolated sensor samples; it evaluates whether the behaviour of a complete manufacturing cycle can be represented, interpreted, and linked to a binary quality outcome. For this reason, the data preparation process focused on forming comparable cycle records that could support both feature extraction and later truth-labelled evaluation.

Within these cycle records, the cavity pressure signal, denoted as MP1, was used as the main physical reference for event anchoring. This choice was made because cavity pressure describes the in-mould response during filling and packing and therefore provides a meaningful event around which the state of the process can be synchronised. Peak detection was applied automatically using the operational method detect_mp_peak(mp, t). Where this detection could not be completed reliably, a fallback based on argmax(mp) was used, together with validity checks and index clamping to avoid invalid signal positions. In this way, the feature-extraction process remained tied to a physical event while still retaining robustness under imperfect or disturbed signal conditions.

After the MP1 peak index had been identified, values from the other monitored variables were extracted at the same cycle position. This produced an event-anchored snapshot in which hydraulic pressure, screw position, nozzle temperature, cavity temperature, and mould heating-water temperature were

aligned to the same physical moment in the cycle. The advantage of this approach was that the compact feature vector did not depend only on absolute time, which may vary between cycles. Instead, it represented the process state at a repeatable event associated with material filling, pressure development, and part formation. This made the extracted features more comparable across both stable and disturbed cycles. Figure 6 shows the experimental data-preparation dashboard used to inspect cycle curves, model outputs, and label-related fields during the preparation and checking of cycle-level records.

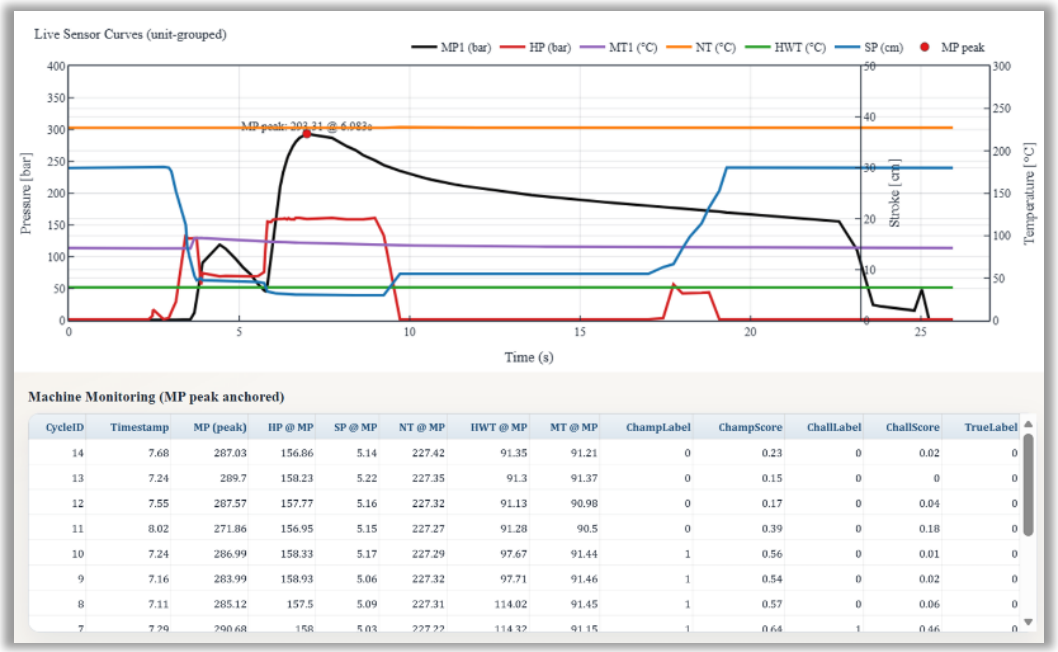


Figure 6. Experimental data-preparation dashboard showing cycle curves, model outputs, and label-related fields.

At the same time, the preparation workflow also preserved the possibility of using a fuller representation of the cycle. Rather than reducing all information to the MP1 peak snapshot, complete time-series records were retained so that full-curve modelling could later be performed. This allowed the thesis to compare two different ways of representing the same production evidence. The event-anchored route provided a compact, physically meaningful snapshot suitable for efficient live inference, while the full-curve route preserved broader temporal behaviour across the cycle and supported a richer reference model. Because both routes originated from the same accepted cycle records, their later comparison was based on a consistent experimental foundation. Table 5 summarises the two cycle representations prepared from the experimental records and shows how each representation was used in the later modelling and live-evaluation stages.

Table 5. Cycle representation summary comparing the Champion full-curve route and Challenger event-anchored route.

Representation	Input basis	Transformation/construction	Main content retained	Model path	Operational role	Main strength	Main limitation	Governed updating
Full-curve representation	Accepted complete cycle snapshot	Time-series channels clipped to safe ranges, resampled to a consistent length, and normalised using stored scaler statistics.	Broader temporal structure of the cycle, including wider in-cycle process behaviour across the measured channels	Champion	Stable reference analytical route during live operation	Richer structural description and higher interpretive stability	Higher computational cost and heavier runtime handling	No — fixed during live operation
Event-anchored snapshot representation	Accepted complete cycle snapshot	Compact feature vector generated around the selected MP1 cavity-pressure peak, using aligned process-state values and engineered interactions.	Local event-centred process state at the anchor, including physically meaningful peak-aligned features	Challenger	Lightweight operational route for live inference and controlled adaptation	Faster execution, simpler feature form, easier governed updating	Stronger dependence on robust cycle segmentation and stable anchor behaviour	Yes — only path eligible for governed retraining

The same record structure also supported traceability. By preserving the link between raw signals, cycle identifiers, extracted features, quality labels, and later model outputs, the dataset could be followed from physical production through to prediction and evaluation. This was necessary because the later stages of the thesis rely not only on producing predictions, but also on showing which cycle produced each prediction, which representation was used, and how the prediction related to the available quality evidence. The detailed runtime mechanisms that extend this structure into database storage, dashboard operation, drift monitoring, event logging, and governed retraining are therefore built on the data-handling foundation established in this section.

Overall, this stage converted raw multi-sensor outputs into traceable cycle-level records suitable for the later analytical and operational stages of the thesis. MP1 peak anchoring provided a repeatable physical event for compact feature extraction, while full-curve preparation preserved the wider temporal behaviour of each cycle. Together, these prepared records formed the basis for the statistical analysis, machine-learning modelling, and governed live evaluation that follow in the subsequent chapters.

3.6 Handling outliers, extreme conditions, and data integrity

During the experimental trials, extreme values and abnormal cycles were encountered as a result of intentional process variation, machine behaviour, sensor limitations, and occasional data-handling faults. These events were not treated uniformly as noise. Instead, they were separated according to whether they represented physically meaningful process behaviour or corrupted records that did not describe the process correctly. This distinction was important because the aim of the research was not to build a model only for clean steady-state data, but to evaluate whether cycle-level monitoring could remain credible under the non-ideal conditions that occur in practical manufacturing.

Some unusual cycles arose from planned parameter changes and from unplanned machine interruptions encountered during the trial programme. These cycles produced altered curve shapes, shifted timing, or different peak behaviour. When such changes were linked to real operating variation, they were retained as valid extreme cycles because they described genuine process behaviour near the edge of the operating envelope. Other unusual records were caused by data faults,

including duplicated values, partial cycle segments, constant-zero sections, misaligned timestamps, or interrupted export. These records were not considered meaningful process extremes because they did not reliably represent the physical cycle.

Sensor-related issues were also considered during data preparation. For example, screw-position measurement required consistent zeroing so that the same physical movement was represented consistently across cycles. The hot-water temperature sensor, which was external to the main Kistler sensor set, required separate calibration before stable values were obtained. Early inaccurate readings were therefore checked and corrected once the calibration procedure had been stabilised. These examples show why data integrity could not be treated as a simple automatic filtering task; interpretation was needed to distinguish true process variation from measurement or recording artefacts.

A further source of complexity was that some part defects were observed even when the monitored variables appeared to remain within expected ranges. Surface defects such as silver streaks and line defects were later associated with uneven cooling behaviour caused by limitations in the cooling-channel design. This observation was important because it showed that not all quality-relevant causes are fully captured by the monitored signal set. It also reinforced the need for cautious interpretation of model confidence, especially when boundary conditions or unmeasured physical effects influence the final part quality.

For downstream modelling and evaluation, unusual data were therefore divided into two categories. Valid extremes were cycles that were unusual but physically meaningful, such as cycles produced under planned parameter variation, boundary operation, or genuine process disturbance. These cycles were retained because they represent realistic operating conditions and allow model behaviour to be tested where robustness is most needed. Invalid records, by contrast, were cycles that did not represent the process correctly, such as records with corrupted timing, duplicated segments, incomplete buffers, or sensor dropouts. These records were excluded or marked as unusable because they could introduce false patterns and bias the evaluation.

A structured validation stage was applied before feature extraction and model training. Raw cycle files were inspected for invalid patterns, including constant-zero sections, repeated blocks, incomplete cycles, discontinuities, and isolated electrical noise spikes. Where single-sample artefacts occurred, spike handling or

smoothing was applied to prevent them from dominating extracted features. By contrast, cycles generated under planned disturbances were preserved. This approach ensured that the dataset retained meaningful process variability while excluding records that were unreliable as physical evidence. Table 6 summarises the main data issues encountered during the trials and the handling decisions applied before feature extraction and model training.

Table 6. Data issues encountered and handling decisions

Data condition	Detection rule	Action	Justification
Constant-zero sections	Check if the signal remains unchanged for a defined window	Remove cycle	Indicates sensor or recording failure
Discontinuities	Detect sudden large differences between consecutive samples	Smooth or flag	Prevent noise from dominating features
Repeated blocks	Detect identical repeating segments in time series	Remove cycle	Indicates duplication or logging error
Incomplete cycles	Check for missing start or end of cycle	Skip cycle	Cannot represent full process behaviour
Electrical noise spikes	Detect isolated extreme values outside the local trend	Apply spike filtering	Avoid single-point distortion
Planned disturbances	Identified from known process conditions or experiment design	Retain	Represents valid operating behaviour

Parts were labelled as either acceptable or defective using dimensional checks, thickness and weight validation, and visual inspection. Visible defects such as holes, short shots, surface anomalies, and other non-conformities were used together with the physical checks to assign the binary quality label. This labelling strategy provided the ground truth used in the later statistical and machine-learning chapters, while still allowing disturbed but physically meaningful cycles to remain in the dataset.

The same data-integrity philosophy was also required for the online pipeline. Valid extreme cycles had to remain available for evaluation, while invalid records had to be detected and handled safely. In the deployed pipeline, this was supported through cycle-boundary checks, buffer validation, peak-anchoring safeguards, and event logging. The purpose was to prevent partial or corrupted cycles from entering the stored dataset silently, while still allowing physically meaningful deviations to be retained as part of the operating envelope.

Figure 7 illustrates the difference between a disturbed but valid cycle and a corrupted or invalid record. This distinction is central to the methodology because it explains why some unusual data are retained for robustness evaluation, while other records are excluded to protect dataset integrity.

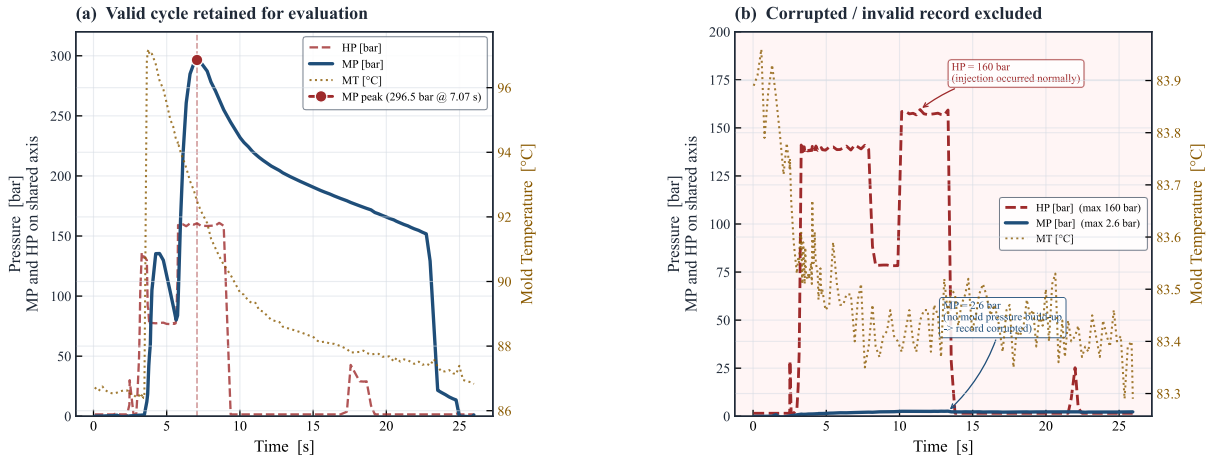


Figure 7. Disturbed but valid cycle versus corrupted/invalid record

To support reliable live operation, a concise set of data-governance rules was applied. These rules control cycle boundary detection, buffer completeness, peak anchoring, signal validity, model input preparation, final decision traceability, run separation, human correction, drift monitoring, and event logging. They were not treated as optional cleaning steps, but as part of the operational methodology required to maintain trustworthy cycle-level records under realistic production conditions.

Table 7. Live governance rules and recorded outcomes

Rule area	Trigger/condition	Action taken	Logged fields/traceability
Cycle boundary governance	Debounced rising edge of CycleDone; fallback to time-reset when CycleDone is missing or unstable; boundary too close to previous boundary	Accept boundary, use fallback boundary, or ignore duplicate / bounce cycle	Boundary source, duplicate/bounce reason, event log entry
Buffer completeness and alignment checks	Missing required channel, incompatible sample lengths, invalid time vector, abrupt buffer ending, repeated blocks, sudden discontinuity	Skip the cycle and do not store the record	Skip reason, buffer integrity event, cycle exclusion log

Rule area	Trigger/condition	Action taken	Logged fields/traceability
Peak anchoring safety rules	detect_mp_peak(mp, t) fails, NaN in MP or time, mismatched lengths, invalid peak index, unusable MP signal	Use argmax(mp) fallback where valid; clamp index if needed; mark cycle invalid and exclude if MP unusable	Peak method used, fallback flag, invalid-cycle reason
Signal validity rules for online use	Frozen signal, flat zero dropout, stopped time vector, and other corruption patterns.	Treat as invalid data and exclude; retain physically meaningful but unusual cycles.	Corruption type, exclusion reason, retained-extreme context if applicable
Model input governance	Champion full-curve inputs are unavailable, or challenger snapshot features cannot be produced.	Log failure; continue safely without forcing corrupted prediction	Model input failure, affected route, exception/warning log
Final decision policy and traceability	Champion/challenger predictions available and final rule applied	Store both model outputs, the final label, and the final decision reason	Champion prediction, challenger prediction, final label, *final_reason*
Run separation and database integrity.	New run started, duplicate cycle detected, DB path/table counts checked, export attempted with too few cycles	Separate by run ID, block duplicate insert, confirm DB state, gate weak export	*run_id,* duplicate protection event, DB path, table counts, export gate status
Human correction and retraining gates	Newly reviewed corrections available; minimum counts, class counts, correction-rate cap, training sufficiency, class-balance checks	Allow retraining assessment only if gates pass; otherwise, block retraining.	Correction counts, class counts, correction rate, gate outcome
Drift and retrain governance	*zscore_max* indicates possible drift; insufficient evidence; cooldown after retraining, insufficient rows	Trigger only after sufficient evidence, enforce cooldown, and log non-evaluable state explicitly	Drift metric, drift status, insufficient-row state, cooldown / retrain event

Taken together, these decisions provided a controlled basis for using imperfect experimental data without weakening the credibility of the later modelling results. Valid physical extremes were retained because they represent realistic shop-floor behaviour, while invalid records caused by sensor faults, export faults, or buffering errors were excluded or skipped with recorded reasons. This strengthened dataset integrity, reduced the risk of biased model evaluation, and prepared the foundation for governed cycle-level monitoring.

3.7 Governed real-time system methodology and architecture

The preceding sections defined the experimental cell, sensor arrangement, data acquisition route, and cycle-level data preparation process. These elements form the physical and data foundation for the governed real-time architecture developed in this thesis. The architecture is designed to connect cycle measurements with digital representation, predictive analysis, decision support, monitoring, evidence storage, and controlled model management within a traceable operational structure.

The system is concerned with quality monitoring and decision support rather than autonomous process control. Process measurements are captured and organised into traceable cycle records, evaluated using predictive models, monitored for changing behaviour, and used to support governed operational decisions and controlled model adaptation. The architecture therefore establishes a continuous operational workflow between sensing, cycle segmentation, predictive interpretation, decision governance, evidence storage, human review, and controlled updating, while no automatic feedback signal is sent to modify machine actuators or production setpoints.

This scope reflects the experimental implementation developed in this research while providing an architectural basis that could support future integration with higher-level manufacturing systems or separately governed machine-control functions.

3.7.1 System scope and operational definition

The system developed in this thesis is treated as a governed real-time digital representation and decision-support architecture. It is not limited to passive visualisation or offline analysis. Instead, it connects the physical manufacturing process, acquired sensor signals, cycle-level digital representations, predictive models, decision logic, monitoring functions, and stored evidence within one governed structure. Each accepted production cycle becomes the basic operational unit through which process behaviour is observed, represented, predicted, evaluated, and recorded.

The proposed system does not claim to provide a full physics-based replica of the injection moulding process, nor does it claim autonomous control of machine

actuators. Its contribution lies in providing a governed digital representation and decision-support structure through which cyclic process behaviour can be monitored, evaluated, and managed in a traceable way. Model adaptation is not treated as unrestricted self-learning. It is permitted only when sufficient evidence, reviewed labels, and predefined retraining conditions are available.

To clarify this functional scope, Table 8 distinguishes the proposed architecture from a conventional monitoring dashboard and a more general monitoring system.

Table 8. Distinction between a monitoring dashboard, a monitoring system, and the governed real-time architecture proposed in this thesis.

Capability	Monitoring Dashboard	Monitoring System	Governed Real-Time Architecture Proposed in This Thesis
Live sensor connectivity	✓	✓	✓
Real-time process visualisation	✓	✓	✓
Cycle-level process representation	✗	Partial	✓
Predictive quality inference	✗	Partial	✓
Explicit decision logic	✗	Partial	✓
Traceable decision records	✗	Partial	✓
Drift awareness and monitoring	✗	Partial	✓
Human-in-the-loop correction	✗	Partial	✓
Controlled model adaptation	✗	✗	✓
Governed evidence management	✗	✗	✓
Run-level traceability	✗	Partial	✓
Governed information and update workflow	✗	Partial	✓

As shown in Table 8, the proposed architecture extends beyond data visualisation and process monitoring alone. It performs cycle-level digital representation, predictive inference, governed decision-making, drift-aware monitoring, evidence generation, and controlled model adaptation. The system is therefore

treated as an integrated governed real-time architecture rather than solely as a monitoring dashboard or standalone machine-learning application.

3.7.2 Methodology-level architecture

The methodology is organised around a layered structure linking the physical process to data acquisition, cycle representation, prediction, governance, monitoring, and evidence storage. At the lowest level, sensor signals are acquired from the production system and organised into cycle-synchronised streams. These signals are then converted into accepted cycle records through boundary detection, buffer validation, and integrity checks.

Once a cycle record is accepted, two complementary digital representation routes can be generated. The first is a compact event-anchored representation based on the MP1 peak. The second is a fuller curve-based representation that preserves broader temporal behaviour within the production cycle.

These representations support the Champion–Challenger modelling structure used later in the thesis. The Champion acts as a stable reference route, while the Challenger provides an updateable operational route that can be modified only under governed conditions. A Final decision layer resolves the model outputs according to a locked policy so that released decisions remain consistent and auditable.

Above the predictive layer, monitoring and adaptation functions track changes in process behaviour, correction evidence, retraining eligibility, and runtime behaviour. Model adaptation remains separate from drift detection itself. A detected change therefore provides monitoring evidence but does not automatically authorise model updating. Any retraining or model-management action is subject to defined governance conditions.

The final evidence layer preserves cycle records, predictions, decision reasons, drift states, retraining outcomes, timing information, and run-level metadata. This allows process evidence, predictive outputs, operational decisions, and subsequent model-management actions to remain connected to the corresponding production-cycle identity.

Table 9 summarises the methodology-level layers, their main inputs, outputs, and stored evidence.

Table 9. Methodology-level system layers, inputs, outputs, and stored evidence.

Layer	Main input	Main output	Stored evidence/traceability
Data stream layer	Sensor signals from DAQ and OPC UA, including MP1, MT1, HP, SP, NT, HWT, *CycleDone* mould open/close	Live cycle-synchronised process stream	Raw signal availability, channel state context
Cycle management layer	Live stream and cycle-state signals	Buffered cycles, accepted boundaries, skipped-cycle events	Boundary source, skip reason, duplicate/bounce events
Representation layer	Accepted cycle buffer	Event-anchored features and full-curve representation	Peak index, feature set, preprocessing context
Model layer	Event-anchored and full-curve inputs	Champion and Challenger probabilities/labels	Model route outputs, model metadata reference
Decision layer	Champion and Challenger predictions	Final label and decision reason	Final label, *final_reason*, policy setting
Monitoring and adaptation layer	Event-anchored features, live predictions, and reviewed corrections	Drift status, retrain decisions, gate outcomes	Drift metric, drift state, retrain event log, blocked/admitted outcome
Evidence and audit layer	Cycle records, predictions, drift snapshots, retrain events, and latency records	Export-ready figures, tables, logs, and metadata	*run_id*, stored outputs, evidence package contents

3.7.3 Operational reliability and performance considerations

Because the proposed framework is intended for real-time operation, reliability and timing were considered as part of the methodology. Runtime performance was monitored through cycle processing time, model inference time, database write time, and dashboard update behaviour. Both average latency and tail latency are important because occasional slow cycles may affect real-time usability even when the mean processing time appears acceptable.

The system also included reliability measures to reduce the risk of silent failure during extended operation. These included OPC UA reconnection logic, watchdog checks for frozen updates, safe skipping of invalid cycles, boundary-integrity checks, and validation that required OPC UA nodes were available and updating as expected. These measures ensured that invalid data or temporary communication faults were handled explicitly rather than being allowed to corrupt stored cycle records or model inputs.

The reliability provisions therefore form part of the architecture rather than being treated only as software implementation details. Reliable cycle identification, valid signal acquisition, controlled handling of invalid records, and preservation of traceable evidence are necessary before predictive outputs can be interpreted as meaningful manufacturing decision support.

3.7.4 Summary

This section defines the governed real-time methodology and system scope used in the thesis. The proposed system is not an autonomous actuator-control system or a full physics-based replica of injection moulding. It is a governed real-time digital representation and decision-support architecture that connects sensing, cycle-level representation, predictive inference, explicit decision governance, monitoring, evidence storage, human oversight, and controlled model adaptation.

The methodology provides the bridge between the experimental setup described in Chapter 3 and the real-time implementation and evaluation presented in the later chapters. It establishes how physical process evidence is transformed into cycle-level digital information and how predictive and model-management actions are organised within a traceable governed architecture.

Chapter 4
Cycle-Level Data Processing and Dual-Representation Feature Engineering

4.1 Introduction

This chapter explains how the raw process signals collected from the experimental injection moulding setup were transformed into structured datasets suitable for analysis and modelling. While Chapter 3 established the physical case-study environment and the data acquisition arrangement, the purpose of Chapter 4 is to describe how those raw multi-sensor signals were organised at the cycle level, prepared for interpretation, and converted into representations suitable for statistical analysis and machine learning.

The central task of this chapter is therefore data structuring rather than data acquisition. Although preprocessing raw process signals is not unique to injection moulding, it is particularly important in this case study because each moulding cycle produces multiple time-series signals whose interpretation depends on the cycle boundary and the corresponding stage of part formation. The signals must therefore be organised by cycle, temporally aligned, and linked to the quality label of the corresponding moulded part before modelling. This cycle-level structuring is required to construct both the event-anchored representation and the full-curve representation developed in this chapter.

Two complementary representations are developed in this chapter. The first is an event-anchored representation, in which process variables are aligned at a critical in-cycle landmark. In the injection moulding case study, this landmark is the peak cavity pressure during the packing stage. The second is a fuller curve-based representation, which preserves more of the original temporal information of the cycle and supports comparison with the more compact event-anchored form. Together, these two views allow the thesis to examine the trade-off between richer signal description and more compact cycle-level feature construction.

In addition to representation, this chapter addresses the preparation of structured datasets for subsequent analysis. This includes the handling of cycle-level records, the linkage between process variables and quality labels, the management of signal consistency across trials, and the preparation of feature tables suitable for later statistical modelling and machine learning. In this way, Chapter 4 forms the bridge between the physical data collection environment presented in Chapter 3 and the analytical and predictive stages developed in Chapters 5 and 6.

The detailed runtime implementation of live cycle handling, online prediction, governed storage, and operator-facing monitoring is presented later in Chapter 7.

In contrast, the focus of the present chapter is on the offline structuring logic that establishes the dataset foundation required for those later stages.

4.2 Feature Compilation Pipeline

Figure 8 summarises the feature compilation pipeline adopted in this research. The pipeline begins with the acquisition of raw multi-sensor signals from the mould and injection moulding machine through the data acquisition system. The collected channels include cavity pressure (MP1), cavity temperature (MT1), hydraulic pressure (HP), screw position (SP), nozzle temperature (NT), and mould heating-water variables where available. Together, these channels capture both the machine-side actuation behaviour and the in-cavity process response needed to represent each production cycle in a consistent and physically meaningful form.

Two complementary data access modes were used during the development of the dataset and the later evaluation stages. In the offline mode, each injection cycle was exported as a structured CSV file containing timestamped multi-channel time-series data. This mode was primarily used for exploratory analysis, feature development, dataset preparation, and offline model training. In the live mode, selected variables and cycle markers were transferred through OPC UA into the Python-based environment developed in this research. Within the context of the present chapter, this live linkage is relevant because it ensured consistency between the datasets used for offline feature compilation and the signals later used in the deployed environment. The detailed runtime handling of live buffering, online segmentation, prediction, and logging is presented later in Chapter 7.

The purpose of the feature compilation stage was therefore to convert these raw signals into structured cycle-level records that could support later statistical analysis and machine learning. This required preservation of signal identity, cycle consistency, and alignment between variables so that each cycle could be linked reliably to a quality outcome and compared fairly across the experimental trials.

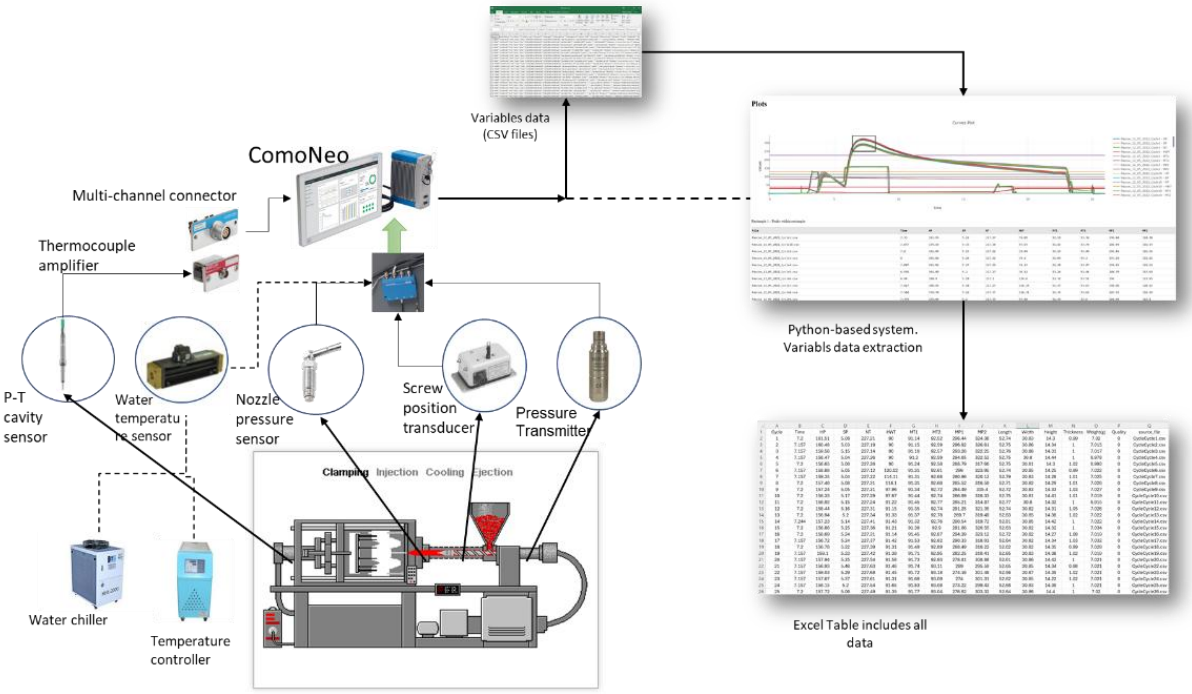


Figure 8. Data flow from sensor acquisition to cycle-level feature compilation.

4.2.1 Cycle segmentation and buffering

To compile features reliably, the continuous signal stream was first segmented into complete cycles. Samples were buffered until a cycle boundary was detected using the primary trigger (CycleDone rising edge with debounce), with a time-reset boundary used as a backup in cases where the primary trigger was unavailable or inconsistent. This design reduced the risk of duplicated cycles and prevented partial cycles from being silently included in the dataset. When an incomplete or corrupted cycle was detected, the cycle could be safely skipped and recorded as a traceable event, preserving dataset integrity.

4.2.2 Event-anchored feature extraction at MP1 peak

After a cycle was finalised, the cavity pressure curve (MP1) was used as the reference signal for defining a consistent event anchor. Peak time and value were identified using the operational peak detector $detect_m p_p eak(mp, t)$, with a safe fallback to $\mathrm{argmax}(MP1)$ when peak detection was not possible, such as NaN handling or length mismatch. In the implemented runtime pipeline, the physical MP1 maximum occurs during the production cycle, but it is identified

computationally only after cycle completion, when the completed buffered MP1 curve is available for evaluation. The event-anchored Challenger prediction is therefore a post-cycle prediction based on the process state at the MP1 event, rather than an early-in-cycle prediction generated when the physical peak occurs. This timing reflects the architecture evaluated in this thesis rather than an inherent limitation of the event-anchored representation. The Challenger requires only the variables aligned to the MP1 anchor and could, in principle, be evaluated shortly after the peak has been robustly confirmed from the live signal. However, the evaluated architecture combines this route with a full-curve Champion that acts as the stable reference and requires the completed cycle record. Because the governed Final decision is resolved after the available Champion and Challenger outputs have been evaluated, producing the Challenger output earlier would not advance the timing of the Final decision in the present configuration. The implementation therefore performs both inference routes after cycle completion to preserve a common cycle boundary and consistent decision point. Consequently, the present architecture provides cycle-level quality monitoring and decision support rather than within-cycle corrective control; the released decision cannot alter the quality outcome of the same part whose cycle has already been completed. No fixed peak threshold or window constraint was imposed beyond validity guards and index clamping to signal length. Once the MP1 peak index was determined, aligned values of other channels (HP, SP, NT, MT1, HWT) were extracted at the same index to create a cycle-level snapshot feature vector. Where engineered interaction terms were used later (e.g., ratios or combined indicators), these were computed from the same aligned cycle-level values to preserve physical consistency.

The event-anchored approach reduces sensitivity to small cycle-time variation because features are extracted at a repeatable physical moment rather than at absolute time. This improves comparability across cycles collected under both stable and disturbed conditions.

4.2.3 Full-curve representation for Champion inference

In addition to snapshot features, a full-curve representation was compiled for model inference in the baseline path. In this representation, each channel is clipped to valid bounds, resampled to a fixed length, normalised using scaler statistics stored in the model metadata, and then passed to the baseline model for

inference. This approach preserves temporal shape information and supports consistent comparison against the event-anchored representation used by the alternative modelling path.

4.2.4 Dataset assembly and governed storage

Compiled outputs were stored cycle-by-cycle as structured records rather than only as ad-hoc spreadsheets. Each cycle record includes identifiers, extracted features, model outputs, and label fields when available (predicted, human-corrected, and truth when synced). A run identifier is used to prevent mixing records across experiments and to support reproducible analysis. Export to CSV/Excel remained available for inspection and reporting, but the governed storage approach supports long-run monitoring, traceability, and later evidence generation (e.g., confusion matrices, ROC/PR plots, and latency summaries).

4.2.5 Interactive visual selection tool

An interactive Python-Dash plotting interface was also used during development to visualise multi-channel curves and validate where critical phases occur in the cycle. In early iterations, rectangular selection overlays were used to inspect regions of interest (e.g., injection/packing phases) and confirm that the peak anchoring logic was selecting the intended physical event. In the operational pipeline, however, cycle segmentation and MP1 peak detection were executed automatically to ensure repeatability and reduce dependence on manual interaction.

4.3 Experimental Procedure and Trial Observations

The primary goal of this stage is to assess the relationship between process parameters and the quality of parts produced by the plastic injection moulding machine. This foundational analysis establishes whether the observed patterns and trends in the process data are sufficient to support the development of machine learning models capable of predicting product quality with reasonable accuracy. The dataset constructed during this study includes multiple parameters compiled at the cycle's MP1 peak anchor point, detected automatically from the

MP1 curve, which is considered the most informative segment of the cycle due to its strong correlation with part integrity.

Each injection cycle is treated as a single data instance. A consistent set of independent variables was extracted for every instance, representing critical aspects of the process. These variables include the time of the cycle, hydraulic pressure (HP), screw position (SP), nozzle temperature (NT), hot water temperature (HWT), mould cavity pressure (MP1), and mould cavity temperature (MT1). These process features were measured at the precise moment the MP1 value reached its peak within the cycle. This specific time point was chosen as it corresponds to the transition between the injection and packing stages, where pressure stabilisation and material compaction affect the resulting product quality.

The dependent variable in the dataset is the binary quality label, indicating whether the part produced in a given cycle was acceptable, labelled as 0, or defective, labelled as 1. The final curated dataset used for statistical analysis, machine-learning development, and evaluation consisted of 903 complete cycle records. Each cycle was linked to the output part produced during that cycle and was manually inspected and evaluated. The inspection protocol comprised dimensional measurements of part length, width and thickness using calibrated instruments, together with weight validation, followed by visual inspection for defects such as sink marks, warpage, short shots, flash and other non-conformities. These assessments informed the overall binary quality classification, while the individual dimensional measurements were not used as prediction targets in this thesis. The resulting dataset contained 488 acceptable cycles and 415 defective cycles. These labels formed the authoritative quality ground truth used throughout the modelling and evaluation stages.

Unlike approaches that remove atypical observations solely to maintain data homogeneity, this research purposefully retained physically meaningful outlier values and extreme observations where the underlying cycle record remained analytically valid. These included cycles affected by known process faults, genuine operational disturbances, machine stops, and parameter extremes. The rationale for this decision was twofold. Firstly, it enables the machine learning model to develop robustness against variations and faults that can occur in practical manufacturing settings. Secondly, it provides a broader view of the process behaviour under both standard and boundary conditions, allowing the system to detect early signs of deviation or potential failure. These records comprised both

deliberately introduced parameter perturbations and naturally occurring operational or sensing anomalies, which were retained where they remained analytically valid.

The experimental campaign was conducted over three separate trials on different days. Across the campaign, selected process parameters were varied in planned stages around baseline operation, while unplanned operational and sensing events were documented separately. The purpose was to create controlled changes in process behaviour and corresponding quality outcomes for subsequent monitoring and predictive evaluation. The planned disturbances were not independently repeated as isolated experimental factors, and their individual quality effects were therefore not treated as statistically established causal effects. Instead, the observed responses were documented descriptively at the trial level and later analysed within the combined cycle-level dataset using the statistical and machine-learning methods presented in Chapters 5 and 6.

4.3.1 First Trial

The first experimental trial consisted of 250 full injection cycles. The primary aim of this phase was to establish a baseline dataset under standard operating conditions and then progressively introduce parameter variations to study their effects on part quality. The first 100 cycles were conducted using stable and calibrated machine settings. These baseline parameters included standard screw position, default packing pressure, nominal nozzle and barrel heater temperatures, and a Heating Water Temperature (HWT) targeted at approximately 91°C. However, during the early part of the trial, the HWT recorded by the DAQ system exceeded expectations, reaching up to 95°C. This discrepancy was closely monitored and later traced to a calibration issue in the HWT sensor, which was subsequently corrected.

Visual inspection of parts revealed that cycles 14 to 19 consistently produced defective components, characterised by small holes in both cavities. This observation was believed to be linked to instability in the cooling water circuit, which resulted in uneven cooling, particularly on one side of the mould. The issue was significant enough to prompt a review of the cooling system's physical setup. It was later discovered that the design of the cooling pipe network was insufficient to maintain a uniform temperature distribution. As a result, a physical

modification was made to the pipe design to enhance thermal regulation and improve process stability.

After the first 100 cycles, a planned packing-pressure perturbation was introduced to examine how changes in the pressure profile affected part quality under otherwise controlled operation. The disturbance was intended to challenge the process away from its baseline condition and generate non-nominal quality behaviour, rather than to reproduce a single pre-specified defect type. Three documented pressure-setting stages were applied: P1 = 23, P2 = 38 and P3 = 95 during Cycles 101–126; P1 = 24, P2 = 24 and P3 = 97 during Cycles 127–134; and P1 = 20, P2 = 30 and P3 = 85 during Cycles 136–151. Physically, reduced pressure support was expected to weaken cavity filling and packing, thereby increasing the likelihood of incomplete or dimensionally poor parts. During these stages, part quality deteriorated and many components exhibited short-shot defects, characterised by visibly incomplete cavity filling. The observed response was therefore consistent with the expected effect of insufficient pressure support, although the disturbance was not independently replicated as an isolated factor and the result is not interpreted as a statistically isolated causal effect. At Cycle 152, the machine settings were restored to their original values, and the remaining cycles were completed without further issues. This final stretch served as a return-to-normal benchmark to assess whether prior defects were reversible and to confirm the effectiveness of re-establishing baseline conditions.

The first trial provided essential insights into the role of packing pressure in achieving dimensional accuracy and complete cavity filling. It also highlighted the system's sensitivity to seemingly minor disturbances in thermal conditions. The data collected during this trial formed a core subset of the overall dataset, containing both nominal and faulty cycles, and served as a valuable training set for early-stage feature analysis and labelling validation.

Figure 9 shows the cycle-by-cycle variation of the key process variables during the first trial, including the initial baseline period, the staged packing-pressure reductions, and the recovery after the machine settings were restored.

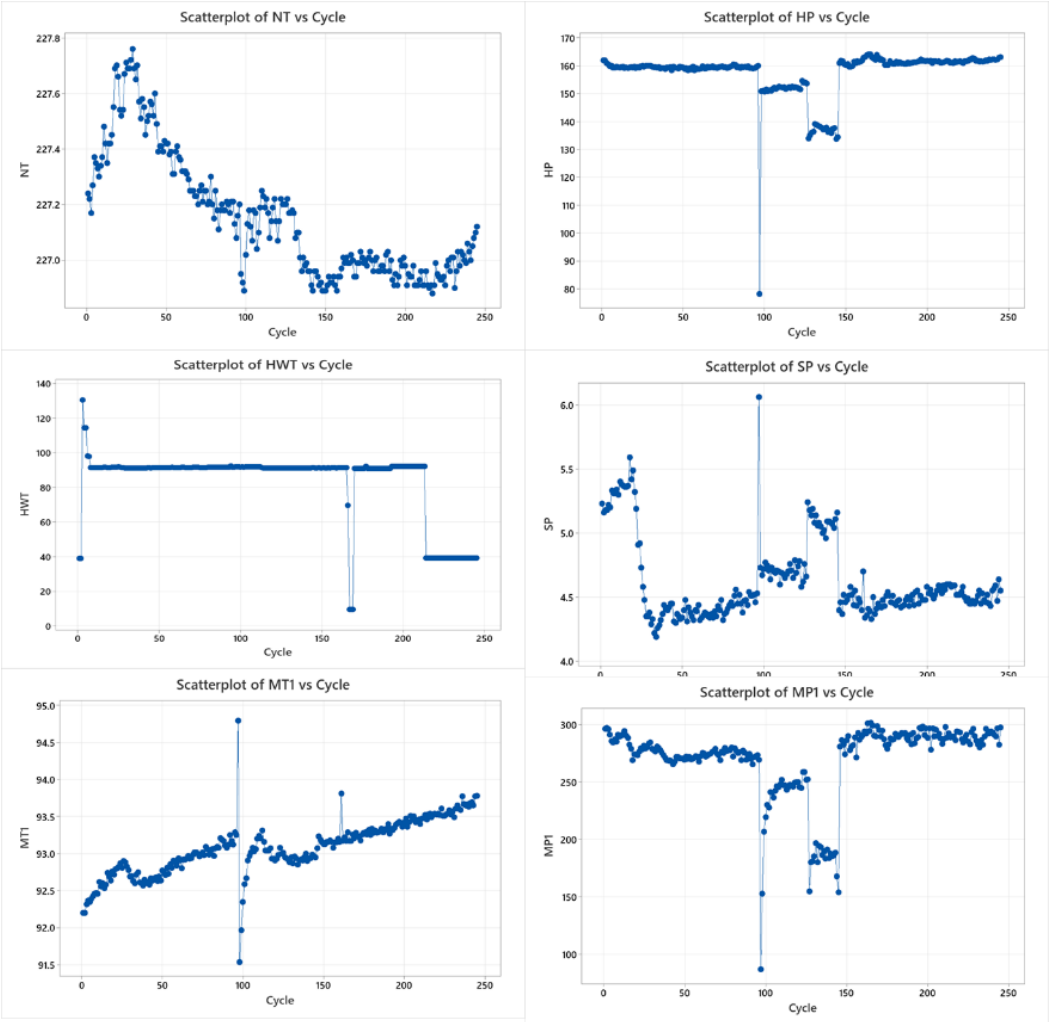


Figure 9. Cycle-by-cycle variation of key process variables during the first trial, showing baseline operation, packing-pressure reduction phases, and recovery after restoration of nominal settings.

4.3.2 Second Trial

The second trial focused on evaluating the effects of thermal and mechanical process parameters on part quality by deliberately altering the Heating Water Temperature (HWT), the nozzle heater band temperature, and the screw position settings. This stage was designed to simulate a variety of operational disturbances commonly encountered in real manufacturing environments, assessing the machine's sensitivity and the system's robustness under suboptimal conditions.

The trial began by modifying the Heating Water Temperature (HWT) supplied to the mould thermal-control circuit. The purpose was to examine how changes in the mould thermal condition affected part quality and surface appearance. The

nominal HWT was 93°C. Starting from Cycle 5, the temperature was reduced in stages, first reaching 85°C and then 80°C by Cycle 53. Visible surface marks began to appear from Cycle 24, and at the lower-temperature condition the parts showed more severe quality degradation, including visible holes in both cavities. The disturbance was therefore continued to examine the process response under a wider thermal deviation from the nominal condition. From Cycle 86 onwards, HWT was restored to 93°C. The visible holes were no longer observed after restoration, although some surface marks persisted temporarily, indicating that the process did not return immediately to its earlier quality condition.

Following the HWT disturbance, a second planned thermal perturbation was applied by reducing the nozzle heater temperature from its nominal value of 230°C, beginning at Cycle 151. The purpose was to examine the process response to a reduced melt thermal condition. A lower nozzle temperature was expected to increase melt viscosity and flow resistance, thereby making complete cavity filling more difficult. Defects were observed from Cycle 153, including visible holes in the moulded parts, and the process continued under the reduced-temperature condition until Cycle 198, when production stopped because of a screw blockage. The exact intermediate nozzle-temperature set-points were not recorded separately in the trial description and are therefore not interpreted as independently controlled treatment levels. The nozzle heater temperature was restored to 230°C by Cycle 202, and production resumed at Cycle 203. Part quality subsequently returned to predominantly acceptable condition, although one short-shot part was observed at Cycle 206.

The final phase of the second trial applied a planned screw-position disturbance to examine whether a reduced injection stroke could still provide sufficient material for complete cavity filling. The nominal screw retraction limit was 23 mm. At Cycle 207, it was reduced to 20 mm and remained at this setting until Cycle 225; at Cycle 226, it was further reduced to 15 mm and remained there until Cycle 237. Reducing the screw retraction limit decreases the available injection stroke and therefore the volume of material available for delivery into the mould. The disturbance was not introduced to target a specific defect type, but to test the filling response under progressively reduced material delivery. As the screw position was reduced, increasingly incomplete parts and repeated short-shot defects were observed. At Cycle 238, the screw position was restored to 23 mm, after which acceptable part quality was re-established. The observed response was consistent with insufficient material delivery at the reduced screw-position

settings, although the disturbance was not independently replicated as an isolated factor.

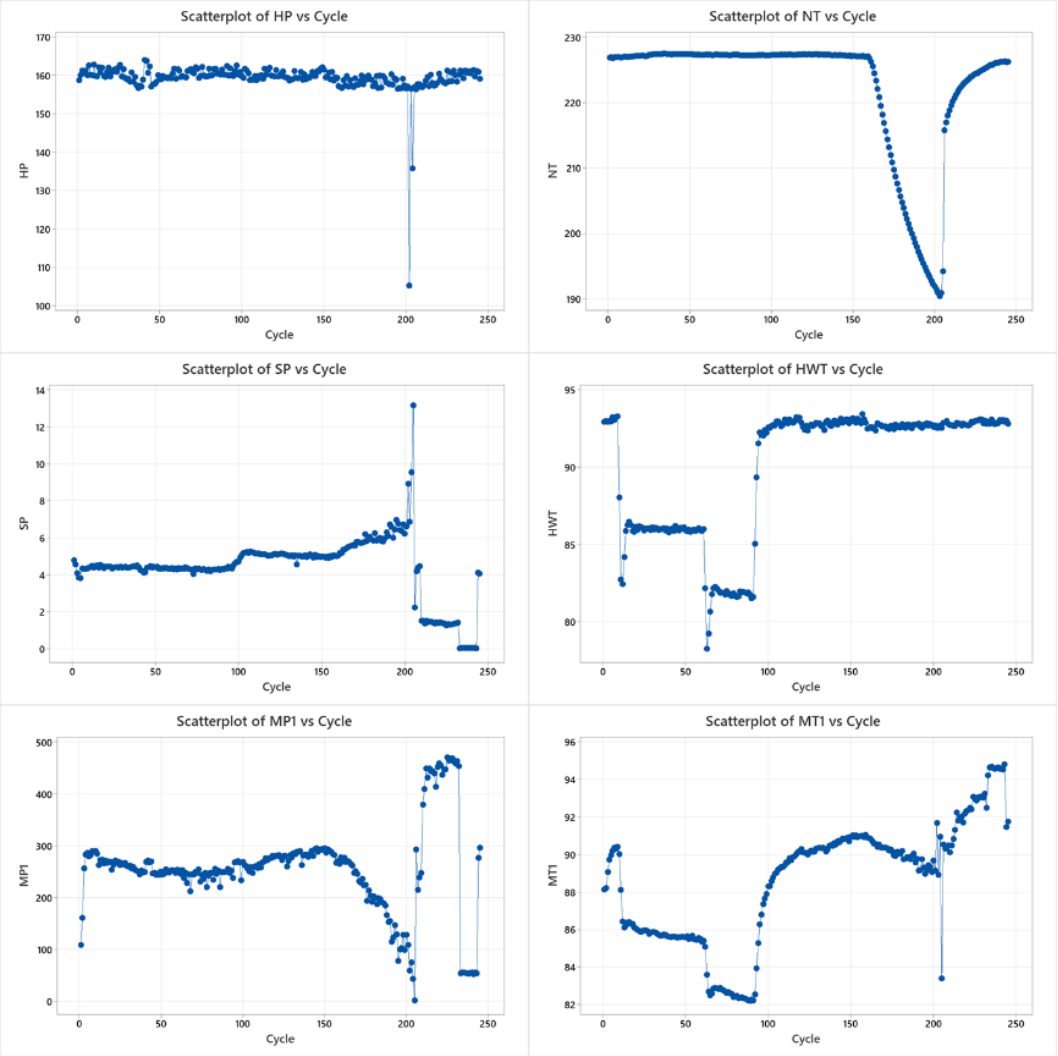


Figure 10. Cycle-by-cycle variation of key process variables during the second trial, showing staged disturbances in heating-water temperature, nozzle temperature, and screw position, followed by restoration to nominal conditions.

Overall, the second trial documented the machine's response to planned thermal and mechanical perturbations involving Heating Water Temperature, nozzle heater temperature, and screw position. The observed quality changes showed that departures from the nominal thermal and material-delivery conditions were associated with both subtle and severe defects within the investigated operating range. These observations were used to broaden the range of cycle behaviours available for subsequent statistical analysis and machine-learning evaluation

rather than to define statistically validated operating limits. Figure 10 shows the cycle-by-cycle variation of the key process variables during the second trial, including the staged disturbances in HWT, nozzle temperature, and screw position, followed by restoration to nominal operating conditions.

4.3.3 Third Trial

The third and final trial aimed to investigate the influence of advanced parameter changes, particularly focusing on variations in flow rate and adjustments in hydraulic pressure, which had not been thoroughly explored in the earlier stages. The first phase, spanning Cycles 1 to 150, involved systematic monitoring of key process parameters, including hydraulic pressure, nozzle temperature, screw position, and core cooling water temperature.

At Cycle 151, the machine stopped unexpectedly following a decrease in heater-band temperature. This event was recorded as an unplanned operational interruption rather than as an experimental disturbance. After production resumed, two planned setting changes were introduced to examine the process response to higher machine-side operating conditions. Injection speed was increased from 37 mm/s to 40 mm/s starting at Cycle 200, and hydraulic pressure was increased from 67 bar to 69 bar at Cycle 269. The injection-speed change was expected to alter the rate of material delivery during filling, while the hydraulic-pressure change was expected to alter the force applied during injection and packing. These changes were intended to observe the process response to the higher settings rather than to produce a specific defect type. In the final stage of the trial, a separate flow-rate measurement assessment was conducted rather than an intentional process disturbance. Mould cooling-water flow was measured over 25 cycles and core cooling-water flow over 15 cycles. The recorded signals showed noticeable oscillation, indicating possible calibration or measurement instability. Nozzle-pressure readings also appeared inconsistent with the expected intensification behaviour, raising additional concerns about sensor accuracy. These observations were therefore treated as data-quality and sensor-verification findings rather than as evidence of a process-induced quality effect, highlighting the need for further calibration and verification before quantitative interpretation of these measurements.

Overall, the third trial documented the process response to planned increases in injection speed and hydraulic pressure, while also capturing an unplanned

machine interruption and separate sensor-quality limitations during the flow-rate assessment. These observations broadened the range of operating and data-quality conditions represented in the experimental dataset without treating the individual changes as statistically isolated causal effects. Figure 11 shows the cycle-by-cycle variation of the key process variables during the third trial, including the machine interruption, the later increases in injection speed and hydraulic pressure, and the unstable sensor behaviour observed during the flow-rate assessment stage.

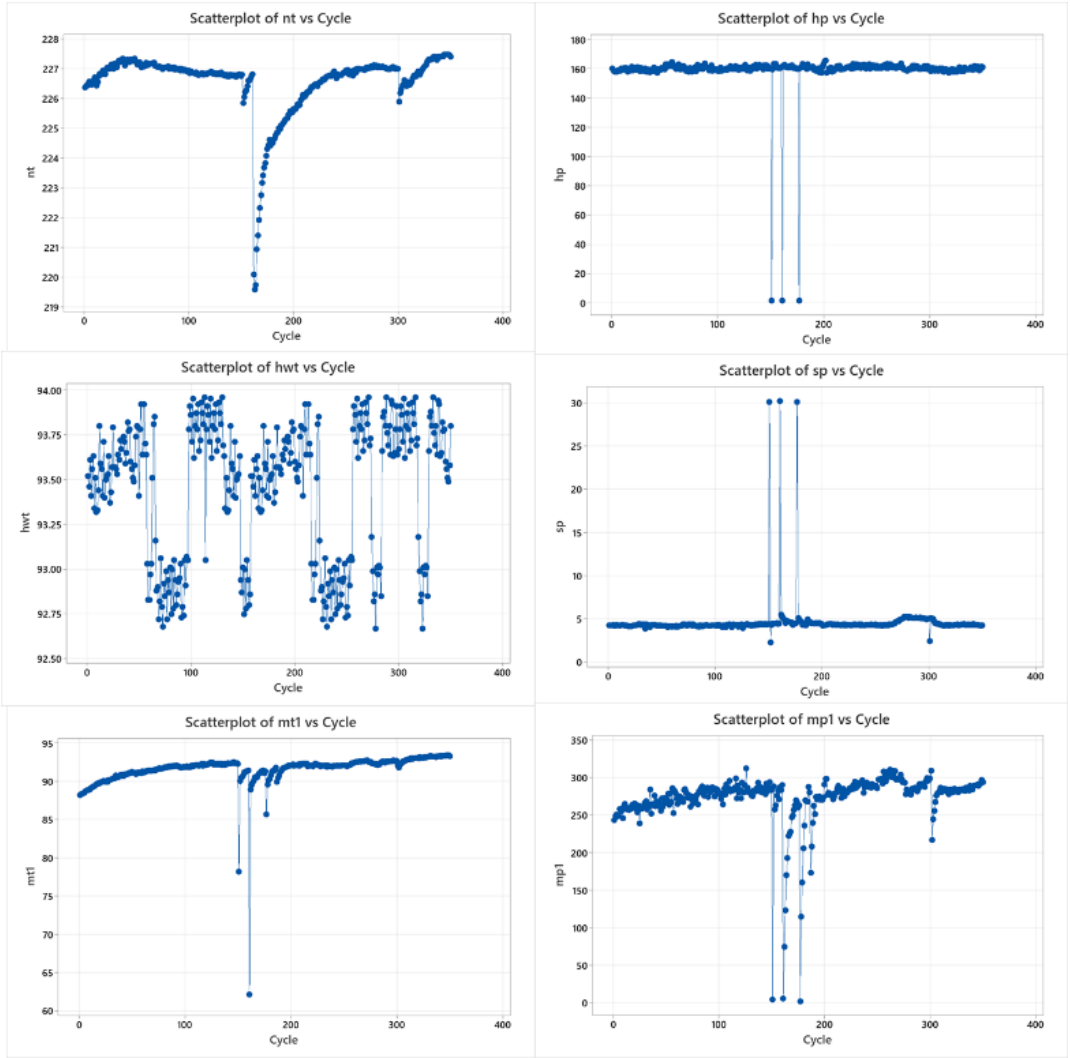


Figure 11. Cycle-by-cycle variation of key process variables during the third trial, highlighting machine interruption, later increases in injection speed and hydraulic pressure, and unstable sensor behaviour during flow-rate assessment.

4.4 Features Selection and Description

After completion of the experimental trials, a structured cycle-level dataset was compiled to capture selected process variables at a common in-cycle reference point: the peak mould cavity pressure during the packing stage, together with the corresponding peak time. The purpose of this representation was to produce a consistent cycle-level snapshot that could support comparison across cycles and provide a compact input for later analysis and modelling. The extracted variables form the basis of the statistical analysis presented in Chapter 5 and contribute to the development of machine learning models presented in Chapter 6. The packing stage was selected as the reference region because it is widely recognised as an important phase in injection moulding, during which compaction, shrinkage compensation, and thermal-mechanical interaction strongly influence the final condition of the part.

In this thesis, the MP1 peak is used primarily as an alignment landmark rather than being assumed a priori to be the single dominant predictor of part quality. Operationally, the MP1 anchor is identified from the cavity pressure time series using an automatic peak-detection routine, with a safe fallback to the raw maximum when needed. This provides a consistent temporal reference for extracting aligned values from the other measured channels, even when individual cycles are disturbed or contain moderate signal irregularity.

The selected features were chosen on the basis of process relevance, sensitivity to cycle-to-cycle variation, and their expected relationship to part quality according to the injection moulding literature. Hydraulic Pressure (HP) represents the machine-side pressure behaviour associated with injection and packing actions. Variations in HP may be associated with differences in filling stability, packing consistency, and general process response. Screw Position (SP) reflects screw displacement and therefore provides information related to shot delivery and dosing behaviour. Variation in SP may indicate changes in material delivery and packing behaviour across cycles.

Nozzle Temperature (NT) was included because it is closely related to melt temperature at the machine outlet and may influence viscosity and flow conditions entering the mould. Heating Water Temperature (HWT), although measured externally to the cavity, was included to represent part of the thermal boundary condition of the moulding process. Changes in HWT may be associated

with differences in cooling balance, thermal recovery, and cycle-to-cycle heat transfer behaviour.

It should be noted that the first 12 cycles of Trial 1 were affected by an HWT calibration error. To preserve analytical integrity, these cycles were excluded from all statistical summaries, correlation analysis, and interpretation involving HWT in this chapter.

On the cavity side, Mould Cavity Pressure (MP1) and Mould Cavity Temperature (MT1) provide direct measurements from within the mould and therefore offer a closer representation of the actual forming environment experienced by the polymer. MP1 reflects the local pressure response during filling and packing, while MT1 provides a local thermal indication linked to the cavity-side process condition. Together, these variables help describe the process's internal state at the selected reference moment.

The time coordinate at which the MP1 peak occurs was also retained as a feature. This variable provides a temporal reference for examining shifts in cycle behaviour across the trials. Differences in peak timing may indicate changes in process dynamics, response delay, or altered progression through the filling-to-packing transition.

The use of a packing-phase anchor provides two advantages. First, it reduces dependence on absolute cycle duration by aligning variables at a physically consistent event within each cycle. Second, it allows multi-sensor values to be compared at a common reference point rather than at arbitrary timestamps. For this reason, the event-anchored representation was adopted as a compact and interpretable cycle-level description for later modelling.

The rationale for selecting these variables is consistent with the defect mechanisms and monitoring principles discussed in Chapter 2. In the later modelling stages of the thesis, these peak-aligned variables are used as the compact event-based feature set, while the fuller time-series channels are retained separately to support comparison with a richer curve-based representation under the same production conditions.

4.5 Data Quality and Integrity

A structured quality assurance (QA) and integrity workflow was applied to ensure that the final cycle-level dataset was appropriate for subsequent statistical

analysis and machine learning development. This QA step did not treat data quality as a one-off offline action; instead, the same integrity logic was aligned with the operational monitoring pipeline so that the dataset and the live system follow consistent rules for what constitutes a valid cycle record.

An initial evaluation was performed to check completeness and consistency across all selected features in the compiled cycle-level table. The final feature dataset used for analysis contained no missing values in the selected fields; however, this does not imply that raw acquisition data always arrived in a perfect form. During trials, interrupted cycles, misaligned exports, calibration periods, and occasional sensor or logging artefacts were observed. Therefore, cycle integrity checks were applied upstream to prevent corrupted or incomplete cycles from being silently included in the analytical dataset. Cycles that failed integrity checks were safely excluded ("safe-skip") and recorded as traceable events rather than being merged into the training set.

A key integrity requirement in this work is the one-to-one correspondence between a physical production cycle and a stored dataset row. In the live pipeline, this correspondence is enforced through explicit cycle boundary logic. The primary boundary trigger is the rising edge of the CycleDone signal, with debounce applied to avoid duplicate boundaries. A time-reset rule is retained as a secondary boundary trigger to recover safely if the primary signal is missing or delayed. Mould open/close states are published and logged as contextual signals, but they are not used as the snapshot trigger. These boundary rules, together with buffering and safe-skip behaviour, ensure that each stored record represents a complete and internally consistent cycle.

Feature scaling was handled in a representation-dependent way. For event/peak-anchored snapshot features, values were retained in physically meaningful units where possible to support interpretability and process understanding. In contrast, the curve-based representation used by the Champion model requires standardisation for inference; therefore, full-curve channels were clipped, resampled, and normalised using stored scaler statistics (from the model metadata) prior to prediction. This separation preserves physical interpretability for cycle-level engineering features while ensuring stable model behaviour for high-dimensional curve inputs.

Outliers and extreme values were intentionally retained when they represented valid process behaviour. Because trials included deliberate parameter variations

and realistic disturbances, extreme readings often correspond to meaningful deviations (e.g., pressure loss, temperature drift, dosing instability) rather than statistical noise. Retaining these extremes supports the development of models that remain reliable under non-ideal shop-floor conditions and improve sensitivity to early-stage faults and rare failure modes. At the same time, invalid anomalies—such as duplicated blocks caused by interrupted logging, discontinuities caused by export faults, or constant zero values due to sensor disconnect—were treated as data integrity failures rather than "useful outliers," and were excluded under the integrity rules described above.

Although the current dataset does not cover the full combinatorial space of injection moulding settings, it provides a scalable and extensible foundation. The governed cycle structure (run separation, integrity checks, and traceable event logging) supports continued growth by allowing additional cycles to be appended safely, while maintaining auditability of what was included, what was skipped, and why. This design supports progressive refinement of the monitoring system towards robust quality assurance under evolving production conditions.

Injection moulding data are commonly represented in the literature either by complete process signals or by features derived from individual signals, such as peak values, gradients, integrals, statistical measures, and characteristic process points. In many such approaches, feature extraction is primarily used to construct a modelling dataset, while the organisation and traceability of the underlying cycle records receive less methodological emphasis. In the present study, a cycle-resolved data structure was established in which each completed production cycle was identified, validated, associated with its corresponding quality outcome, and retained within its experimental run. Feature construction was then performed across the monitored channels using a single physically defined reference event, the maximum cavity pressure, so that the process variables were sampled at the same in-cycle state rather than being extracted independently from unrelated points in each signal. The distinction of the proposed approach therefore lies in integrating traceable cycle-level data organisation with physically synchronised multivariable feature extraction, while preserving the original cycle signals for the alternative full-curve modelling route.

4.6 Summary

This chapter has described the data preparation and feature engineering workflow used to convert raw multi-sensor signals into a cycle-level dataset suitable for downstream statistical analysis and supervised modelling. Sensor signals acquired during the injection moulding trials were processed through a structured pipeline that (i) segments and validates cycles, (ii) extracts physically meaningful features, and (iii) stores results in a consistent format that supports both offline analysis and live monitoring.

Feature compilation focused on event-anchored extraction at the moment of peak cavity pressure during the injection/packing transition. In each cycle, the peak location was determined using the operational peak detection routine detect_mp_peak(mp, t), with a safe fallback to the raw argmax(mp) when needed. At the detected peak index, aligned values of Hydraulic Pressure (HP), Screw Position (SP), Nozzle Temperature (NT), Heating Water Temperature (HWT), Mould Cavity Pressure (MP1), Mould Cavity Temperature (MT1), and the corresponding Time value were extracted and compiled into a single cycle record. This event-anchored approach provides a consistent physical reference point across cycles, reducing sensitivity to small variations in cycle duration while preserving strong links to part formation behaviour.

To ensure the integrity of the cycle-level dataset, segmentation and validation rules were applied upstream of feature compilation. Cycle boundaries were primarily triggered by a debounced rising edge of *CycleDone*, with a time-reset rule retained as a secondary trigger for safe recovery when required. Cycles that were incomplete or corrupted (e.g., interrupted logging, duplicated blocks, or misalignment) were handled using safe-skip logic and recorded as traceable events, ensuring that the final analytical dataset preserves a one-to-one correspondence between stored records and valid physical cycles.

The experimental programme intentionally introduced systematic variations in key process settings, allowing the dataset to capture both stable operation and boundary conditions that induce defects. As a result, extreme but physically meaningful cycles were retained to support robustness and generalisation, while invalid anomalies were excluded through integrity checks. The final compiled feature table used for analysis contains no missing values for the selected fields, reflecting completion after validation rather than an assumption of perfectly clean raw acquisition

The work presented in this chapter directly supports the first research question and first objective by establishing a cycle-level analytical structure in which synchronised multi-sensor process data are organised, validated, and linked to the corresponding part-quality outcome. In doing so, the chapter addresses the first research gap concerning the limited cycle-centred integration of process evidence and quality prediction. The development of the event-anchored and full-curve representations also establishes the methodological basis required for the second research question and second objective, which are evaluated quantitatively in Chapter 6. Chapter 5 first builds on the cycle-level dataset through statistical analysis of process–quality relationships, while Chapter 6 evaluates the predictive performance and representation requirements of the two modelling routes.

Chapter 5
Statistical Analysis of Process Variables and Their Relationship with Part Quality

5.1 Introduction

This study explores the relationship between key process variables and final part quality, focusing specifically on the moment of peak cavity pressure (MP1) during the injection moulding cycle. The maximum cavity-pressure event was selected as a reproducible, physically interpretable reference within the filling–packing pressure evolution. Although this event is associated with melt compression and packing behaviour, its timing is not assumed to coincide with the machine's filling-to-packing switchover. The pressure and temperature measured at this event provide a synchronised representation for investigating their associations with the final part-quality outcome. Using this point as the basis for analysis provides a consistent and repeatable feature extraction across process cycles. The MP1 peak is physically meaningful and practically reliable because it is clearly captured by sensors and it is less affected by noise than earlier or later cycle points.

Using the maximum cavity-pressure event as a common temporal reference enables the relationships between cavity-pressure behaviour and the synchronised hydraulic pressure (HP), screw position (SP), nozzle temperature (NT), heating-water temperature (HWT), and peak time (Time) to be examined across production cycles. In the exploratory linear regression, maximum cavity pressure (MP1) is treated as the response variable, allowing statistical associations with the measured process variables to be quantified without implying causation. Part quality is examined separately through the recorded binary quality labels in the subsequent logistic regression analysis. Thus, the maximum cavity-pressure event provides a physically meaningful reference for interpreting process behaviour, without assuming that the pressure measurement alone establishes the quality of an individual moulded part.

To add context to MP1, the study also includes mould cavity temperature (MT1) recorded at the same peak point. While MT1 is not used as a primary dependent variable in regression modelling, it provides thermal insight into the cavity’s state during compression. Examining pressure and temperature together enhances understanding of the physical condition of the polymer at this critical stage of the moulding cycle. It also aids in detecting failure modes where pressure instability and thermal imbalance could lead to defects.

The regression analysis part of this chapter aimed to determine how much of the variation in MP1 could be explained by the selected machine-side variables. These

variables are directly accessible from the machine interface or sensor outputs, and they represent controllable production parameters. The regression results show that Time, HP, SP, and NT have p-values below 0.05, indicating that they contribute significantly to MP1 variation within the fitted model. The model achieved an R-squared of 43.09% and an adjusted R-squared of 42.82%, which reflects a moderate but meaningful ability to explain cavity pressure behaviour based on process settings. Although this value may seem modest, it aligns with expectations in industrial scenarios where thermal lag, mechanical effects, and sensor noise limit the predictive power of simple linear models.

The dataset used in this chapter includes 903 cycles, and it retains outliers and extreme values, such as cycles affected by machine stops, abnormal thermal conditions, unstable screw movements, and pressure surges. These anomalies were not removed or smoothed because they reflect real production irregularities, and they are essential for understanding real process behaviour. Including these cycles also supports the later goal of developing machine learning models capable of detecting rare or unexpected defect scenarios. Removing them would bias the analysis towards ideal cycles and limit the model's ability to identify issues in real factory conditions. The class distribution and train–test partition of the 903-cycle dataset are summarised in Table 10.

Table 10. Class distribution and train–test partition of the 903-cycle dataset used for model development and held-out evaluation.

Total cycles	Label 0 (good)	Label 1 (defect)	Positive rate	Train cycles	Test cycles	Split	Seed
903	488	415	0.46	632	271	*basic_stratified*	42

It is important to distinguish this offline model-development dataset from the later governed runtime evaluation records reported in Chapters 7 and 8. The 903-cycle dataset is used for statistical analysis, model development, cross-validation, and offline comparison of the event-anchored and full-curve modelling routes. By contrast, the live evaluation runs reported later contain only the cycles accepted, stored, and truth-linked during governed runtime execution. Therefore, the 903-cycle modelling dataset and the 850-cycle live evaluation records have different inclusion criteria and serve different purposes within this study.

Residual analysis revealed mild non-normality in the model residuals. The probability plot shows deviations in the tails, indicating skewness and outliers. The residuals versus fitted values plot shows a widening pattern, which is consistent with heteroscedasticity where variance increases at higher MP1 values. The residual histogram confirms a right skew, while the residuals versus order plot shows periodic variation that may indicate time-dependent drift during production. These findings reflect the dynamic nature of the injection moulding process and support retaining raw data rather than forcing the dataset to fit ideal linear assumptions. The residual diagnostic plots are shown in Figure 12.

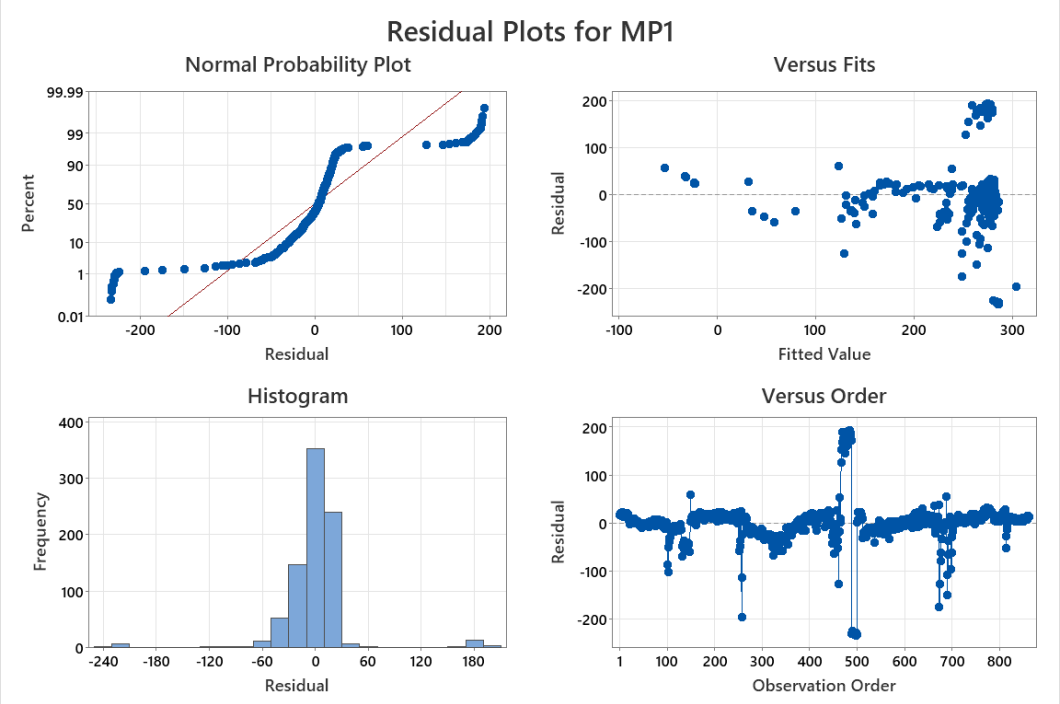


Figure 12. Residual diagnostic plots for the MP1 regression model, showing non-normality, heteroscedasticity, skewness, and possible time-dependent drift.

From a physical process perspective, the regression coefficients support known relationships in polymer processing. Hydraulic pressure showed a strong positive effect on MP1, which aligns with the expectation that higher injection force increases cavity pressure during packing. Screw position showed a negative regression coefficient. This relationship can be explained by the machine sensor definition and motion logic, where a higher SP can indicate material displacement that reduces resistance in screw movement, which in turn reduces internal

pressure. This highlights the need to interpret coefficients within the context of machine mechanics and sensor definitions.

Nozzle temperature also demonstrated a statistically significant negative relationship with MP1. This conforms with polymer melt behaviour, where increasing NT reduces viscosity and eases flow into the cavity. As flow resistance decreases, less pressure is needed for filling and packing, which reduces MP1. Nevertheless, excessively high NT can introduce other quality risks, so temperature control remains important for balancing flow and material stability.

The variable Time showed a meaningful statistical contribution to MP1. Its inclusion reflects the timing at which peak pressure occurs during the cycle. Variation in this parameter can be linked to process delays, material response changes, or cycle rate adjustments. Time, therefore, adds a temporal aspect of process stability, confirming that both the magnitude and the timing of process events influence pressure behaviour.

Although not directly modelled as a response, MT1 was monitored alongside MP1 at the same critical point. Cycles with irregular MT1 values, either elevated or delayed, often aligned with known defects. This suggests that poor heat dissipation or uneven mould temperature distribution may impact part quality, even when pressure levels appear within an acceptable range. Including MT1 enhances diagnostic interpretation by combining mechanical and thermal information at the same time.

This section's analysis answered several key questions. It examined the extent to which machine-side variables can predict cavity pressure during packing, which variables exert the most influence, and how their variation relates to defect likelihood. The regression outcomes provide statistical support and physical insight to guide subsequent modelling. The same variables are later used to classify and forecast part quality. By anchoring the analysis at the MP1 peak and using it as a consistent reference point across cycles, the approach remains relevant to production monitoring. It also simplifies the data representation used in later chapters and ensures that each part is represented by a comparable snapshot of its most critical process stage.

5.2 Data Preparation and Feature Description

The statistical models in this chapter were constructed using cycle-level records derived from the raw injection moulding signals. The purpose of this preparation is to ensure that each production cycle is represented in a consistent way so that comparisons between cycles are meaningful and repeatable. Injection moulding is a highly dynamic process, and the same part can be produced under slightly different timing and sensor conditions even when nominal set points are unchanged. For this reason, the analysis requires a representation that reduces sensitivity to sampling noise, small time shifts, and short transient events, while preserving the key mechanical and thermal behaviour that drives part formation.

A physically meaningful reference point in injection moulding is the transition from filling to packing. This transition is strongly associated with the moment when cavity pressure reaches its maximum, referred to here as the MP1 peak. At this stage, the polymer melt has fully filled the cavity, and the machine begins compressing the melt to compensate for volumetric shrinkage during cooling. The pressure state at this moment is closely linked to how well the cavity is packed, how voids are suppressed, and how dimensional stability is achieved. At the same time, the thermal state of the mould and melt affects viscosity, solidification rate, and internal stress development. The MP1 peak is therefore not only a signal feature but also a physical marker that aligns well with process understanding. It also tends to be clearly detected in sensor data and is less ambiguous than earlier stages, where signals may be dominated by transient flow effects.

The raw dataset contains many time-series rows per cycle. For modelling, the data were reduced to one structured record per accepted cycle using the MP1 cavity-pressure peak as the common physical anchor. The MP1 peak was first identified from the cavity-pressure signal, and the corresponding values of the other measured variables were then extracted at the same peak time/index. This means that hydraulic pressure, screw position, nozzle temperature, heating-water temperature, and mould temperature describe the process state at the MP1 peak, rather than their independent maximum values within the cycle. This peak-anchored representation supports reproducibility, reduces ambiguity caused by cycle-to-cycle timing shifts, and provides a compact feature set that can later be mapped to governed real-time deployment, where cycle-level decisions are required.

Table 11. Cycle-level feature definitions used for the statistical analysis, showing summary measures, signal source, and inclusion status.

Feature	Meaning	Source	Status
$\mathbf{HP_{mean}}$	Mean value over the cycle	Machine-side	Used
$\mathbf{HP_{peak}}$	Hydraulic pressure value at the MP1 peak time/index	Machine-side	Used
$\mathbf{HP_{min}}$	Minimum value within the cycle	Machine-side	Used
$\mathbf{SP_{mean}}$	Mean value over the cycle	Machine-side	Used
$\mathbf{SP_{peak}}$	Screw position value at the MP1 peak time/index	Machine-side	Used
$\mathbf{SP_{min}}$	Minimum value within the cycle	Machine-side	Used
$\mathbf{NT_{mean}}$	Mean value over the cycle	Machine-side	Used
$\mathbf{NT_{peak}}$	Nozzle temperature value at the MP1 peak time/index	Machine-side	Used
$\mathbf{NT_{min}}$	Minimum value within the cycle	Machine-side	Used
$\mathbf{HWT_{mean}}$	Mean value over the cycle	Machine-side	Used
$\mathbf{HWT_{peak}}$	Heating water temp value at the MP1 peak time/index	Machine-side	Used
$\mathbf{HWT_{min}}$	Minimum value within the cycle	Machine-side	Used
$\mathbf{MT1_{mean}}$	Mean value over the cycle	Cavity-side	Used
$\mathbf{MT1_{peak}}$	Cavity temperature value at the MP1 peak time/index	Cavity-side	Used
$\mathbf{MT1_{min}}$	Minimum value within the cycle	Cavity-side	Used
$\mathbf{MP1_{mean}}$	Mean value over the cycle	Cavity-side	Used
$\mathbf{MP1_{peak}}$	Maximum value within the cycle	Cavity-side	Used
$\mathbf{MP1_{min}}$	Minimum value within the cycle	Cavity-side	Used

For the baseline quality analysis in this chapter, part quality is treated as a binary outcome, where good is coded as 0, and defect is coded as 1. The selected process inputs include hydraulic pressure, screw position, nozzle temperature, and heating-water temperature as machine-side variables. These variables are measurable in production and are either directly controlled by the machine or strongly influenced by machine settings. Mould cavity pressure and mould cavity temperature are included as cavity-side variables because they represent the internal mould response at the point where packing behaviour becomes dominant. In addition to mean and minimum descriptors where required, the peak-labelled variables in this chapter refer to values associated with the MP1 cavity-pressure peak. $MP1_{peak}$ is the cavity-pressure peak value, while HP_{peak}, SP_{peak}, NT_{peak}, HWT_{peak}, and $MT1_{peak}$ are the corresponding values of those

variables at the same MP1 peak time/index. The full feature definitions used in this chapter are summarised in Table 11.

For HWT-related analysis, the first 12 cycles of Trial 1 were excluded because of the known calibration error described earlier.

Standard deviation features and cycle duration were excluded because they were constant for this dataset and, therefore, not informative.

Hydraulic pressure captures the applied force driving injection and packing, and it influences how material is pushed into the cavity and how packing pressure is sustained. Screw position reflects injection unit displacement and provides a direct indication of how the machine delivered material during the cycle. Nozzle temperature represents the melt thermal condition at injection and influences viscosity and flow resistance, which in turn affects pressure build-up and filling behaviour. Heating water temperature represents the mould thermal boundary condition and influences cooling, solidification, and shrinkage response, which can affect how pressure is retained during packing and how stresses develop within the part. Mould cavity pressure provides the cavity side response that reflects the achieved pressure state inside the mould, while mould cavity temperature provides thermal context for interpreting that pressure response. Together, these variables support a combined mechanical and thermal view of cycle behaviour that is relevant to quality formation and defect risk.

In summary, the dataset was prepared so that each accepted cycle is represented by a compact MP1 peak-anchored feature record, supported by additional cycle-level descriptors where required, for hydraulic pressure, screw position, nozzle temperature, heating-water temperature, cavity pressure, and cavity temperature. This provides a consistent basis for statistical modelling in Python and ensures that the results in the following sections are derived from the same reproducible feature extraction process. It also creates a clear link between physical process understanding and the modelling workflow, where the MP1 peak remains the key reference for interpreting the pressure and thermal behaviour that governs the transition into packing.

5.3 Correlation and Preliminary Data Exploration

The dataset was first examined using descriptive statistics and visual exploration to establish an evidence-based understanding of the cycle-level features used in

this chapter. This step is necessary for two reasons. First, it confirms whether the variables exhibit stable operating regions, skewness, or clustered settings that may affect how linear model coefficients are interpreted. Second, it highlights the presence of extreme cycles and distribution asymmetry that can influence modelling assumptions, particularly normality and constant variance, and that later motivates the use of more flexible modelling approaches. All statistics in this section refer to the same cycle-level dataset used throughout this chapter, where each cycle contributes one record. The dataset used here contains N = 903 cycles.

Mould cavity temperature and mould cavity pressure. The plots show that several variables do not follow an ideal normal distribution. Hydraulic pressure shows a dense operating region and occasional excursions, indicating that the process spends most of its time in a typical pressure setting while sometimes entering higher loading conditions. Screw position displays clustered and non-symmetric behaviour, which is consistent with frequent use of specific displacement settings rather than continuous variation. Nozzle temperature is relatively concentrated, indicating a tightly controlled thermal set point over most cycles. Heating water temperature also appears clustered, reflecting a small number of dominant thermal boundary settings with deliberate changes introduced during experimental variation. Mould cavity pressure shows a wider spread compared with the thermal variables, which is expected because cavity pressure responds strongly to coupled changes in injection force, motion behaviour, and melt condition. Mould cavity temperature remains relatively concentrated near its typical operating region while still showing measurable dispersion, which indicates that the cavity thermal state is generally stable but not constant.

This distribution evidence is important for later interpretation. Skewness and clustering indicate that the system operates in distinct regimes rather than behaving as a purely random process around a single mean. The presence of occasional extreme cycles is consistent with the decision to retain outliers as valid process behaviour rather than removing them, because these events represent realistic disturbances and boundary conditions that the later models must learn to handle.

To quantify linear associations between variables, Pearson correlation coefficients were computed for the cycle-level dataset. The correlation table provides an initial view of how machine-side variables move together and how they relate to the cavity-side response variables. These relationships are not used to claim causality. They are used to support the interpretation of later modelling

results and to highlight where multivariate effects are likely, particularly in a process where thermal and mechanical signals are coupled through machine control behaviour and material response.

For HWT-related analysis, the first 12 cycles of Trial 1 were excluded because of the known calibration error described earlier. The correlation table also supports the interpretation of thermal boundary effects. Heating water temperature can show a weaker direct linear association with cycle-to-cycle pressure behaviour because it often acts through a longer time scale thermal equilibrium rather than immediate pressure changes within a single cycle. Even when the direct correlation is modest, it remains important as a boundary condition that can influence shrinkage behaviour, residual stress formation, and dimensional stability. For this reason, it is retained for analysis and interpreted alongside the mechanical variables rather than in isolation.

Overall, the preliminary exploration confirms three key points that strengthen the modelling rationale in this chapter. First, the feature distributions indicate non-ideal behaviour, including skewness, clustering, and extreme cycles, which supports retaining outliers as representative manufacturing behaviour. Second, the correlation structure shows that several machine-side variables are coupled with the cavity response, which reinforces the need for multivariate interpretation rather than single variable reasoning. Third, the combination of regime-like distributions and coupled variables provides a clear justification for the modelling stages that follow in this chapter and for later chapters where interaction effects and non-linear behaviour must be captured more directly.

5.4 Logistic Regression Modelling

Logistic regression is used in this chapter because the quality outcome is binary, with each part classified as good or defective, and because the method provides an interpretable statistical link between the measured process variables and the probability of defect occurrence. Unlike the more flexible machine-learning models examined later in the thesis, logistic regression allows the direction and relative strength of each predictor association to be examined directly through the estimated coefficients while controlling for the other variables in the model. It therefore provides a transparent statistical baseline for assessing whether the selected cycle-level variables contain useful quality-related information and whether approximately linear relationships are sufficient to separate the two

quality classes. The model is not presented as the final deployment model of the thesis. Its role is to support statistical interpretation and to provide a reference against which the additional value of nonlinear machine-learning methods can later be assessed.

The analysis is conducted using the cycle-level dataset developed for this chapter, in which each production cycle contributes one structured record and part quality is treated as a binary outcome, with good coded as 0 and defect coded as 1. The predictors are derived from cycle-level features extracted from the measured injection moulding signals, as defined in Table 11.

5.4.1 Logistic regression between machine variables and quality

Logistic regression was applied as an interpretable baseline model to examine the association between selected cycle-level process variables and part quality. The response variable is binary, with parts labelled as good (0) or defective (1), making logistic regression an appropriate statistical method for estimating the probability of defect occurrence, $P(\text{defect} = 1)$, from the recorded process variables. The aim of this stage is not to claim causal effects or to establish the best-performing predictive model, but to provide a transparent statistical reference that helps interpret the recorded dataset and motivates the use of more flexible modelling approaches in the next chapter. The residual diagnostics reported earlier in this chapter relate to the exploratory linear regression model and therefore should not be interpreted as direct tests of the assumptions of the subsequent logistic regression model. Logistic regression does not require normally distributed residuals or constant residual variance. However, the observations arise from sequential production cycles, and complete independence between neighbouring cycles cannot therefore be assumed. Process persistence or gradual changes in operating conditions may introduce serial correlation between successive observations. Consequently, coefficient significance, standard errors, and associated statistical inference are interpreted cautiously, and the model is used primarily as an interpretable statistical baseline rather than as evidence of causal effects.

The modelling dataset used in this analysis is the structured cycle-level dataset adopted in this chapter, where each cycle contributes one record. The selected predictors represent machine-side and cavity-side process conditions extracted

in a consistent manner across cycles. The analysis retains disturbed and extreme cycles because these represent real operating behaviour that later modelling stages and the proposed governed real-time architecture are intended to handle. In this context, the MP1 peak should be understood primarily as an alignment landmark used to define a consistent reference point for cycle-level feature extraction, rather than as a variable assumed in advance to be the strongest standalone predictor of quality.

Logistic regression models the log odds of defect occurrence. The logit transform is defined as:

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right)$$

Equation 1. Logit transformation of defect probability

And the corresponding probability of defect occurrence is obtained using the inverse-logit function:

$$p = P(Y = 1) = \frac{1}{1 + \exp(-\eta)}$$

Equation 2. Defect probability from the linear predictor

where $p = P(Y = 1)$ is the probability that a part is defective (coded as 1) and $1 - p = P(Y = 0)$ is the probability that the part is good (coded as 0). The linear predictor η is expressed as a weighted combination of the input variables:

$$\eta = \ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 \text{HP}_{\text{peak}} + \beta_2 \text{SP}_{\text{peak}} + \beta_3 \text{NT}_{\text{peak}} + \beta_4 \text{HWT}_{\text{peak}} + \beta_5 \text{MT1}_{\text{peak}} + \beta_6 \text{MP1}_{\text{peak}}$$

Equation 3. Linear predictor for the MP1-anchored logistic regression model

$$\eta = \ln\left(\frac{p}{1-p}\right) = 0.06448 - 0.56778 \text{HP}_{\text{peak}} + 0.46746 \text{SP}_{\text{peak}} - 0.66089 \text{NT}_{\text{peak}} - 1.69890 \text{HWT}_{\text{peak}} - 0.39792 \text{MT1}_{\text{peak}} + 0.09768 \text{MP1}_{\text{peak}}$$

where β_0 is the intercept, and $\beta_1, \ldots, \beta_6$ are the regression coefficients associated with the six MP1-anchored process variables.

The estimated coefficients for the peak-anchored predictors are reported in Table 12.

Table 12. Logistic regression coefficients for defect probability using MP1 peak-anchored predictors.

Predictor and coefficient	Estimated coefficient	Standard Error	Test statistic	P-Value	Variance inflation factor (VIF)	95% CI for coefficient	Odds ratio	Interpretation
$\boldsymbol{\beta_0}$	0.06448	0.13807	0.467	0.641		−0.206 to 0.335	1.067	Baseline log odds when all predictors are at their scaled mean
$\mathbf{HP_{peak}}$, $\boldsymbol{\beta_1}$	-0.56778	0.359	-1.582	0.114	1.74	−1.271 to 0.136	0.567	Higher HP_{peak} is associated with lower defect odds, but not statistically significant.
$\mathbf{SP_{peak}}$, $\boldsymbol{\beta_2}$	0.46746	0.42923	1.089	0.276	1.11	−0.374 to 1.309	1.596	Higher SP_{peak} is associated with higher defect odds, but not statistically significant.
$\mathbf{NT_{peak}}$, $\boldsymbol{\beta_3}$	-0.66089	0.51319	-1.288	0.198	1.21	−1.667 to 0.345	0.516	Higher NT_{peak} is associated with lower defect odds, but not statistically significant.
$\mathbf{HWT_{peak}}$, $\boldsymbol{\beta_4}$	-1.6989	0.12138	-13.997	0.001	1.04	−1.937 to −1.461	0.183	A strong negative association was observed between HWT_{peak} and the model-estimated odds of a defective part, after accounting for the other predictors.
$\mathbf{MT1_{peak}}$, $\boldsymbol{\beta_5}$	-0.39792	0.11897	-3.345	0.000823	1.27	−0.631 to −0.165	0.672	A negative association was observed between $MT1_{peak}$ and the model-estimated odds of a defective part, after accounting for the other predictors.
$\mathbf{MP1_{peak}}$, $\boldsymbol{\beta_6}$	0.09768	0.28045	0.348	0.728	1.85	−0.452 to 0.647	1.103	Higher $MP1_{peak}$ is associated with slightly higher defect odds, but not statistically significant.

Multicollinearity was examined using the variance inflation factor (VIF) reported in Table 12. The VIF values for the six predictors range from 1.04 to 1.85, indicating limited inflation of the estimated coefficient variances due to correlation among the predictors. These values therefore provide no evidence of problematic multicollinearity in the fitted model. VIF should not be interpreted using a rigid universal threshold, but rather as an indicator of the extent to which each predictor is linearly explained by the remaining predictors (O'Brien, 2007). Although the low VIF values indicate that strong linear multicollinearity is not a major concern in the fitted model, they do not exclude other sources of dependence in the observed associations. The dataset contains deliberately introduced process disturbances and observations collected under different operating conditions, so some coefficient estimates may partly reflect the process regime or experimental disturbance under which the cycles were recorded. Unmeasured or incompletely represented process conditions may also contribute to apparent associations between individual predictors and quality. Together with the possible temporal dependence between successive cycles, these factors mean that the estimated coefficients are interpreted as conditional associations within the observed operating conditions rather than as isolated causal effects.

A positive coefficient increases the log odds of defect, while a negative coefficient reduces the log odds of defect, holding other variables constant. To support interpretation, coefficients are also reported as odds ratios,

$\exp(\beta)$, where values greater than 1 indicate increased odds of defect for an increase in the predictor, and values less than 1 indicate reduced odds of defect.

In this baseline model, the predictors are derived from hydraulic pressure, screw position, nozzle temperature, heating water temperature, mould cavity pressure, and mould cavity temperature. The predictors are derived from the MP1 peak-anchored feature definitions summarised in Table 11, where the machine-side and cavity-side variables are aligned to the MP1 peak event. This selection provides a combined mechanical and thermal view of cycle behaviour. Hydraulic pressure and screw position reflect injection and packing actuation and material delivery behaviour. Nozzle temperature reflects the melt thermal condition and flow behaviour. Heating water temperature reflects mould thermal boundary conditions and cooling behaviour. Mould cavity pressure represents the cavity response linked to packing performance, while mould cavity temperature provides thermal context inside the mould.

The logistic regression model can be expressed in the general form:

$$\eta = \text{logit}\big(P(Y=1)\big) = \beta_0 + \sum_{i=1}^{k} \beta_i \, x_i$$

Equation 4. General form of the logistic regression model

where β_0 is the intercept, x_i is the i predictor, β_i is its corresponding regression coefficient, and k is the total number of predictors. In this model,$k = 6$, and the fitted coefficients are reported in Table 12. The interpretation of the coefficients must be considered within the operating conditions represented in the dataset. Logistic regression provides a useful baseline because it gives a clear direction of association for each predictor and highlights which inputs are most strongly linked to defect occurrence in the recorded data. At the same time, the model structure remains linear in the logit space, so it may not capture regime-dependent behaviour and interactions that are known to occur in injection moulding. This limitation is addressed later by comparing against more flexible approaches under the same evaluation conditions.

Table 12 reports the fitted logistic regression coefficients, standard errors, test statistics, p-values, 95% confidence intervals, odds ratios, and variance inflation factors (VIF) for the cycle-level predictors. The table is used to identify the variables with the strongest association with defect probability and to confirm whether those associations are consistent with process understanding. Where a variable shows statistical significance, it is treated as a candidate contributor to defect risk within the observed operating window. Where variables show weaker significance, they are not dismissed as irrelevant. Instead, they are considered in the context of correlation structure, possible interactions, and the fact that some effects may be nonlinear or conditional on the operating regime.

5.4.2 Model Fit and Performance Evaluation

The fitted model was evaluated using a reproducible Python workflow applied to the cycle-level dataset of 903 cycles. Performance is reported on a held-out test subset of 271 cycles created using a stratified random split with seed 42. This separation prevents direct reuse of the same cycles for both training and testing. However, because injection moulding cycles are produced sequentially in time, a purely random split may still mix temporally adjacent cycles across the training and test sets. The reported results should therefore be interpreted as an

indicative baseline of model discrimination rather than a strict forward-in-time validation of predictive deployment performance. A more realistic temporal evaluation is introduced later when the models are assessed within the real-time monitoring environment.

Discrimination performance was assessed using the receiver operating characteristic curve. Figure 13 shows the ROC curve on the test set. The model achieves an AUC of 0.856 on the test set, which indicates good but not perfect class separation. The AUC result shows that the probability scores contain useful ranking information. This supports the use of probability-based decision policies rather than relying on a fixed hard rule.

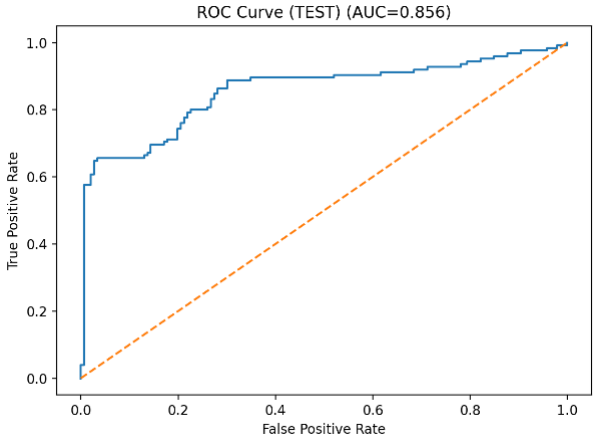


Figure 13. ROC curve for the held-out test subset of 271 cycles, indicating the discrimination performance of the fitted classification model (AUC = 0.856).

Precision and recall were also examined because the defect class is the operational priority in this work. Figure 14 shows the precision-recall curve on the same test set. This figure is important because it makes the trade-off clear. Increasing recall to reduce missed defects typically reduces precision and increases false alarms. This behaviour is expected in manufacturing quality monitoring where borderline cycles exist and class boundaries are not perfectly separable.

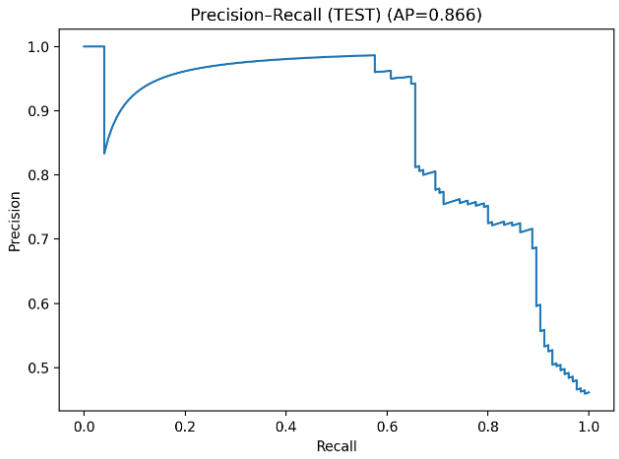


Figure 14. Precision-recall curve for the held-out test subset, illustrating the trade-off between detecting defective cycles and avoiding false alarms (AP = 0.866).

A threshold of 0.50 provides a standard reference point for classification. At threshold 0.50, the confusion matrix on the test set gives TN = 125, FP = 21, FN = 39, TP = 86. This corresponds to a recall for defects of 0.688 and a specificity of 0.856. In practical terms, the model correctly identifies many defective cycles, but it still misses a non-trivial number of defects at the default threshold.

Figure 15 shows the confusion matrix for the held-out test set at the default classification threshold of 0.50.

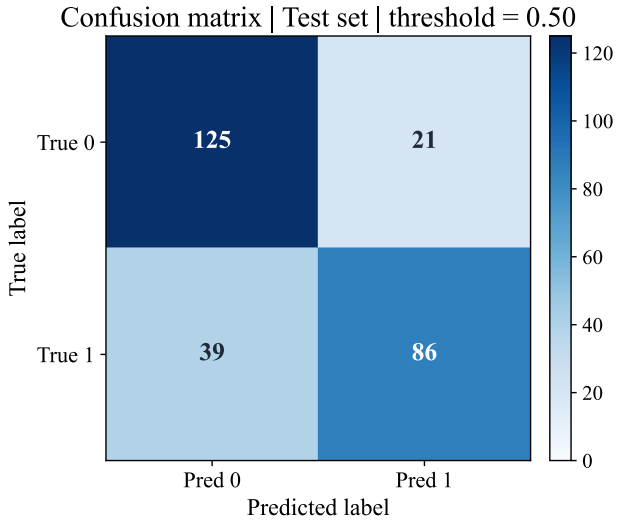


Figure 15. Confusion matrix for the test set at threshold 0.50.

Since the cost of missing a defect is typically higher than raising a false alarm, a second operating point was selected using a cost-based policy. In this thesis, the threshold was chosen by minimising expected cost using a 5-to-1 cost ratio for false negatives compared to false positives. This policy selects a threshold of 0.30. At threshold 0.30, the confusion matrix becomes TN = 96, FP = 50, FN = 14, TP = 111. This increases defect recall to 0.888 and reduces missed defects substantially, at the expense of more false alarms and a lower specificity of 0.658. This change is consistent with a manufacturing setting where preventing defect escape is prioritised. Figure 16 shows the confusion matrix for the held-out test set at the selected operating threshold of 0.30.

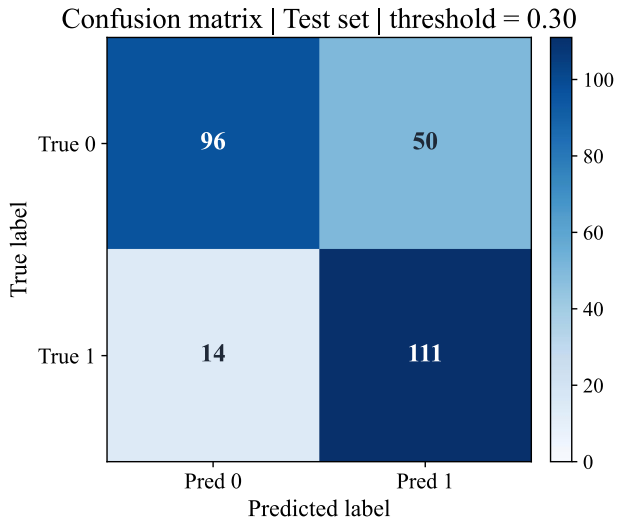


Figure 16. Confusion matrix on the test set at the selected operating threshold of 0.30.

The effect of the threshold on performance was explored using a sweep across thresholds. Figure 17 summarises how precision, recall, and F1 change with threshold on the test set. This figure is included to make the decision policy transparent. It also explains why a model can have a reasonable AUC while still producing an undesired number of false negatives at a threshold of 0.50. AUC measures ranking quality across all thresholds, while false negatives depend on the specific operating threshold.

Figure 17 shows the threshold sweep used to examine how precision, recall, and F1 score change across different classification thresholds.

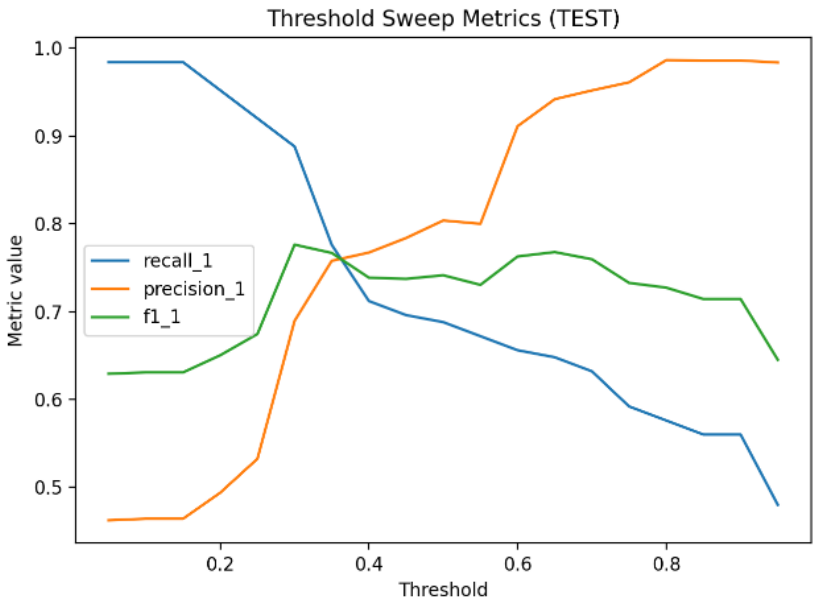


Figure 17. Threshold sweep on the test set.

Calibration was evaluated using the Brier score and a reliability diagram. The test-set Brier score is 0.1455, which suggests moderate probability quality. Figure 18 shows the calibration curve for the test set. Table 13 confirms that probability quality varies across the score range. Some mid-range bins show clear gaps between predicted probability and observed defect rate. This means the probabilities should be used with care in borderline regions, and threshold selection should be justified explicitly rather than assumed.

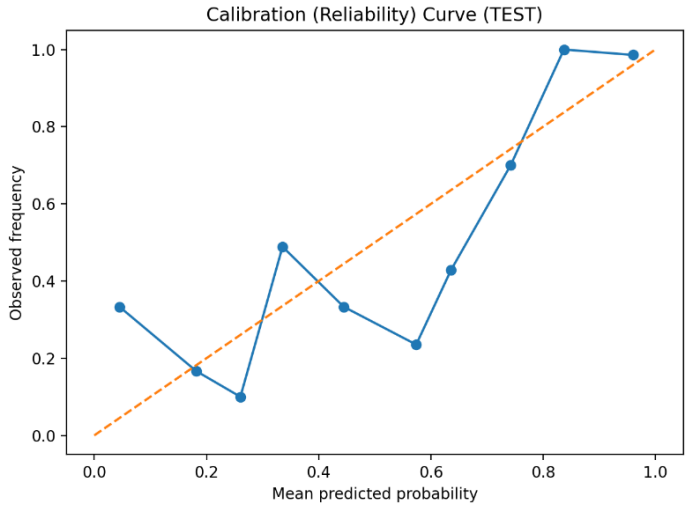


Figure 18. Reliability diagram for the test set.

Table 13. Binned calibration results for the test set.

Predicted probability bin p	Number of cycles n	Observed defect rate $\hat{p}_{obs}$	$\bar{p}$	p_{min}	p_{max}
$[0.0, 0.1]$	6	0.333	0.0448	3.30E-60	0.0892
$(0.1, 0.2]$	24	0.166	0.1816	0.1544	0.1991
$(0.2, 0.3]$	80	0.1	0.2604	0.2007	0.2997
$(0.3, 0.4]$	45	0.488	0.3355	0.3000	0.3885
$(0.4, 0.5]$	9	0.333	0.4444	0.4022	0.4899
$(0.5, 0.6]$	17	0.235	0.5737	0.5006	0.5975
$(0.6, 0.7]$	7	0.428	0.6358	0.6001	0.6945
$(0.7, 0.8]$	10	0.7	0.7420	0.7015	0.7836
$(0.8, 0.9]$	2	1	0.8370	0.8271	0.8469
$(0.9, 1.0]$	71	0.985	0.9602	0.9448	0.9999

The key test-set results are summarised in Table 14. This table reports AUC and Brier score, and it reports confusion matrices and derived rates at threshold 0.50 and at the selected threshold of 0.30. The purpose is to present the operating trade-off clearly and to show how threshold choice changes the balance between missed defects and false alarms.

Table 14. Test-set performance at threshold 0.50 and at the selected operating threshold.

Metric	General/test-set value	Default threshold τ=0.50	Selected threshold τ=0.30
Total cycles, N_{total}	903	—	—
Test cycles, N_{test}	271	—	—
Overall positive rate, $\hat{p}_+$	0.4596	—	—
Test ROC-AUC	0.8562	—	—
Test Brier score	0.1455	—	—
Threshold-selection policy	Minimum expected cost	—	Applied
False-negative cost, c_{FN}	5	—	Applied
False-positive cost, c_{FP}	1	—	Applied
True negatives,TN	—	125	96
False positives,FP	—	21	50
False negatives,FN	—	39	14
True positives,TP	—	86	111

Overall, the evaluation presents a consistent pattern across the reported performance measures. The model has useful discrimination capability on unseen cycles, but the probability quality is not uniform across the range. The default threshold of 0.50 produces a meaningful number of missed defects. A lower threshold improves defect recall substantially, which is more appropriate when defect escape is costly. These findings justify the next chapter, where more flexible models are tested under the same fixed evaluation procedure and where operating policies can be selected transparently and defensibly.

5.5 Summary

This chapter used a peak-anchored, cycle-level representation to provide a transparent statistical baseline for understanding how injection moulding conditions relate to part quality. The evaluation confirms that probability-based classification is feasible using the adopted dataset of 903 cycles and a fixed held-out test set. The ROC analysis shows that the model can separate good and defective cycles with useful ranking ability on unseen data, with an AUC of 0.856 on the test set. This supports the idea that the selected process signals contain meaningful quality information even when the process operates with drift, regime-like behaviour, and occasional extreme cycles.

At the same time, the results show that classification behaviour depends strongly on the chosen decision threshold. Using the standard threshold of 0.50 produces a balanced result, but it still misses a noticeable number of defects. When the operating point is adjusted to prioritise defect detection, the threshold can be reduced to improve recall and reduce false negatives, at the cost of more false alarms. This trade-off is expected in manufacturing settings where the cost of defect escape is often higher than the cost of inspection or rework triggered by a false alarm. Presenting both operating points makes the decision policy explicit and prevents the evaluation from being reduced to a single accuracy value.

Calibration analysis also highlights an important practical point. Although discrimination is reasonable, probability quality is not uniform across the full score range. The calibration table and reliability diagram show that mid-range probabilities can deviate from observed defect rates. This means that the probability output should not be treated as perfectly calibrated uncertainty without further checks. Instead, probability should be used as a risk score,

supported by calibration reporting and by an operating threshold justified against manufacturing priorities.

The coefficient analysis provides an interpretable statistical reference for later modelling. In the simplified peak-variable logistic model, heating water temperature and cavity temperature show strong associations with defect probability, while the remaining peak-only predictors are weaker when fitted in a linear logit form. This outcome does not imply that those variables are unimportant. It indicates that defect mechanisms are likely driven by interactions, non-linear effects, and regime changes that are not fully captured when only peak values and linear decision boundaries are used. This supports the need to move beyond a single linear model and to evaluate models that can learn interaction structure from the same cycle-level data.

Overall, the chapter establishes three implications for the next chapter on machine learning. First, evaluation must remain reproducible and test-based, with the same split logic and clear reporting of which subset each metric refers to. Second, threshold selection must be treated as part of the decision policy, not an arbitrary default. Third, machine learning models should be compared under the same evaluation framework and should be expected to improve on the baseline by capturing non-linear relationships, feature interactions, and regime-dependent behaviour that are typical in real injection moulding production data.

Chapter 6
Machine Learning for Cycle-Level Quality Classification in Cyclic Manufacturing

6.1 Introduction

This chapter introduces the machine learning framework used for cycle-level quality prediction within the monitoring system. The objective is to evaluate whether process signals captured during each production cycle can be used to classify part quality reliably under realistic operating variation. Two complementary modelling pathways are explored. The first uses a compact event-anchored feature representation derived from the peak cavity pressure anchor defined in Chapter 4. The second retains the full time-series representation of the process signals to preserve richer temporal information. These two representations provide different trade-offs between interpretability, computational cost, and sensitivity to dynamic process behaviour.

Chapter 5 established a statistical basis for this work by analysing the relationship between key process variables and part quality. In particular, it showed that the process state at the moment of peak mould cavity pressure is strongly associated with the final quality outcome in the injection moulding case study. However, regression analysis is limited because it primarily examines linear relationships and individual variable effects. Many manufacturing defects arise from coupled behaviour, interactions between variables, and nonlinear responses across stages of the cycle. For this reason, this chapter moves from statistical association to predictive modelling and focuses on supervised classification methods that can learn nonlinear relationships from labelled data.

The primary purpose of this chapter is to provide engineering evidence that machine learning can predict part quality at the cycle level using data collected during production. The case study used throughout the chapter is injection moulding, but the modelling problem is framed in a general way so that the reasoning and conclusions apply to cyclic manufacturing more broadly. The models are trained using labelled cycles, where each cycle is linked to a binary quality label, good or defective. The outputs of the models are then assessed using evaluation methods designed for imbalanced industrial datasets, where defective parts are less frequent but more costly.

Cycle-level quality prediction can be approached using either complete process signals or more compact feature-based representations, creating a practical trade-off between information retention, input complexity, robustness, and computational demand. A full cycle contains multivariate time series signals, and these signals can be used directly as model inputs. However, this can introduce

practical difficulties, including variation in cycle length, sensitivity to cycle alignment, and increased computational cost. An alternative is to represent each cycle using a compact set of features extracted at a meaningful event within the cycle. In the present injection moulding case study, the highest cavity-pressure value within each cycle is used as the reference event. Under the characteristic cycle morphology examined in this study, this maximum occurs around the transition from filling to packing and reflects the combined process state developed up to that point. Using this event as a common temporal anchor allows a compact, synchronised process snapshot to be extracted across multiple channels.

Based on this, the chapter tests two modelling routes. The first route uses an event-anchored, peak-based representation, where each cycle is summarised by the synchronised process values sampled at the cavity pressure peak. The second route uses a full curve representation, where the complete multivariate cycle curves are aligned and resampled to a common length to support sequence-based learning and fair comparison across cycles. By evaluating both routes using the same dataset and validation protocol, the chapter quantifies how close the compact peak-based approach can come to full curve-modelling performance and identifies the trade-off between input complexity and classification capability. This comparison directly addresses the second research question, which was formulated from the research gaps identified in Chapter 2.

This chapter is structured as an offline modelling study. It focuses on dataset preparation, cycle representation, model selection, training and validation, and results reporting using reproducible evaluation procedures. The outcome of the chapter is a clear set of validated results, supported by tables and figures in later sections, showing which models perform best under each representation and how the two routes compare. Chapter 7 then builds on this evidence by integrating the selected modelling approach into the governed real-time architecture for quality monitoring and decision support. The broader governance and deployment logic are therefore presented in Chapter 7, while Chapter 6 provides the modelling foundation and the quantitative evidence needed to justify those later system choices.

6.2 Supervised learning for cyclic manufacturing quality

Supervised learning is a practical method for quality prediction in cyclic manufacturing because it uses examples where both the process measurements and the final quality outcome are known. In manufacturing terms, the model is trained using historical production records, where each record represents one completed cycle and one produced unit, together with an inspection-based quality label. After training, the model can estimate the likely quality class of new cycles, using only the process measurements collected during production. This is useful in industrial settings because it supports earlier quality decisions, reduces reliance on delayed inspection, and provides a consistent decision rule that can be applied across long production runs.

In a cyclic process, each cycle produces a single unit, and the decision that matters is whether that unit is acceptable or not. For this reason, the learning objective in this thesis is defined at the cycle level rather than at the time sample level. A cycle-level decision aligns with industrial acceptance because quality control actions are typically taken per unit. Examples include rejecting a part, stopping the machine, adjusting settings, or triggering further inspection. Predicting quality at the cycle level also supports traceability because each quality decision can be linked to a cycle identifier and to the process history of that cycle.

The supervised classification problem can be defined as follows. For each cycle i, an input representation is constructed from the measured process signals, and the target output is the corresponding quality label. The goal is to learn a function that maps the cycle input to the quality class. In many industrial applications, the class label is binary, meaning that each unit is classified as good or defective. In this thesis, the injection moulding case study follows the same principle, where each moulding cycle produces one part, and each part is assigned a binary quality label based on inspection outcomes.

A useful way to represent the output of a classifier is through a probability or confidence score. Many classification models estimate the probability that a cycle belongs to the defective class, rather than outputting a hard decision directly. This probability can be interpreted as a risk score that summarises the likelihood of a defect given the measured cycle data. In production terms, a probability output is often more informative than a single label because it allows engineers to adjust the operating point of the decision rule. For example, a plant may choose a

conservative setting that prioritises defect capture, even if it increases false alarms, or a stricter setting that reduces false alarms when inspections are expensive or disruptive.

Equation 5 defines the predicted defect probability p_i produced by the binary classifier. For cycle i, the model estimates the probability that the corresponding part belongs to the defective class:

$$p_i = P(y_i = 1 \mid \mathbf{x}_i)$$

Equation 5. Predicted defect probability for cycle i

where p_i is the predicted probability that cycle i is defective, $y_i = 1$ denotes the defective defective class, and $\mathbf{x_i}$ is the input representation constructed for that cycle. This probability can then be converted into a binary decision using a threshold. The threshold is a design choice that determines how sensitive the system is to defects. A lower threshold flags more cycles as defective, increasing recall but potentially increasing false positives. A higher threshold flags fewer cycles as defective, increasing precision but potentially missing true defects. The choice of threshold is therefore part of the classifier's engineering design and should be justified using performance metrics that reflect the manufacturing objective.

Equation 6 defines the decision rule used to convert the predicted probability $p_i$into the binary class prediction $\hat{y}_i$. Given a classification threshold τ, the predicted class for cycle $i$is:

$$\hat{y}_i = \begin{cases} 1, & p_i \geq \tau \\ 0, & p_i < \tau \end{cases}$$

Equation 6. Binary classification decision rule

where $\hat{y}_i = 1$denotes a predicted defective cycle, $\hat{y}_i = 0$denotes a predicted good cycle, and τ is the selected classification threshold.

This formulation also clarifies why accuracy alone is insufficient for typical manufacturing datasets. In real production, defective parts are usually a minority because the process is controlled to produce acceptable output most of the time. This can lead to class imbalance in many manufacturing datasets, where good cycles may outnumber defective cycles. Under an imbalance, a classifier can achieve high accuracy by predicting most cycles as good yet still fail to reliably

detect defects. From an engineering standpoint, this failure is unacceptable because the cost of passing defective parts can be significantly higher than the cost of additional inspection or occasional false alarms. For this reason, evaluation must consider how well the model identifies defects, not only its overall accuracy. This conceptual workflow for cycle-level supervised classification is illustrated in Figure 19.

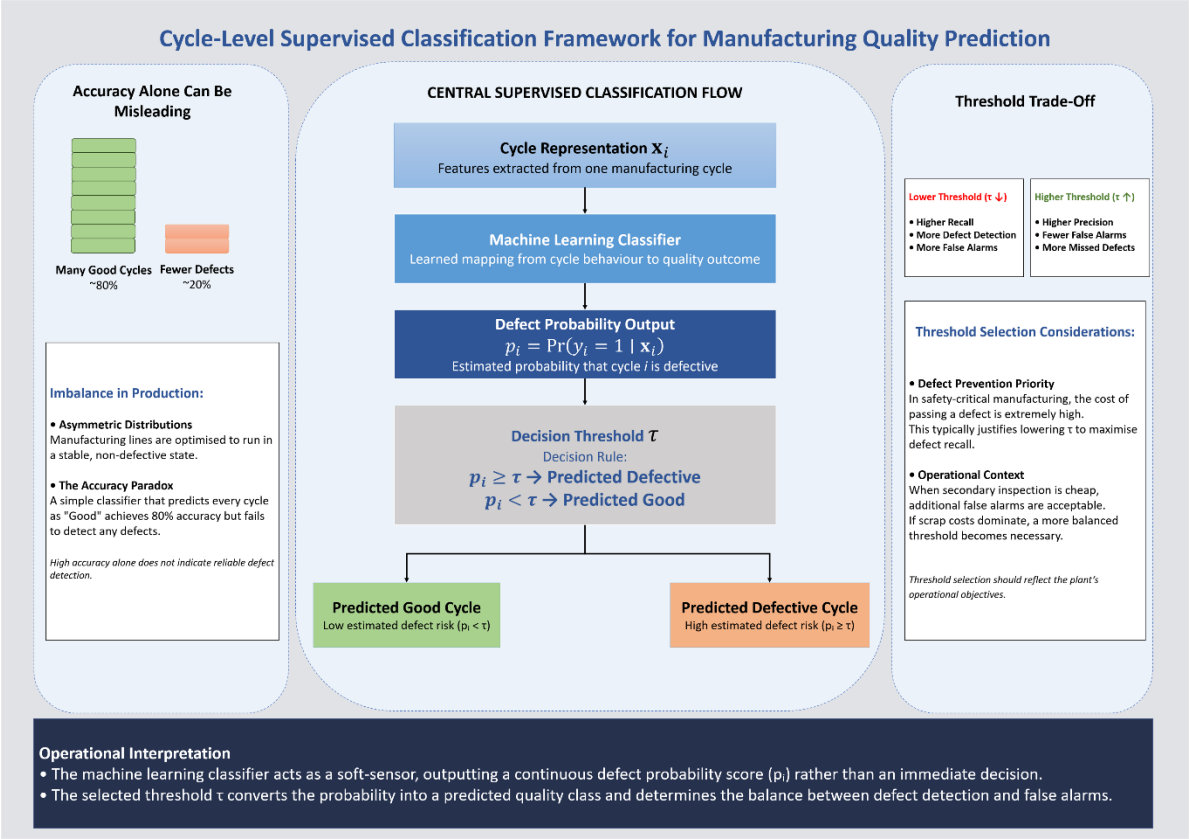


Figure 19. Cycle-level supervised classification framework for manufacturing quality prediction. Structured representations of manufacturing cycles are evaluated by a supervised classifier that estimates the probability of a defect. A decision threshold then converts this probability into a predicted quality class, illustrating the trade-off between defect detection and false alarms.

A schematic showing one manufacturing cycle as input data, the classifier output as a predicted probability of a defective part and the threshold τ converting this to a final class decision $\hat{y}$.

In the injection moulding case study, the cycle input is derived from signals captured during the moulding cycle. These include machine level signals, such as hydraulic pressure and screw position, and mould-level signals, such as cavity

pressure and cavity temperature. Each cycle corresponds to one produced part, and the target label reflects inspection outcomes. Mapping this case study to the general cyclic manufacturing framework clarifies the main point of the chapter. The learning problem is not specific to injection moulding. It is a general cycle level quality classification problem where the aim is to predict acceptability from measured process behaviour, and to produce an interpretable risk score and decision rule that can later be embedded within a real-time monitoring system.

6.3 Dataset and case study description

Injection moulding is a suitable case study because it is inherently cyclic, each cycle produces one part, and the process can be monitored using both machine-level sensors and mould cavity instrumentation. At the same time, the data structure and the learning problem are not unique to injection moulding. The same principles apply to other cyclic manufacturing processes where sensors provide process traces, and each cycle ends with a unit-level quality outcome.

The dataset was collected on the shop floor across three production campaigns. This is important because it introduces realistic variability that is difficult to capture in short laboratory trials. In industrial production, process conditions change over time due to planned parameter adjustments, material batch variation, interruptions, maintenance actions, and gradual sensor drift. By combining data from multiple campaigns, the dataset reflects these natural sources of variation and provides a stronger basis for evaluating whether a model can generalise across realistic operating conditions rather than memorising a single stable run. This campaign structure also supports a more credible comparison between modelling approaches, because it tests performance under conditions that resemble real manufacturing deployment. Each observation in the dataset corresponds to one complete injection moulding cycle, and therefore to one produced part. A cycle is defined as the sequence of events that begins when the mould closes and the injection stage starts, continues through filling and packing, and ends when the mould opens and the part is ejected. Cycle boundaries were determined using the machine and process signals that mark the start and end of the cycle. Once a cycle boundary is detected, all sensor readings within that interval are grouped as one cycle record. This cycle-level segmentation is essential because it ensures that the model inputs align with the manufacturing decision unit. The model is trained to classify a full cycle, and the

predicted label is interpreted as the predicted quality of the part produced in that cycle.

Six process signals were recorded for each cycle. These include hydraulic pressure, screw position, nozzle temperature, heating water temperature, mould cavity pressure, and mould cavity temperature. The cavity pressure signal is treated as a key indicator because it reflects the material behaviour inside the mould. In particular, the highest cavity-pressure value within each cycle is used as a repeatable reference event. Under the characteristic cycle morphology represented in the dataset, this maximum occurs around the transition from filling to packing and provides a common temporal point for synchronising the remaining process signals. The other signals provide complementary information about machine behaviour and thermal conditions during the same cycle.

Table 15. sensor channels and their role in the cycle.

Variable	Full Name	Measurement Source	Typical Unit	Physical Meaning	Role in the Cycle
HP	Hydraulic Pressure	Hydraulic pressure sensor	bar	Pressure generated by the hydraulic system during machine operation	Indicates machine loading and injection-force behaviour
SP	Screw Position	Position transducer	mm	Axial position of the injection screw	Represents material movement and injection progression
NT	Nozzle Temperature	Nozzle thermocouple	°C	Melt temperature before entering the mould	Indicates thermal state of the molten polymer
HWT	Heating Water Temperature	Heating-water sensor	°C	Temperature of the mould heating/cooling circuit	Reflects thermal regulation of the mould
MP1	Cavity Pressure	Kistler cavity pressure sensor	bar	Pressure inside the mould cavity	Primary indicator of filling, packing, and cavity behaviour
MT1	Cavity Temperature	Kistler cavity temperature sensor	°C	Temperature inside the mould cavity	Indicates thermal conditions during filling and cooling

Table 15 lists HP, SP, NT, HWT, MP1, and MT1, with a short description of what each signal represents physically, the measurement location or source, the typical

unit used in the acquisition system, and its role within the cycle, such as hydraulic loading, material flow position, thermal condition, or cavity state.

Each cycle was assigned a binary quality class. The label represents the final part outcome for that cycle, where 0 indicates a good part and 1 indicates a defective part. Labelling was based on the inspection criteria used during production. This includes dimensional checks and weight measurements, supported by visual inspection for surface defects. Defective parts include outcomes such as flash, short shots, and sink marks. These defect types are relevant because they are linked to filling and packing behaviour, thermal stability, and pressure evolution, which are all reflected in the recorded sensor signals.

The final offline modelling dataset used in this study contains 903 complete cycles. Of these, 488 cycles were labelled as good parts (Quality = 0), and 415 cycles were labelled as defective parts (Quality = 1). This distribution is near-balanced, with good parts forming a slight majority. Although the dataset is not strongly imbalanced, defective parts remain critical to detect because missed defects can lead to scrap, rework, or downstream quality failures. This distribution is near-balanced, with good parts forming only a slight majority. Although the dataset is not strongly imbalanced, defective parts remain critical to detect because missed defects can lead to scrap, rework, or downstream quality failures. This motivates the evaluation approach used later in the chapter, since a model that performs well on average may still be unsuitable if it fails to detect defective cycles reliably. A visual comparison of the six monitored variables for both a representative good and defective cycle is shown in Figure 20.

This dataset description provides the foundation for the modelling work in the following sections. It clarifies the industrial context, the cycle definition, the measured variables, the labelling criteria, and the realistic variability included through multiple production campaigns. Based on this dataset, the next section defines how each cycle is represented for modelling, including the event-anchored peak-based representation and the full curve representation used for comparison in this chapter.

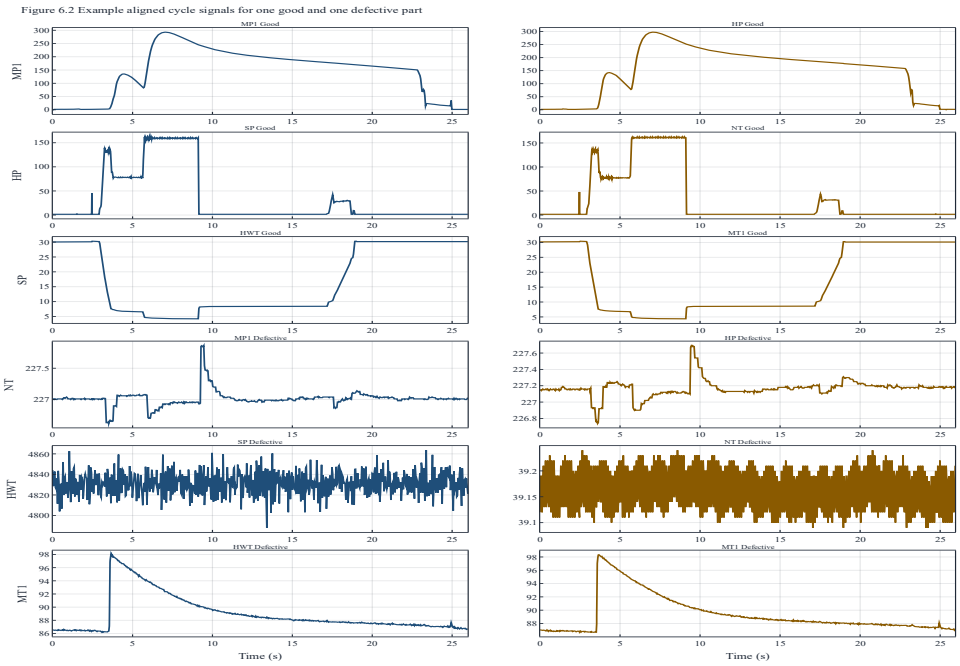


Figure 20. Example good and defective injection moulding cycles across the six monitored process variables.

6.4 Cycle representations and the two modelling routes

This section defines how each manufacturing cycle is represented for machine learning and explains why two modelling routes are required in this study. The representation choice is not a formatting detail. It changes what information is available to the model, how sensitive the model is to segmentation and timing errors, and how practical the approach becomes when applied to long production runs. The main purpose of this section is to justify, in engineering terms, why an event-anchored peak-based representation is a valid and useful approach, and why it should be compared directly against a full curve representation as a reference.

6.4.1 Why a cycle representation is needed

In cyclic manufacturing, sensors often produce multivariate time series signals, sampled continuously during production. If these signals are used directly, each cycle is represented by a set of curves. This representation contains rich information, but it also introduces several practical and methodological challenges. The first challenge is that raw curves are long. A single cycle may

contain hundreds or thousands of time samples per channel, and with multiple channels, the total number of data points per cycle becomes large. As a result, the input dimension increases quickly, which raises computational cost and increases the risk that a model learns noise or runs specific patterns rather than generalisable features.

A second challenge is variable cycle length. Even when a process is nominally stable, cycle duration can change due to machine behaviour, material variation, operator intervention, or control actions. This means that different cycles contain different numbers of samples and different time alignments. Many machine learning models require fixed-size inputs. Without additional processing, variable-length cycles cannot be compared directly and cannot be used consistently in training. This is one reason why cyclic manufacturing studies often convert curves into engineered features or apply alignment and resampling steps before modelling.

A third challenge is sensitivity to cycle segmentation and alignment. Cycle boundaries are detected from process signals and are rarely perfect. Small errors in identifying the start or end of a cycle, or small timing offsets between channels, can change the apparent shape of the curve and can shift key events within the cycle. When the full curve is used as input, these segmentation errors can affect a large fraction of the data, because the model sees the entire curve. This sensitivity is especially important when the learning objective is to detect defects, because defects may be relatively subtle and can be masked or distorted by misalignment. For this reason, representation design must consider robustness to segmentation imperfections.

A fourth challenge is the compute and storage burden. Full curve representations require storing and processing all sampled points for each channel and each cycle. Over long production periods, this can become costly, particularly when high sampling rates are used. Even in offline studies, this affects training time, memory requirements, and the feasibility of testing multiple models and validation folds. In later chapters, where real-time integration is considered, the same burden can affect response time and system complexity. For this reason, it is valuable to explore compact representations that reduce data volume while retaining the most informative process content.

Finally, there is a methodological need for fair comparison between representations. If different representations are evaluated using different

datasets, different splits, or different preprocessing rules, it becomes difficult to conclude whether differences in performance are due to the representation itself or due to confounding factors. In this chapter, the aim is to compare two routes that are built from the same cycles and the same labels, so that the only major difference is the representation and the model family that matches it. This creates a clear test of the research question, whether a compact peak anchored representation can approach the performance of full curve modelling. These representation choices are conceptually illustrated in Figure 21.

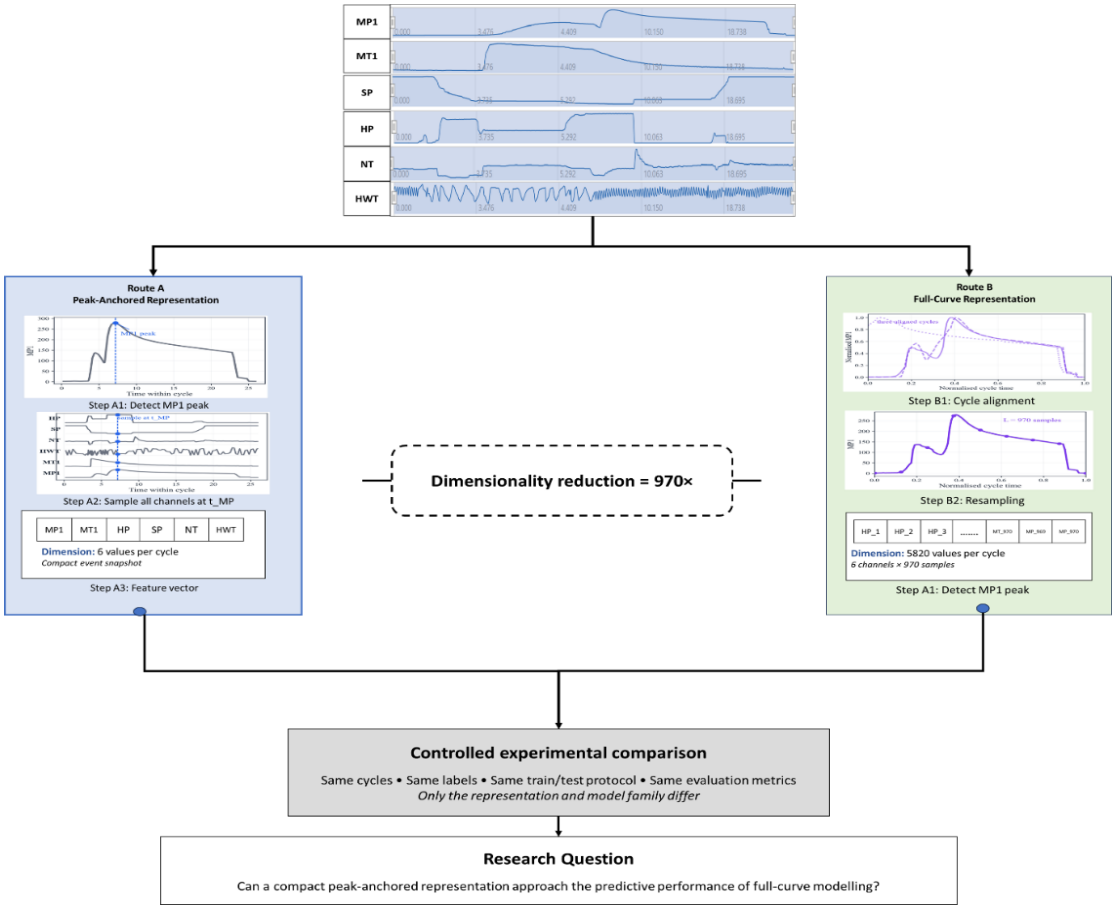


Figure 21. Comparison of the peak-anchored and full-curve cycle representations evaluated using the same cycles, labels, train/test protocol, and metrics.

Based on these reasons, this chapter defines two modelling routes. One route represents each cycle using a compact event-anchored feature vector, and the other route represents each cycle using the complete multivariate curves after alignment and resampling. The next subsections describe these two routes in detail and explain how they are constructed consistently from the same underlying cycle data.

6.4.2 Event-anchored peak-based representation

The event anchored peak-based representation summarises each cycle using a physically meaningful reference event and the synchronised process state at that moment. The purpose is to reduce each cycle to a compact set of values that remain comparable across cycles, while still retaining information that is strongly linked to part formation. This representation is designed for cyclic manufacturing processes where a repeatable event exists within the cycle and where the process state at that event is expected to reflect the combined outcome of upstream conditions.

The event-anchored representation developed in this research builds on established evidence that characteristic points and physically meaningful features extracted from injection moulding process signals can carry quality-relevant information. The contribution claimed here is therefore not the general use of cavity pressure, peak values, or process-event features. Instead, the proposed representation uses one physically defined event in the cavity-pressure signal as a common temporal anchor for the monitored process state. At the maximum cavity-pressure event, the concurrent values of the other process channels are sampled at the same time index to form a synchronised multivariable representation of the cycle. This compact representation is then evaluated directly against a full-curve representation under the same cycle-level prediction framework. The contribution therefore lies in the construction and evaluation of this synchronised event-anchored representation rather than in the individual use of a cavity-pressure peak as a quality indicator.

In the injection moulding case study, the anchor event is defined as the highest peak of the mould cavity pressure signal MP1 within the cycle. This maximum peak value is written as $MP1_{peak}$, and the time at which it occurs is written as t_{MP}. The maximum cavity-pressure event is cycle critical because it represents the strongest cavity pressure response reached during the cycle and reflects the combined outcome of filling and packing behaviour. During filling, the polymer melt advances through the cavity, and the pressure rises as flow resistance increases. In some cycles, small earlier peaks or local fluctuations may appear during the injection phase. These smaller peaks may reflect transient behaviour, local flow effects, or early pressure variations before the cavity reaches its main loading condition. For this reason, the highest peak is selected as the anchor event rather than the first local peak. The highest peak is more directly associated with the final cavity loading condition and therefore provides a stronger physical basis

for quality prediction. It captures the combined effect of machine settings, melt temperature, viscosity behaviour, and cavity restrictions at the most significant pressure state reached within the cycle. As a result, it concentrates decisive quality information because many common injection moulding defects are sensitive to the conditions around the maximum cavity pressure condition. For example, short shots are linked to insufficient filling, flash can occur when pressure and clamping conditions interact and sink marks can be influenced by packing pressure and thermal behaviour. Although defects can also depend on later stages, the maximum cavity-pressure event provides a strong and physically meaningful snapshot of the cycle state.

A second reason for selecting this anchor is repeatability. The cavity pressure signal typically contains a clear dominant peak structure in normal cycles, even when smaller fluctuations appear elsewhere in the curve. This makes the maximum cavity-pressure event easier to define consistently than an arbitrary time-based sampling point. Using a process-defined event rather than a fixed time index also reduces sensitivity to cycle time variation. The event occurs when the process reaches a specific physical condition, not when the clock reaches a specific value. This supports interpretability because the extracted features correspond directly to a meaningful cavity response state and can be discussed using process physics rather than purely statistical reasoning.

The peak-based representation is constructed using a consistent rule applied to each cycle. First, the cycle boundaries are used to isolate the $MP1$curve, after which the implemented method selects the global maximum using argmax($MP1$) In this injection-moulding case study, this maximum is not treated as an arbitrary numerical feature, but as the characteristic dominant cavity-pressure event around which the labelled cycle patterns were established. Normal cycles exhibit a clear dominant pressure rise and maximum, whereas a markedly reduced peak, two comparable dominant peaks, or a shift in which a different physical pressure event becomes the maximum represents a departure from the normal cycle morphology rather than an alternative valid anchor. Such departures are therefore presented to the trained prediction routes as abnormal cycle behaviour: the full-curve Champion evaluates the changed pressure profile, while the Challenger evaluates the process state extracted at the selected $MP1$maximum. If the two predictive routes disagree, the Final policy resolves the cycle conservatively as defective. A small noisy fluctuation does not affect the anchor unless it exceeds the main $MP1$ maximum; if it does, argmax($MP1$) will select that

point. Signal clipping or sensor drift are treated differently because they may alter the measured morphology without representing a genuine second physical pressure event; in such cases, the selected numerical maximum should not automatically be interpreted as equivalent to the normal $MP1$ anchor.

A formal sensitivity comparison between alternative peak definitions was not undertaken because the event-anchored models in this study were developed and evaluated consistently using the highest $MP1$maximum as the predefined process anchor; assessment of alternative event definitions would therefore constitute a separate modelling investigation.

Once the peak is detected, the peak value $MP1_{peak}$ is recorded, and the time of the peak is recorded as t_{MP}. The key step then is synchronised sampling. At the time index t_{MP}, the concurrent values of the other process channels are extracted from their respective signals within the same cycle. This produces a compact cycle snapshot that includes the machine and thermal state at the same physical moment identified by the maximum cavity-pressure event. In the injection moulding case study, the sampled values include hydraulic pressure, screw position, nozzle temperature, heating water temperature, mould cavity temperature, and the cavity pressure itself at the peak. These values form the peak-anchored feature vector for the cycle. The structure of this event anchored feature vector is summarised in Table 16.

Table 16. Structure of the event anchored feature vector.

Feature name	Symbol/definition	Physical meaning	Role in the peak-based route
Hydraulic pressure at the anchor event	$HP_{peak,i} = HP_i[k_{MP1,i}]$	Hydraulic pressure measured at the time of the maximum cavity-pressure event	Captures the machine side loading state at the most significant cavity pressure condition in the cycle
Screw position at the anchor event	$SP_{peak,i} = SP_i[k_{MP1,i}]$	Screw position measured at the time of the maximum cavity-pressure event	Represents the material feed position and volumetric process state at the anchor event
Nozzle temperature at the anchor event	$NT_{peak,i} = NT_i[k_{MP1,i}]$	Nozzle temperature measured at the time of the maximum cavity-pressure event	Captures the melt thermal condition at the nozzle when the dominant cavity pressure response occurs
Heating water temperature at the anchor event	$HWT_{peak,i} = HWT_i[k_{MP1,i}]$	Heating water temperature measured at the time of the maximum cavity-pressure event	Represents the mould thermal control condition at the anchor event

Feature name	Symbol/definition	Physical meaning	Role in the peak-based route
Mould cavity temperature at the anchor event	$\mathrm{MT1}_{\mathrm{peak},i} = \mathrm{MT1}_i[k_{\mathrm{MP1},i}]$	Cavity temperature measured at the time of the maximum cavity-pressure event	Describes the local thermal state inside the cavity at the anchor event
Mould cavity pressure at the anchor event	$\mathrm{MP1}_{\mathrm{peak},i} = \mathrm{MP1}_i[k_{\mathrm{MP1},i}]$	Maximum cavity pressure value within the cycle	Defines the anchor event itself and captures the strongest cavity pressure response in the cycle

A practical issue arises when a cycle does not contain a valid peak event. This can occur in abnormal cycles, such as severe short shots, where cavity pressure may not rise to a clear peak, or in cycles with sensor dropouts, where the signal is incomplete. These cases must be handled explicitly because they affect both the integrity of the dataset and the validity of the modelling comparison. In this study, cycles were checked for peak validity, and any cycle that did not contain a detectable peak meeting the required criteria was excluded from the peak-anchored modelling dataset. This rule was applied consistently across all cycles to preserve a valid and physically meaningful event anchored representation. This process of peak detection and synchronised sampling is visualised in Figure 22.

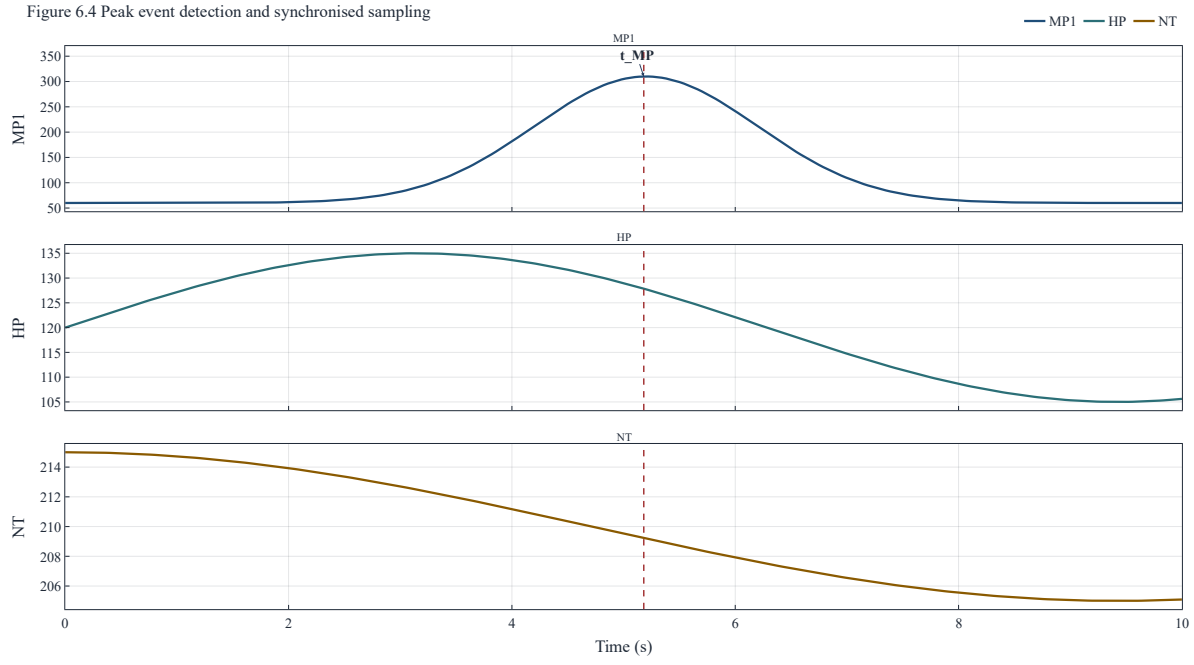


Figure 22. Peak event detection and synchronised sampling at the MP1 anchor.

This event anchored representation provides a compact and physically grounded way to describe each cycle. It reduces the amount of data required per cycle while preserving a clear connection to process physics. In the next subsection, the full curve representation is defined, and the study design is then able to compare whether the compact peak anchored snapshot can achieve classification performance close to that of models trained on the complete multivariate cycle curves.

6.4.3 Full curve representation and resampling

The full curve representation preserves the complete multivariate time series content of each cycle. Instead of describing a cycle using a compact snapshot at one event, this route uses the evolution of the signals over the whole cycle as the model input. The motivation is that some defect mechanisms may be expressed as changes in curve shape, timing, rise and decay behaviour, or stage-specific instability that cannot be captured fully by a single event-anchored sample. The full curve route, therefore, provides a reference representation that retains the maximum available information from the recorded signals.

The first step in constructing the full curve input is cycle boundary alignment. Each cycle is segmented using the cycle start and cycle end markers described in Section 6.3. Once the boundaries are identified, the raw samples for each channel are extracted within the same cycle window. This ensures that the multivariate signals remain synchronised. Boundary alignment is also important for interpretation, because it means that the beginning of the curve corresponds to the same cycle stage across cycles, and the end of the curve corresponds to the completion of the cycle. Although the physical time duration of cycles may differ, the boundary-based segmentation ensures that each curve represents the same sequence of manufacturing stages.

A practical challenge is that cycles often have variable length, meaning that different cycles contain different numbers of time samples. Many machine learning models, particularly deep sequence models, require fixed-size inputs. For this reason, the curves are resampled to a fixed length L = 970-time stamps. Resampling maps each channel from its original number of samples to L samples, producing a standardised representation. This allows direct comparison between cycles, enables batching during training, and supports the use of neural network models that expect a consistent input shape.

After resampling, each cycle is represented as a multivariate sequence containing a fixed number of time steps and process channels. The full-curve input for cycle $i$is defined as:

$$\mathbf{X}_i^{(f)} \in \mathbb{R}^{L \times C}$$

Equation 7. Full-curve input representation

where $\boldsymbol{X}_i^{(f)}$is the full-curve input matrix for cycle i, $L = 970$is the fixed number of resampled time steps, and $C = 6$is the number of monitored process channels. Therefore, each cycle is represented by an $L \times C$matrix. Depending on the model implementation, the matrix may be transposed and represented in the $C \times L$form.

In this study, the channels correspond to HP, SP, NT, HWT, MP1, and MT1, so C equals six. The fixed length L is chosen so that it preserves sufficient time resolution to represent the main cycle features while keeping the input size manageable. This fixed-size multivariate representation is essential for sequence models such as one-dimensional convolutional neural networks, bidirectional recurrent networks, and temporal convolutional networks, because these models learn patterns across time steps and require consistent alignment of the sequence length across training samples.

Resampling does not remove the need for careful preprocessing. The numerical scale of each channel must be treated consistently, especially because different signals have different units and typical magnitudes. Many learning models, particularly neural networks, are sensitive to input scaling. For this reason, scaling is applied per channel, meaning that each channel is normalised using statistics computed on the training data only. This is done inside each training fold during cross-validation to prevent information leakage. In other words, the mean and standard deviation used to scale each channel are estimated from the training portion of a fold, and then the same transformation is applied to the corresponding test portion. This ensures that the model is evaluated on test data that has not influenced any part of the preprocessing. For model families that are less sensitive to scaling, such as tree ensembles, scaling may not be required, but the fold-based preprocessing structure is retained for reproducibility and fairness across comparisons.

Figure 23 illustrates the resampling process used to convert a variable-length cycle signal into a fixed-length representation. The original and resampled signals

follow the same overall shape, indicating that resampling standardises the number of time steps while preserving the main temporal pattern of the cycle.

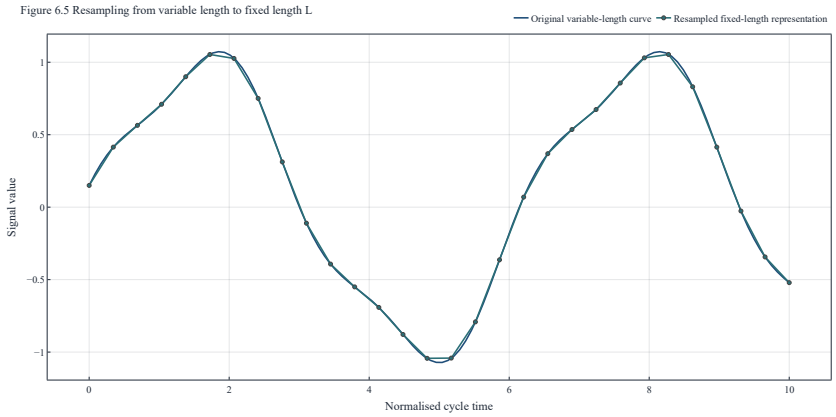


Figure 23. Illustration of resampling from variable-length cycle data to a fixed-length representation of length L.

The outcome of this subsection is a full curve representation where every cycle is described by a fixed-length multivariate sequence. This representation forms the basis of the full curve modelling route. It allows deep sequence models to be trained directly on the multichannel curves and provides a reference benchmark for assessing how much predictive power is gained by using complete cycle information rather than the compact event-anchored snapshot.

6.4.4 Engineering trade-off, information versus cost

The choice between a peak-anchored representation and a full curve representation is not only a modelling decision. It is also an engineering decision because it affects the amount of data that must be stored, processed, and transferred per cycle. In cyclic manufacturing settings, where production can run for long periods and where monitoring is expected to operate continuously, the input representation influences the practical cost of the system. This includes storage requirements, computational load during training and inference, and communication load if data must be transferred between acquisition hardware and a processing computer. For this reason, the two modelling routes defined in this chapter are compared not only by classification performance, but also by input size and data handling burden per cycle.

In the peak anchored route, each cycle is represented by a compact feature vector sampled at the anchor time t_{MP}. In the injection moulding case study, the peak-anchored vector contains one value from each of the monitored channels at the same time. This means that the number of values per cycle is equal to the number of channels used in the representation. If C channels are used, the peak representation contains C values per cycle. In this study, C equals six, corresponding to HP, SP, NT, HWT, MP1, and MT1. The peak-anchored input is therefore compact and constant in size, independent of cycle duration and sampling rate.

In the full curve route, each cycle is represented by the complete multivariate curves after alignment and resampling to a fixed length L. The number of values per cycle is therefore proportional to both the number of channels and the number of time samples. If the cycle is represented as C channels and L resampled time points per channel, the full curve representation contains C × L values per cycle. Even after resampling, this can be two orders of magnitude larger than the peak vector, depending on the choice of L. This increase in input size affects storage and processing. For example, it increases the memory footprint during training, increases computation per forward pass in deep models, and increases the time required to generate predictions, especially when a model must process a full sequence rather than a small vector.

The difference in input size can be formally defined by comparing the number of values required per cycle for each route. Let C denote the number of process channels and L the fixed number of resampled time steps. The numbers of input values required per cycle by the event-anchored and full-curve representations are denoted by N_{E} and N_{F}, respectively. Their input sizes and dimensionality ratio are defined as:

$$N_{\mathrm{E}} = C, \qquad N_{\mathrm{F}} = CL,$$

The ratio of input dimensionality between the two representations is therefore:

$$\frac{N_{\mathrm{F}}}{N_{\mathrm{E}}} = \frac{CL}{C} = L$$

Equation 8. Input sizes and dimensionality ratio of the cycle representations

In this study, $C = 6$ and $L = 970$. Therefore, the event-anchored representation contains $N_{\mathrm{E}} = 6$ input values per cycle, whereas the full-curve representation contains $N_{\mathrm{F}} = 6 \times 970 = 5{,}820$ input values per cycle. Consequently, the full-

curve representation contains 970 times as many input values as the event-anchored representation. This simple comparison highlights the main engineering trade-off. The full-curve route retains more information because it preserves the time evolution of the signals across the whole cycle. This can be beneficial if defect mechanisms depend on curve shape, timing, or stage-specific behaviour that is not captured at a single event. However, the increased input size increases the cost of data handling and model computation. By contrast, the peak-anchored route reduces the cycle to a physically meaningful event-aligned snapshot, which significantly reduces input size and can make the approach more efficient for long production runs. This trade-off is directly relevant to practical deployment. A compact representation reduces the amount of data that must be transferred per cycle if processing is performed away from the machine. It also reduces latency because feature extraction and inference can be completed quickly after the anchor event is detected. In addition, a compact representation simplifies model management because it uses a small number of interpretable inputs that can be monitored and validated more easily. These engineering motivations do not replace the need for accuracy and defect detection performance. Instead, they provide a reason to test whether the compact event anchored representation can approach the predictive performance of the full curve representation. If the performance difference is small, the compact approach may offer a favourable balance between information content and operational cost. This supports its later use within the governed real-time monitoring architecture discussed in Chapter 7, where continuous cycle-level processing and timely decision support are important.

6.5 Models considered and why they fit the study

This section explains the machine learning models evaluated in this chapter and justifies why each model family is relevant to the study aims. The selected model families reflect approaches previously applied to injection-moulding quality prediction, including conventional supervised classifiers, tree-based ensembles, and neural-network methods. Punyangarm and Chotayakul (2025) evaluated Random Forest, XGBoost, CatBoost, and deep-learning models for defect classification, while Ding et al. (2025) demonstrated sequence-based feature extraction for predicting injection-moulded part weight from sensor time-series data. The goal is not only to report which model achieved the highest score, but

to provide engineering evidence about what types of learners are suitable for cycle-level quality prediction under industrial variability. Since the chapter compares two input representations, a compact peak-anchored feature vector and a full multivariate cycle curve, the model set is split by route. This prevents confusion and ensures that each model is matched to the type of input it is designed to handle.

6.5.1 Models for peak anchored features

The peak anchored route represents each cycle as a compact feature vector sampled at the anchor time t_{MP}, with one value per channel. This representation produces a small tabular dataset where each row corresponds to one cycle, and each column corresponds to a physical process variable sampled at the same event. Tabular learning is well-suited to this format, and it supports model families that are widely used in manufacturing analytics because they can handle nonlinearities, variable interactions, and noisy measurements.

The model families selected for the peak route were chosen to cover a spectrum from simple and interpretable baselines to strong nonlinear learners. This is important because it allows the chapter to answer several technical questions. It tests whether the peak anchored features contain enough separable information for even linear boundaries to perform reasonably, it tests whether nonlinear interaction modelling is required, and it examines whether flexible models provide meaningful gains or mainly increase complexity.

Logistic regression is used as the primary linear baseline. It is included because it provides a clear reference for what can be achieved using a linear decision boundary in the peak feature space. In engineering terms, logistic regression tests whether defect separation is approximately linear in the selected variables at t_{MP}. It also provides interpretability through coefficient signs and magnitudes, which can support discussion of whether the feature directions align with process expectations. While logistic regression is unlikely to capture complex interactions, it establishes a baseline level of performance that more advanced models should exceed if a nonlinear structure is present.

Naïve Bayes is included as a probabilistic baseline. It assumes conditional independence of features given the class label. This assumption is not expected to hold strictly in injection moulding because process variables are coupled, for example, pressure and position behaviours are linked through machine control

and melt response. However, Naïve Bayes is still useful as a benchmark because it is simple, fast, and provides a clear lower bound reference. If Naïve Bayes performs poorly, it provides evidence that feature interactions and dependencies are important. If it performs reasonably well, it suggests that the peak features contain strong marginal separation signals even without modelling interactions.

Support vector machines are included because they can learn nonlinear boundaries while remaining effective on moderately sized datasets. In the peak feature route, the dataset is low-dimensional, but defect separation may still be nonlinear due to interactions between pressure, position, and temperature. Support vector machines can use kernel functions to represent complex decision surfaces, and they often provide strong performance when the separation margin is meaningful, and the dataset size is not extremely large. From a manufacturing perspective, the SVM route tests whether a smooth nonlinear boundary provides an advantage over linear models without requiring large model capacity.

Tree-based ensemble models are included because they are among the strongest learners for tabular industrial data. Decision trees can model nonlinear relationships and interactions by splitting the feature space into regions, but a single tree can be unstable and sensitive to noise. Ensembles address this by combining many trees to improve generalisation. Random forests use bagging and random feature selection to reduce variance and improve robustness. Gradient boosting methods build trees sequentially, where each tree focuses on correcting the errors of the previous ones, often producing strong performance on structured tabular datasets. These models are relevant to injection moulding data because the relationship between process variables and quality can include threshold effects, interactions, and regime changes, for example, a combination of high pressure and specific thermal conditions leading to flash risk. Tree ensembles can represent these patterns naturally without requiring explicit feature engineering beyond the peak sampling.

An optimised gradient boosting implementation is included because it provides strong performance and efficient training, and it is commonly used in industrial predictive maintenance and quality classification tasks. It can handle nonlinearities and interactions, it supports class weighting, and it often performs well when the dataset includes noise and imbalance. Including this model provides a strong reference point for what can be achieved in the peak route using modern boosting methods.

Artificial neural networks are included as flexible nonlinear approximators. In the peak route, the input dimension is small, so the purpose of an ANN is not to exploit temporal patterns, but to test whether a smooth nonlinear function approximator can extract additional separability from the compact features beyond what is captured by trees and kernel methods. Neural networks can learn complex nonlinear mappings if sufficient data are available and if training is stable. They are included here to provide a representative deep learning baseline in the tabular feature space, and to test whether their flexibility translates into improved defect detection under this compact representation.

The outcome of this model set is a structured comparison within the peak-anchored feature space. It provides evidence about whether peak anchored features are sufficient for accurate classification, and which type of learner is most suitable for extracting predictive value from them under realistic variability. The models evaluated in the peak-anchored route are summarised in Table 17, including their model family, purpose in this study, expected strength, and main limitation.

Table 17. Peak-anchored route models and their role.

Model	Family	What it tests in this study	Strength in the peak-based route	Limitation
Logistic Regression	Linear statistical classifier	Tests whether the peak-anchored feature space can be separated using a simple linear decision boundary	Simple, interpretable, fast to train, and useful as a baseline reference	Cannot capture strong nonlinear interactions or complex decision boundaries
Naïve Bayes	Probabilistic classifier	Tests whether the peak features contain enough class separation under a simple independence-based probabilistic assumption	Very fast, simple, and useful as a weak baseline for comparison	Assumes conditional independence between features, which is rarely valid in coupled manufacturing signals
SVM	Margin-based nonlinear classifier	Tests whether a nonlinear separating boundary improves classification over linear methods in the compact peak feature space	Effective on moderate-sized datasets and able to model nonlinear separation through kernel functions	Sensitive to parameter choice, less interpretable, and may be less stable under noisy overlap
Random Forest	Tree-based ensemble	Tests whether bagged decision trees can capture nonlinear relationships and variable interactions in the peak anchored features	Robust to noise, handles interactions well, and performs strongly on industrial tabular data.	May be less sharply optimised than boosting methods for difficult boundary refinement

Model	Family	What it tests in this study	Strength in the peak-based route	Limitation
Gradient Boosting	Tree-based boosting ensemble	Tests whether sequential boosting improves defect separation in the peak feature space	Strong performance on structured tabular data and effective at modelling nonlinear threshold behaviour	Can overfit if not controlled carefully, and requires tuning of boosting settings.
XGBoost	Optimised boosted tree ensemble.	Tests whether a regularised and efficient boosting method provides the strongest discrimination on peak-anchored features	Very strong performance, efficient training, and effective for nonlinear and imbalanced classification problems	More complex to tune and less directly interpretable than simpler baseline models
ANN	Feedforward neural network	Tests whether a flexible nonlinear function approximator can improve separation in the compact peak feature space	Can learn nonlinear combinations of the peak features without manual interaction design	May not outperform strong tree ensembles on moderately sized tabular data and is less interpretable.

Table 17 shows that the peak-based route includes a progression from simple interpretable baselines to stronger nonlinear learners, allowing the study to test whether the compact event-anchored feature space requires linear, nonlinear, or interaction-driven modelling.

6.5.2 Models for full curve inputs

The full curve route represents each cycle using the complete multivariate signal evolution over the cycle after boundary alignment and resampling to a fixed length L. This produces a structured multichannel sequence input for each cycle. The motivation for using full curves is that defect mechanisms may not be expressed only at one instant. Some defects can be linked to how signals rise, stabilise, oscillate, or decay, and to how these behaviours evolve across phases such as filling, packing, and cooling. Full curve inputs, therefore, preserve information about curve shape and timing that is not available in a compact snapshot.

Models for the full curve route are divided into two subgroups. The first subgroup comprises sequence models that consume the multichannel tensor directly and learn temporal patterns. The second subgroup includes strong tabular learners that are trained on a vectorised version of the curves. This split is important because it separates the effect of having full curve information from the effect of using a specialised sequence learning architecture. It also provides a fair

engineering comparison, because in manufacturing practice, it is common for strong classical models to remain competitive when full signals are flattened into fixed-length feature vectors.

The first subgroup consists of deep sequence models designed to learn temporal patterns. A one-dimensional convolutional neural network is included because it learns local patterns through convolutional filters and pooling. In manufacturing curve terms, a CNN can detect characteristic shapes such as sharp rises, plateaus, inflexion points, and repeated local features, and it can learn them at different positions along the cycle. This is relevant because many process signals contain phase-specific patterns, for example, a pressure rise during filling followed by a stabilisation region during packing. A CNN can learn these patterns without requiring explicit manual feature engineering, and it can combine information across channels through multi-filter learning.

A bidirectional long short-term memory network is included to model recurrent sequences. In manufacturing curve terms, a BiLSTM learns dependencies across the sequence by maintaining a state that can capture how early behaviour influences later behaviour. The bidirectional structure processes the sequence in both forward and reverse directions, allowing the model to use information from both earlier and later portions of the cycle when forming its internal representation. This is useful for cycle classification because the label is determined after the complete cycle is observed, meaning that both early and late cycle behaviours may contain relevant information. For example, a defect may be associated with an early instability that influences later packing behaviour, or it may appear as a late deviation in temperature or pressure decay.

A temporal convolutional network is included because it provides another sequence modelling approach based on causal or dilated convolutions. In manufacturing curve terms, a TCN can capture patterns over longer time horizons by expanding the receptive field through dilation. This allows it to model relationships across wider parts of the cycle while still using convolutional operations that are often stable and efficient. A TCN can therefore represent both local phase features and longer-range dependencies without relying on recurrent state updates. Including both BiLSTM and TCN allows the study to test whether recurrent memory-based modelling or convolutional long-range modelling better captures the structure of injection moulding cycles.

The second subgroup consists of tree-based ensemble learners applied to flattened full curves. In this approach, the resampled multichannel sequence is vectorised into a fixed-length feature vector. If there are C channels and L time points, the flattened input contains C × L values, and each of these values is treated as a feature. Gradient boosting and boosted tree models are then trained on this vector. This approach is included for two reasons. First, it provides a strong baseline that often performs competitively on industrial datasets. Tree ensembles can exploit nonlinear interactions and can handle many input features, particularly when the dataset contains clear discriminative regions. Second, it tests whether the main benefit of using full curves comes from the additional information content itself rather than from the specific deep architecture. If flattened boosted trees perform similarly to deep sequence models, this suggests that the discriminative information can be captured without complex temporal modelling. If deep sequence models perform better, this suggests that learning temporal structure provides additional value beyond simply having more data points.

Flattened curve models also have practical relevance because they can be easier to train and tune than deep sequence models in some manufacturing contexts. They often require fewer decisions about network design and can perform well with moderate dataset sizes. However, flattening removes explicit temporal structure, meaning that the model does not inherently understand that adjacent time points are related. Any temporal pattern must be learned indirectly through feature interactions. Including this subgroup, therefore, provides a balanced comparison between sequence-aware learning and strong tabular learning on the same full curve information. The models used in the full curve route are summarised in Table 18, which distinguishes between direct sequence models and ensemble learners trained on flattened curve inputs.

Table 18. Models evaluated for the full-curve route and their roles.

Model	Family	Input form	What it tests in this study	Strength in the full curve route	Limitation
CNN-1D	Convolutional neural network for sequences	Direct multichannel sequence input	Tests whether local temporal patterns and curve shape features can be learned directly from the resampled cycle signals	Effective at learning local shape patterns such as rises, plateaus, and local transitions across the cycle	May miss longer range temporal dependencies if the receptive field is limited and can require careful architecture tuning
BiLSTM	Bidirectional recurrent neural network	Direct multichannel sequence input	Tests whether sequence memory and dependence across early and late parts of the cycle improve classification	Able to capture temporal dependence across the cycle in both forward and backward directions	More computationally expensive, slower to train, and may need larger datasets for a stable advantage
TCN	Temporal convolutional network	Direct multichannel sequence input	Tests whether dilated temporal convolutions can capture longer-range temporal structure across the cycle	Can model both local and broader temporal patterns while remaining convolution-based	Performance depends strongly on architecture design and may still underperform robust ensemble baselines in moderately sized industrial datasets.
Gradient Boosting	Tree-based boosting ensemble	Flattened full curve vector	Tests whether the full curve information itself is sufficient to improve classification when treated as a high-dimensional tabular input	Strong performance on structured high-dimensional data and is effective at capturing nonlinear interactions between time-indexed inputs.	Does not model temporal order explicitly and may rely on indirect interaction learning.
XGBoost	Optimised boosted tree ensemble	Flattened full curve vector	Tests whether a regularised and efficient boosted tree model provides the strongest full curve discrimination without explicit sequence learning	Very strong performance, efficient training, and highly effective on high-dimensional structured inputs	More complex to tune and does not explicitly preserve sequence structure
ANN	Feedforward neural network	Flattened full curve vector	Tests whether a flexible nonlinear learner can exploit the additional information in the flattened full curve representation	Can model nonlinear combinations across the full curve input space	Does not explicitly encode temporal structure and may be less robust than boosted tree models on moderately sized industrial datasets

Model selection was based on comparative predictive performance under the common validation protocol described in Section 6.6 rather than on model complexity alone. Within each representation route, the candidate models were compared using ROC-AUC and PR-AUC as the primary threshold-independent measures of discrimination, supported by Accuracy, Precision, Recall and F1 at the selected operating threshold. Threshold-independent metrics were given particular importance because they allow model quality to be compared without depending on one arbitrary classification threshold. Where models produced similar results, preference was given to the model providing the strongest and most consistent discrimination across the reported metrics. This procedure resulted in XGBoost being used as the representative model for the direct comparison between the event-anchored and full-curve routes because it achieved the strongest ROC-AUC and PR-AUC in both routes.

6.6 Training and validation protocol

This section describes the training and validation protocol used to evaluate all models in a fair and reproducible manner. The main purpose is to compare the candidate models and cycle representations on held-out cycles under a common validation protocol while preventing information leakage through preprocessing. Because the observations originate from sequential production cycles, the resulting performance estimates represent generalisation to held-out cycles drawn from the operating conditions represented in the historical dataset rather than prospective generalisation to later production regimes. Since the dataset is moderately sized and imbalanced, the evaluation design is especially important, and it must be consistent across both modelling routes, the peak anchored route and the full curve route.

The evaluation was performed at the cycle level using a stratified five-fold cross-validation. In this approach, the full dataset is divided into five folds, and each fold contains approximately the same proportion of good and defective cycles as the full dataset. Stratification is used to preserve the good and defective class proportions within each fold. Without stratification, one-fold may contain an unrepresentative number of defective cycles, producing unstable estimates of recall, precision, and F1 score. Five-fold cross-validation provides a practical balance between training data size and evaluation stability. Each model is trained five times. In each run, four folds are used for training, and one-fold is held out for

testing. This is repeated until every fold has served as the test fold once. The reported performance, therefore, reflects behaviour across all cycles, not only on a single held-out subset.

Although each test-cycle prediction is generated by a model that was not trained on that cycle, stratified random fold assignment does not preserve production order. Successive injection moulding cycles may share persistent thermal, mechanical or operating conditions, and cycles originating from the same local production regime can therefore occur in both the training and test folds. The resulting train-test separation is valid at the individual-cycle level but does not guarantee temporal independence between the two sets. Consequently, the reported cross-validation performance may be more favourable than performance obtained when predicting a later production period affected by process drift or an operating regime not represented during training. The five-fold results are therefore interpreted as comparative evidence of model discrimination across held-out cycles from the observed production conditions, rather than as a direct estimate of prospective performance under future temporal regimes. A temporally blocked or forward-ordered validation design would be required to evaluate that latter form of generalisation.

A key feature of this protocol is the use of out-of-fold predictions. For each cycle, a probability prediction is generated only when that cycle is held out as part of the test fold, meaning that the model producing the prediction has not been trained on that cycle. The resulting probabilities can then be aggregated across the five folds to form a single consolidated set of predictions covering the full dataset. This approach has two benefits. First, it provides an out-of-fold probability estimate for every cycle, with each prediction generated by a model that was not fitted to that cycle. Second, it allows ROC curves and precision-recall curves to be computed from one unified set of predictions rather than averaging curves across folds, which can be difficult to interpret. It also allows a single operating threshold to be selected from the aggregated predictions and then used to generate one confusion matrix that reflects performance across the full dataset.

For deep learning models, training is performed within each training fold using an internal validation split. In this case, a portion of the training fold is set aside as a validation set. This validation set is used for early stopping and for selecting the best model checkpoint during training, based on validation performance. This reduces overfitting and stabilises training, particularly when the dataset is

limited in size. The test fold remains fully unseen throughout training and is used only for final evaluation in that cross-validation run.

Reproducibility is treated as a requirement because a thesis must support independent verification. For this reason, the cross-validation splits are created with a fixed random seed, and the fold assignment is kept consistent across models. Any stochastic training elements, such as data shuffling and model initialisation, also use fixed seeds where possible. This ensures that differences in performance across models are driven by modelling choices rather than random variation in splitting or training.

Preprocessing steps are performed inside each fold using the training data only, and then applied to the corresponding test data. This avoids information leakage. For example, when feature scaling is required, the scaling parameters are computed from the training portion of a fold and then applied to both training and test data within that fold. The same principle applies to any preprocessing of full curve inputs. Channel-wise scaling is applied using training fold statistics only, and test fold samples are transformed using the same parameters. This ensures that no information from the test fold influences the preprocessing stage.

Although the final 903-cycle dataset used in this chapter is near-balanced, class weighting or balanced sample weighting was retained where supported as a conservative safeguard against fold-level imbalance and to maintain a consistent evaluation protocol across model families. This is important because defective cycles are the class of greatest quality-control concern. Where class weighting is not directly supported, equivalent strategies are used, such as balanced sample weighting at training time.

The performance of each model is evaluated using confusion matrix-based metrics and curve-based metrics. Confusion matrix metrics measure performance at a chosen operating threshold. Curve-based metrics evaluate performance across all possible thresholds, which is useful for comparing models without fixing a single threshold early in the analysis. The following definitions are used throughout this chapter:

- True Positives (TP): Defective cycles are correctly predicted as defective.
- True Negatives (TN): Good cycles are correctly predicted as good.
- False Positives (FP): Good cycles are incorrectly predicted as defective.
- False Negatives (FN): Defective cycles are incorrectly predicted as good.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
$$F_1 = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Equation 9. Standard classification metrics

In manufacturing, precision and recall are often more informative than accuracy. Precision indicates how many predicted defects are truly defective, which relates to false alarms and unnecessary intervention. Recall indicates how many true defects are detected, which relates directly to escaped defects, the most critical failure mode in many quality systems. The F_1 score provides a balanced summary of precision and recall and is useful as a single operating objective when both types of error matter.

To compare models across thresholds, ROC and precision recall curves are computed from the aggregated out-of-fold probabilities. The ROC curve plots the true positive rate against the false positive rate as the decision threshold varies. The true positive rate is equivalent to recall. The false positive rate quantifies how often good cycles are incorrectly flagged as defective.

The true-positive rate and false-positive rate used to construct the ROC curve are defined as:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

Equation 10. Rates used to construct the ROC curve

where TPR is the true-positive rate, which is equivalent to recall, and FPR is the proportion of good cycles incorrectly classified as defective.

The ROC area under the curve, ROC AUC, summarises the ability of a model to rank defective cycles above good cycles across all thresholds. However, ROC AUC alone does not fully describe the precision–recall trade-off for the defective class, especially when the operating objective is to detect defects while controlling false alarms. For this reason, precision-recall curves are also reported. A precision

recall curve directly reflects the trade-off between precision and recall for the defective class as the threshold changes. The precision, recall, and area under the curve, PR AUC, provides an informative summary because it focuses on the precision–recall trade-off for the defective class and is sensitive to false positives in a way that matters in quality control.

For the event-anchored route, operating thresholds and confusion matrices are derived from aggregated out-of-fold probability predictions. Candidate thresholds are evaluated using those predictions, and the threshold maximising F1 is selected to balance precision and recall. The selected threshold is then applied to the same out-of-fold predictions to produce the corresponding confusion matrix and threshold-dependent metrics. For the full-curve route, classifier decisions from the held-out folds are retained without an additional F1-based threshold search. The direct comparison between representations remains based primarily on threshold-independent ROC-AUC and PR-AUC values. The false-positive and false-negative counts were also retained for later operational interpretation. Although class weighting or balanced sample weighting was retained during training as a conservative safeguard against fold-level imbalance, this is different from the later cost-sensitive decision analysis. Class weighting helps reduce bias toward one class during model fitting, whereas the latter decision-cost analysis compares the operational consequences of errors made after prediction. In this context, a false positive represents a good cycle being incorrectly flagged as defective, while a false negative represents a defective cycle being incorrectly accepted. Retaining these counts allows the later evaluation to compare decision outputs not only by statistical performance, but also by their practical error consequences.

This training and validation protocol produces a consistent set of out-of-fold probability predictions for every model under both representations. In the direct route comparison presented later, the primary comparison is based on threshold-independent ranking metrics, particularly ROC AUC and PR AUC, because these provide the fairest common basis for comparing the peak-anchored and full curve routes. The next sections use these predictions to report the results first for the peak anchored route and then for the full curve route, including performance tables, ROC and precision recall curves, and confusion matrices derived from the selected operating thresholds.

The offline modelling workflow was separated into four methodological stages. First, each candidate model was trained using the implementation settings

defined for that model; the reported five-fold evaluation did not include a separate nested hyperparameter-optimisation stage. Second, the candidate model families were compared under the same stratified five-fold cross-validation framework. ROC-AUC and PR-AUC provided the principal threshold-independent basis for comparing discrimination between models, while Accuracy, Precision, Recall and F1 provided complementary operating-point information. The representative models used in the subsequent route comparison were identified from these comparative results. Third, operating-threshold handling was performed after probability generation according to the implementation used for each representation route. For the event-anchored route, the held-out probabilities from the five folds were aggregated and an operating threshold was selected by maximising F1. For the full-curve route, the classifier decisions generated within the corresponding held-out folds were retained without an additional F1-based threshold search. Finally, the performance values reported in this chapter represent the cross-validated offline evaluation obtained from these held-out fold predictions. No additional post-selection hold-out dataset was introduced as part of the original Chapter 6 evaluation. This separation clarifies the distinction between model configuration, comparative model selection, operating-threshold selection and final performance reporting.

Table 19 shows that all models were evaluated under the same core validation structure, while allowing appropriate differences in preprocessing and training control according to the requirements of each model family.

Table 19. Training and validation settings by model family.

Model family	Preprocessing	Class imbalance handling	Validation method	Early stopping	Reproducibility setting	Main training setting
Linear and probabilistic models, Logistic Regression and Naïve Bayes	Fold-wise feature scaling, where required, was fitted on the training fold only and applied to the corresponding test fold.	Class weighting or equivalent balanced sample weighting, where supported	Stratified five-fold cross-validation at the cycle level with aggregated out-of-fold probability predictions	Not used	Fixed random seed for fold generation, consistent fold assignment across models	Trained on peak-anchored tabular features within each fold, operating threshold selected from aggregated out-of-fold predictions
SVM	Fold-wise feature scaling is required, fitted on the training fold only and applied to the corresponding test fold.	Class weighting / balanced weighting	Stratified five-fold cross-validation at the cycle level with aggregated out-of-fold probability predictions	Not used	Fixed random seed for fold generation, consistent fold assignment across models	Trained on peak-anchored tabular features within each fold, evaluated by out-of-fold probabilities and threshold-based confusion matrix
Tree ensembles, Random Forest, Gradient Boosting, XGBoost	No mandatory scaling for tabular inputs; preprocessing still restricted to the training fold, for full-curve route, flattened inputs prepared within-fold only	Class weighting or equivalent balanced sample weighting, where supported	Stratified five-fold cross-validation at the cycle level with aggregated out-of-fold probability predictions	Not used	Fixed random seed for fold generation, consistent fold assignment across models	rained on peak-anchored tabular features or flattened full-curve inputs, depending on route; evaluated from aggregated out-of-fold predictions
ANN on tabular or flattened vector inputs	Fold-wise scaling of input features using training-fold statistics only	Class weighting / balanced weighting during training	Stratified five-fold cross-validation with an internal validation split inside each training fold	Yes, based on validation performance and best checkpoint selection	Fixed seeds for fold generation, data shuffling, and model initialisation where possible	Trained within each fold using a training/validation split, evaluated only on the unseen test fold
Deep sequence models, CNN1D, BiLSTM, TCN	Channel-wise scaling of full-curve sequence inputs using training-fold statistics only	Class weighting / balanced weighting during training	Stratified five-fold cross-validation with an internal validation split inside each training fold	Yes, based on validation performance and checkpoint selection	Fixed seeds for fold generation, data shuffling, and model initialisation where possible	Trained on direct multichannel sequence inputs within each fold; validation was used for training control, with final evaluation on the unseen test fold

6.7 Results for peak anchored models

This section presents the classification results obtained from the event anchored peak-based route, where each cycle is represented by the synchronised process snapshot sampled at the anchor time

t_{MP}. The purpose of this analysis is to determine how much predictive information is retained when the cycle is reduced to a compact feature vector centred on the maximum cavity-pressure event, and to identify which model families are most effective for learning from this representation. Since the peak anchored route is the compact and physically interpretable route in this study, its performance is central to the argument of the chapter. If this route performs close to the full curve route, then the maximum cavity-pressure event can be justified not only as a physically meaningful event, but also as a practically useful basis for cycle-level quality classification.

All results in this section are based on the stratified five-fold cross-validation protocol described in Section 6.6. The reported ranking metrics were derived from aggregated out-of-fold probability predictions, so that each cycle was evaluated only when it was part of the held-out fold. This produces a fair estimate of generalisation performance and allows consistent generation of ROC and precision recall curves across the full dataset. The threshold-based confusion matrices shown later in this section provide an operational view of classification performance and illustrate how the best peak anchored models balance false alarms against missed defects. The performance of all peak anchored models is summarised in Table 20.

Table 20. Peak anchored model performance summary

Model	Accuracy	Precision	Recall	F1	ROC AUC	PR AUC
Random Forest	0.958	0.976	0.935	0.955	0.989	0.99
XGBoost	0.963	0.981	0.942	0.961	0.991	0.991
Gradient Boosting	0.966	0.974	0.953	0.963	0.989	0.988
SVM	0.831	0.803	0.756	0.779	0.858	0.857
ANN	0.82	0.786	0.734	0.759	0.833	0.804
Logistic Regression	0.803	0.627	0.902	0.74	0.851	0.846
Naïve Bayes	0.7	0.373	0.515	0.432	0.61	0.662

The peak anchored results show a clear separation between the stronger nonlinear ensemble models and the simpler baselines. Among all peak anchored models, the best overall threshold-based performance was achieved by Gradient Boosting, which reached an Accuracy of 0.966, a precision of 0.974, a recall of 0.953, and an F1 score of 0.963. XGBoost produced almost identical classification quality, with an Accuracy of 0.963, a precision of 0.981, a recall of 0.942, and an F1 score of 0.961. In terms of threshold independent ranking performance, XGBoost achieved the strongest result, with ROC AUC of 0.991 and PR AUC of 0.991. Random Forest also performed strongly, with an Accuracy of 0.958, a precision of 0.976, a recall of 0.935, an F1 score of 0.955, ROC AUC of 0.989, and PR AUC of 0.990. Taken together, these three models confirm that the peak anchored feature set contains strong discriminative information and that tree-based ensembles are able to exploit it effectively.

By contrast, the weaker baseline models lagged. SVM improved substantially over the simplest baselines, but remained well below the ensemble models, with an F1 score of 0.779, ROC AUC of 0.858, and PR AUC of 0.857. The ANN produced a similar pattern, with an F1 score of 0.759, ROC AUC of 0.833, and PR AUC of 0.804, indicating that a flexible nonlinear function approximator did not outperform the tree ensembles in this compact tabular setting. Logistic Regression, which acts as the linear baseline, achieved high recall at 0.902 but much lower precision at 0.627, resulting in an F1 score of 0.740. This suggests that the peak-anchored feature space is not well described by a simple linear boundary and that a linear model tends to over-flag defects in order to capture more defective cycles. Naïve Bayes gave the weakest result overall, with an F1 score of 0.432, ROC AUC of 0.610, and PR AUC of 0.662. This is consistent with the fact that the peak features are not conditionally independent and that the manufacturing signals remain strongly coupled even when sampled at a single event.

These results provide two important findings. First, the event anchored peak-based representation is clearly capable of supporting high-quality defect classification. The best peak anchored models exceed 0.96 in F1 score and approach 0.99 in both ROC AUC and PR AUC, which is a strong result for a compact representation derived from a single process event. Second, the ranking of models indicates that the defect boundary in the peak feature space is nonlinear, and interaction driven. This explains why tree-based ensembles outperform the linear and probabilistic baselines. The physical interpretation is also consistent with the process, because defects in injection moulding often arise from combinations of

pressure, position, and thermal conditions rather than from one variable acting alone. The ROC curves for these models are visualised in Figure 24.

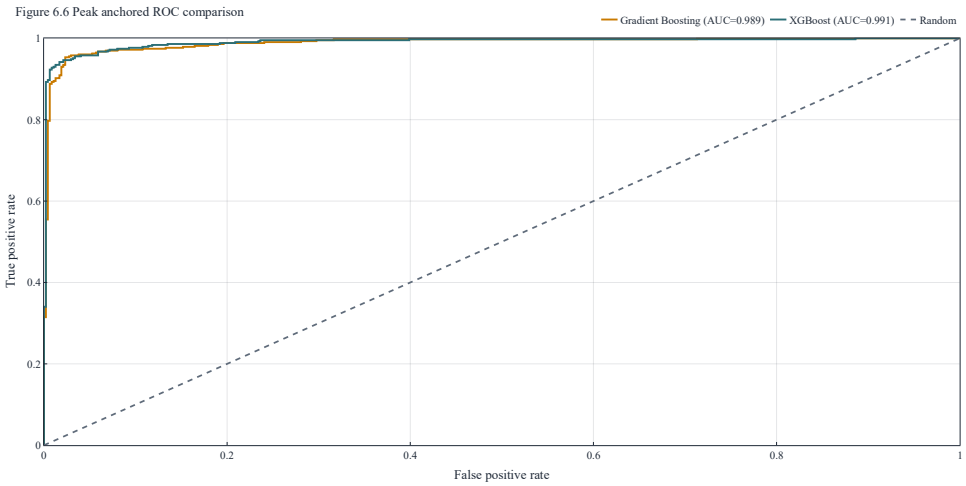


Figure 24. ROC curves for peak anchored models.

The ROC curves confirm the strength of the ensemble models. XGBoost produced the highest ROC AUC at 0.991, while Random Forest and Gradient Boosting both remained very close at 0.989. This indicates that the best peak anchored models rank defective cycles above good cycles with very high consistency across thresholds. The lower ROC AUC values of the baseline models, particularly Logistic Regression and Naïve Bayes, confirm that their class separation is weaker even before threshold selection is applied. The corresponding precision-recall curves are shown in Figure 25.

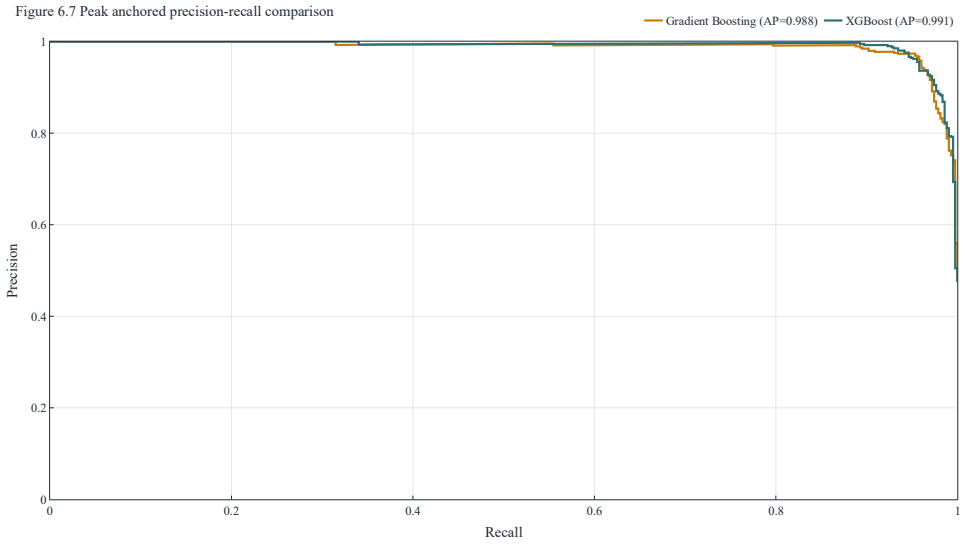


Figure 25. Precision-recall curves for peak anchored models.

The precision-recall curves are especially important in this study because defective parts are the class of greatest industrial concern. In this view, XGBoost again produced the strongest result, with PR AUC of 0.991, followed very closely by Random Forest at 0.990 and Gradient Boosting at 0.988. These values show that the leading peak anchored models maintain high precision while retaining strong recall across a wide range of thresholds. This is a significant result because it demonstrates that the compact peak anchored representation preserves both precision and recall effectively for the defective class, where weaker models often fail to maintain both measures simultaneously.

To provide an operational interpretation, confusion matrices were examined for the two strongest peak-anchored models. These matrices show the actual counts of correctly and incorrectly classified cycles at the selected operating point and therefore give a clearer engineering view of error types than summary metrics alone. The confusion matrices for Gradient Boosting and XGBoost are shown in Figure 26 and Figure 27, respectively.

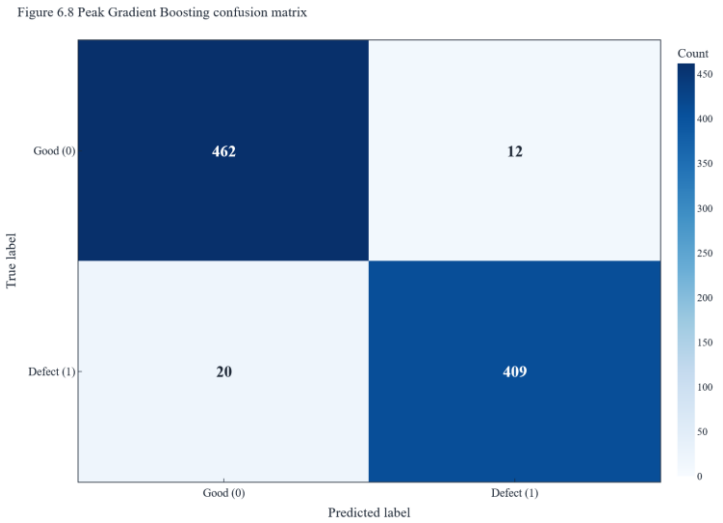


Figure 26. Confusion matrix for Gradient Boosting.

The Gradient Boosting confusion matrix shows a strong balance between defect capture and false alarm control. Only 12 good cycles were incorrectly flagged as defective, while 409 defective cycles were correctly identified. The number of missed defects was 20, which remains low relative to the total number of defective cases in the evaluated set. This pattern is consistent with its leading F1

score and confirms that Gradient Boosting provides a well-balanced operating point for the peak anchored route.

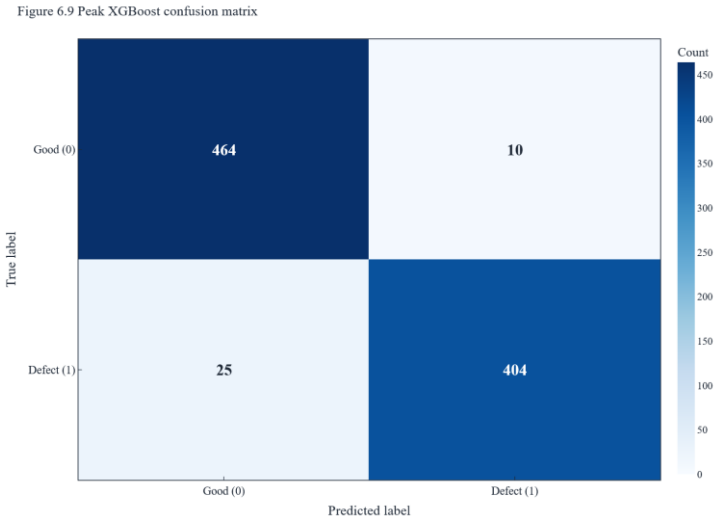


Figure 27. Confusion matrix for XGBoost.

The XGBoost confusion matrix shows an even lower number of false positives than Gradient Boosting, with only 10 good cycles incorrectly flagged, but a slightly higher number of missed defects, with 25 defective cycles incorrectly classified as good. This explains why XGBoost achieved the strongest ranking metrics overall, yet Gradient Boosting remained marginally stronger at the selected threshold used for the final count-based classification view. In engineering terms, XGBoost offers a very strong discriminator across thresholds, while Gradient Boosting gives a slightly better-balanced operating point under the threshold used for the count-based result.

Overall, the peak anchored results provide strong evidence for the central argument of this chapter. A compact event anchored representation, built around the maximum cavity-pressure event and the synchronised process state at that moment, is sufficient to support highly accurate cycle level quality classification. The fact that the best peak anchored models achieve near top performance using only a small number of physically meaningful inputs is an important result because it supports the use of this representation as a practical and scientifically justified route for quality prediction in cyclic manufacturing.

6.8 Results for full curve models

This section presents the classification results obtained from the full curve route, where each cycle is represented by the complete multivariate signal history after cycle boundary alignment and resampling to a fixed length. The purpose of this analysis is to evaluate how much predictive benefit is gained when the model is given access to the full temporal evolution of the cycle rather than a compact event-anchored snapshot. This route acts as the richer reference representation in the chapter, because it preserves the complete signal shape across filling, packing, and the remaining cycle stages.

As described in Section 6.6, all models were evaluated using a stratified five-fold cross-validation at the cycle level. For the full curve route, the reported values represent the overall cross-validation performance of each model. In addition, the leading boosted tree models produced aggregated out-of-fold probability predictions, which were used to generate combined ROC and precision-recall curves and to derive representative confusion matrices. The performance of the full-curve models is summarised in Table 21, which includes the main classification and ranking metrics used to compare the models.

Table 21. Full curve model performance summary

Model	Accuracy	Precision	Recall	F1	ROC AUC	PR AUC
Gradient Boosting	0.959	0.973	0.939	0.956	0.990	0.990
XGBoost	0.959	0.970	0.943	0.956	0.994	0.994
ANN	0.807	0.838	0.770	0.784	0.908	0.902
CNN1D	0.853	0.887	0.810	0.837	0.919	0.926
TCN	0.708	0.659	0.877	0.743	0.851	0.848
BiLSTM	0.854	0.884	0.800	0.837	0.935	0.936

Table 21 shows that the strongest full curve performance is achieved by the boosted tree models trained on flattened full curve inputs, while the deep sequence models provide useful but lower performance in the present dataset. XGBoost and Gradient Boosting both reached an Accuracy of 0.959. Their F1 scores were also equal when rounded to three decimal places, at 0.956 for XGBoost and 0.956 for Gradient Boosting. However, XGBoost produced the strongest threshold independent ranking performance, with ROC AUC of 0.994 and PR AUC of 0.994, while Gradient Boosting followed closely with ROC AUC of

0.990 and PR AUC of 0.990. These results confirm that the full curve representation contains strong discriminative information and that boosted tree ensembles can exploit this information extremely effectively even without an explicitly sequence-aware architecture.

Among the deep sequence models, the best results were obtained by BiLSTM and CNN1D. BiLSTM achieved an Accuracy of 0.854, a precision of 0.884, a recall of 0.800, F1 of 0.837, ROC AUC of 0.935, and PR AUC of 0.936. CNN1D produced a very similar F1 score of 0.837, with Accuracy of 0.853, ROC AUC of 0.919, and PR AUC of 0.926. The TCN performed less strongly in this dataset, with Accuracy of 0.708, Precision of 0.659, Recall of 0.877, an F1 of 0.743, an ROC AUC of 0.851, and a PR AUC of 0.848. Its relatively high recall, combined with lower precision, indicates that it tended to over-flag the defective class, leading to more false positives.

The ANN trained on the flattened full curve input performed better than the weaker baselines in the peak route, but it remained below the boosted tree models. It achieved an accuracy of 0.807, a precision of 0.838, a recall of 0.770, F1 of 0.784, ROC AUC of 0.908, and PR AUC of 0.902. This indicates that while the richer full curve input helps a flexible nonlinear model, the combination of moderate dataset size, industrial variability, and coupled process behaviour still favours more robust structured learners. The ROC curves for these full curve models are visualised in Figure 28.

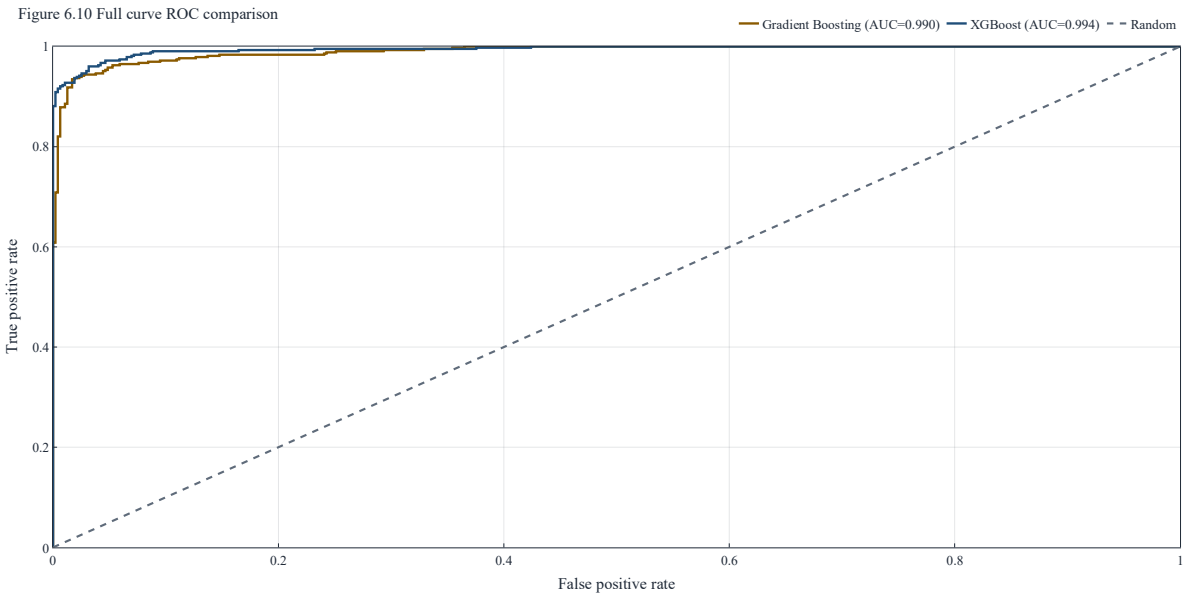


Figure 28. ROC curves for full curve models.

The ROC comparison confirms that the best full curve discrimination is achieved by the boosted tree models. XGBoost gives the highest ROC AUC, followed very closely by Gradient Boosting. BiLSTM and CNN1D form a middle group with clearly improved ranking ability over the weaker models, but they do not reach the same level as the boosted trees. TCN remains the weakest among the full curve deep sequence models in this dataset. This pattern shows that the benefit of the full curve input is real, but it does not automatically mean that sequence models will outperform strong tabular learners. The corresponding precision-recall curves are shown in Figure 29.

Figure 6.11 Full curve precision-recall comparison

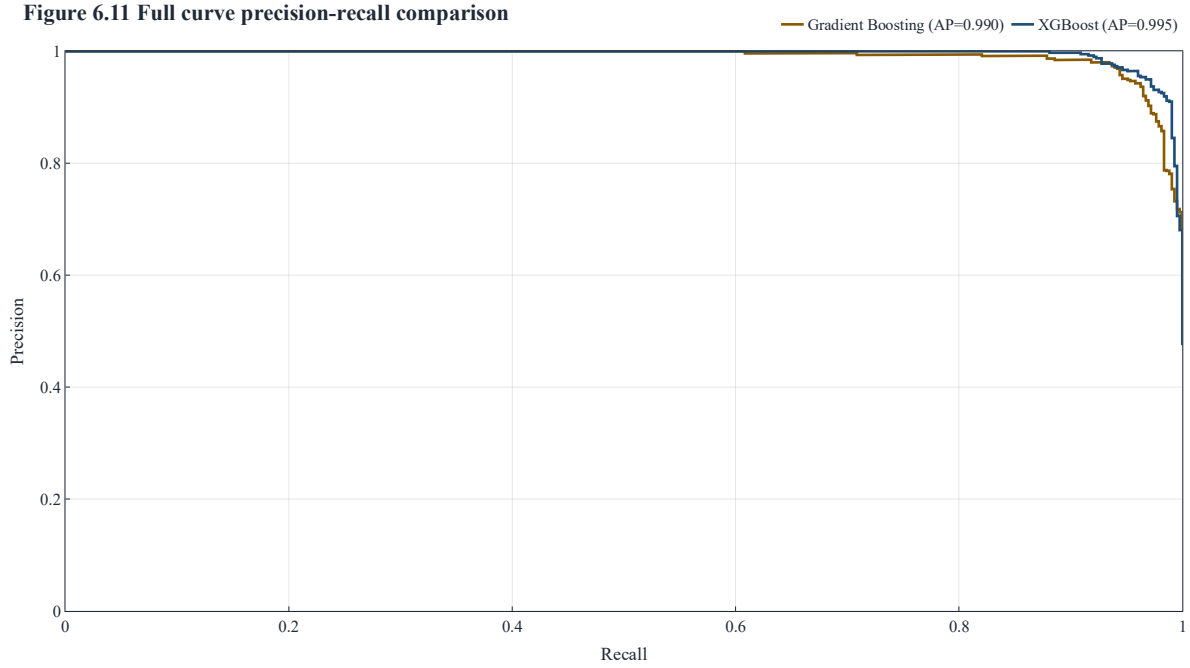


Figure 29. Precision recall curves for full curve models.

The precision-recall curves reinforce the same conclusion for the defective class. XGBoost produced the strongest PR AUC at 0.994, with Gradient Boosting very close at 0.990. BiLSTM and CNN1D show useful defect detection ability, but their precision-recall behaviour remains meaningfully below that of the top two models. Since the defective class is the class of greatest industrial concern, this result is particularly important. It shows that the boosted tree models preserve precision and recall more effectively across thresholds than the deep sequence models in the present dataset.

To provide an operational interpretation, representative confusion matrices are reported for the two strongest full curve models, XGBoost and Gradient Boosting. These matrices were derived from the aggregated out-of-fold predictions of the leading boosted tree models and provide a count-based view of their classification behaviour across the full dataset. These confusion matrices are shown in Figure 30 and Figure 31, respectively.

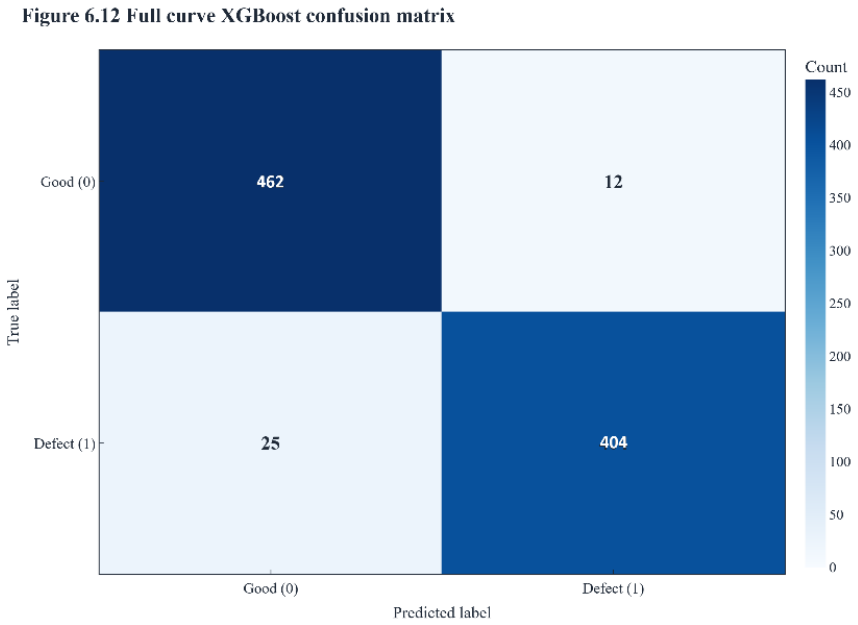


Figure 30. Confusion matrix for full curve XGBoost.

The XGBoost confusion matrix shows a strong and balanced operating point. Only 12 good cycles were incorrectly flagged as defective, while 404 defective cycles were correctly identified. The number of missed defects was 25. This operating pattern is consistent with its leading ROC AUC and PR AUC values, and it confirms that XGBoost provides the strongest overall ranking performance among the full curve models while also maintaining an excellent threshold-based classification balance.

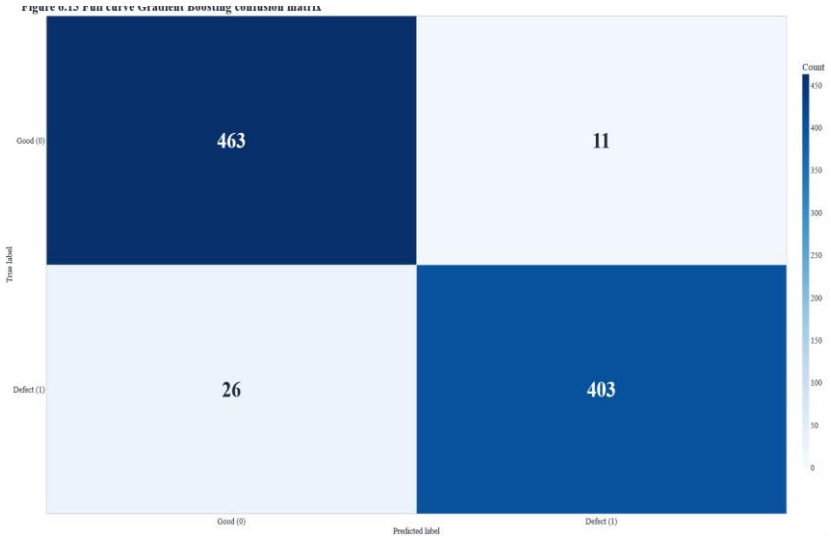


Figure 31. Confusion matrix for full curve Gradient Boosting.

The Gradient Boosting confusion matrix is almost identical to that of XGBoost. It produces one fewer false positive, but one more missed defect. This explains why both models appear tied in the rounded F1 summary, while XGBoost still edges ahead in threshold-independent ranking performance. In engineering terms, both models are highly effective on the full curve representation, but XGBoost offers the strongest overall discrimination across thresholds in this dataset.

A notable result in this section is that the best full curve performance is achieved not by the deep sequence models, but by the boosted tree ensembles trained on flattened full curves. This is an important finding because it shows that, in the present dataset, the main benefit of the full curve route comes from the additional information contained in the full cycle signals rather than from the use of a specialised temporal architecture alone. Flattening the resampled curves creates a high-dimensional but structured tabular input, and the boosted tree models are able to exploit the most informative regions of this space effectively. Their robustness to moderate dataset size, noise, and imbalance appears to be an advantage here. By contrast, deep sequence models such as CNN1D, BiLSTM, and TCN may require either larger datasets, stronger architecture tuning, or more regular signal structure to consistently outperform strong ensemble baselines. This finding is valuable because it prevents an overly simplistic conclusion that full curves necessarily require deep sequence modelling to be useful.

Overall, the full curve results confirm that the richer representation provides very strong defect classification performance. At the same time, the nature of the

winning models is itself informative. The dominant role of boosted tree ensembles suggests that the discriminative information in the cycle is strong and accessible, but that robust nonlinear partitioning of the feature space remains more effective than sequence-specific deep learning in this industrial dataset.

6.9 Direct comparison, peak anchored versus full curve

This section brings the two modelling routes together and presents the central comparison of the chapter. The purpose is to determine whether the compact event anchored peak-based route can approach the classification performance of the full curve route while using a much smaller input per cycle. This is the key comparison because it moves the discussion from model ranking within each route to the broader engineering question of representation choice. In other words, it asks whether the additional information retained in the full curve route leads to a sufficiently large performance gain to justify its greater data and processing cost, or whether the peak anchored route captures most of the useful discriminatory information in a more compact form.

For a fair route-level comparison, the strongest models from each route are compared using the threshold independent ranking metrics that are most stable across operating points. In this study, XGBoost was the strongest ranking model in both routes. In the peak anchored route, XGBoost achieved ROC AUC of 0.991 and a PR AUC of 0.991. In the full curve route, XGBoost achieved ROC AUC of 0.994 and a PR AUC of 0.994. These results show that the full curve route provides the highest overall discrimination, but the margin over the peak anchored route is small. The difference is 0.003 in ROC AUC and 0.003 in PR AUC, which indicates that the compact event anchored representation preserves most of the useful ranking information required for cycle-level defect classification. However, these ROC-AUC values quantify discrimination among held-out cycles under the random stratified cross-validation protocol and should not be interpreted as prospective estimates of performance under temporally separated or previously unseen production regimes.

Table 22. Direct comparison of the best peak-anchored and full curve models

Route	Representative model	Accuracy	Precision	Recall	F1	ROC AUC	PR AUC	Input values per cycle
Peak anchored	XGBoost	0.963	0.981	0.942	0.961	0.991	0.991	C=6
Full curve	XGBoost	0.959	0.97	0.943	0.956	0.994	0.994	C×L=6×970=5820

Table 22 shows that the full curve route provides slightly higher-ranking performance. Still, the peak anchored route remains very close while using only six input values per cycle compared with 5820 values in the full curve representation.

When the comparison is viewed from an operational perspective, the same overall conclusion remains valid. The peak anchored route achieved its strongest count-based operating result with Gradient Boosting, with Accuracy 0.966, Precision 0.974, Recall 0.953, and F1 0.963. In the full curve route, the leading boosted tree models reached an overall Accuracy of 0.959 and an F1 score of 0.956. These values confirm that the compact peak anchored route remains highly competitive, while the fairest direct route comparison is still given by the threshold independent ranking metrics. This means that the compact peak anchored route not only remains close in ranking performance but also achieves a highly competitive operating point in direct classification terms. From an engineering perspective, this is a significant outcome. It shows that a representation built from one physically meaningful event can deliver classification quality that is close to, and in some threshold-based views slightly stronger than, the richer full curve route.

The importance of this result becomes clearer when the input size is considered. In the peak anchored route, each cycle is represented by one value per monitored channel, so the input size per cycle is C. In the present case, where six channels are used, the peak-anchored input contains six values per cycle. In the full curve route, each cycle is represented by C × L values after resampling. This means that the full curve route increases the input size by a factor of L compared with the peak anchored route. Even without assigning a specific numerical value to L in this section, the difference in data burden is substantial. The full curve route, therefore, requires more storage, more preprocessing, and greater computational

effort during both training and inference. A side-by-side visual comparison of these leading models is presented in Figure 32.

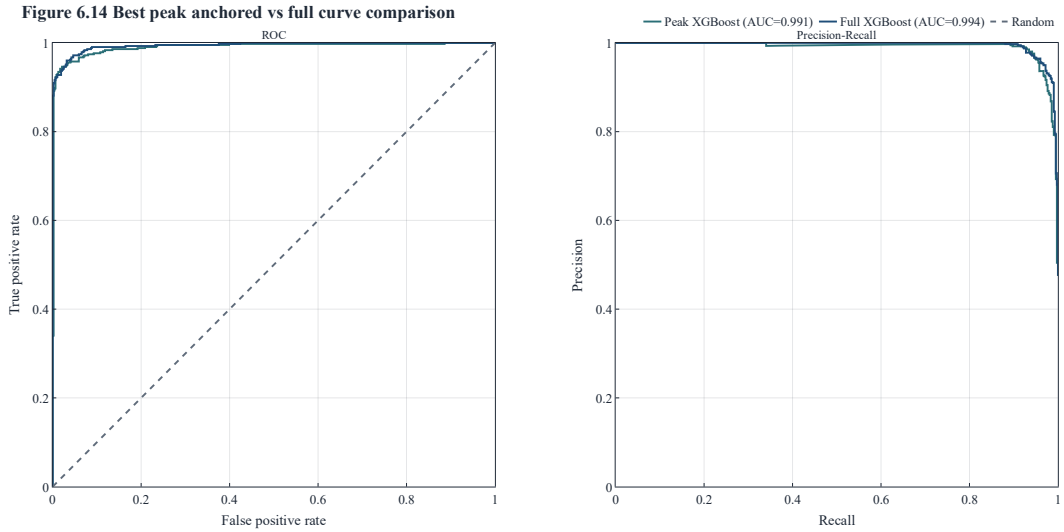


Figure 32. Side-by-side ROC and precision-recall comparison of the best peak-anchored and full-curve models.

The side-by-side curve comparison provides the clearest visual statement of the chapter's main finding. The full curve route remains the strongest in overall discrimination, as expected from its richer representation, but the peak anchored route follows it very closely across the full threshold range. The closeness of the two curves indicates that the information lost by reducing the cycle to the anchor event is smaller than might be expected, at least for the present dataset and defect definitions. This supports the interpretation that the maximum cavity-pressure event is not simply a convenient feature extraction point, but a genuinely informative process event that retains a large share of the predictive discrimination observed in this dataset.

The engineering trade-off can therefore be stated clearly. The full curve route offers the highest discrimination and preserves the complete temporal evolution of the cycle. This makes it a richer representation and the strongest reference benchmark. However, it achieves this at the cost of a much larger input size and greater computational burden. The peak anchored route, by contrast, reduces the cycle to a compact and physically interpretable snapshot centred on the maximum cavity-pressure event. Although it uses far fewer input values, it delivers performance that remains very close to the full curve route. This makes

it highly attractive when the objective is to balance predictive strength with practical efficiency.

From a data analysis perspective, this comparison also strengthens the physical argument made earlier in the chapter. If the full curve route had produced a much larger performance gain, it would suggest that the decisive defect information is distributed widely across the cycle and cannot be captured effectively by a single event. Instead, the observed result shows that the maximum cavity-pressure event and the synchronised machine state at that moment retain much of the predictive information needed for accurate quality classification in this dataset. The full curve route still offers a small but measurable improvement in ranking performance, which indicates that some additional useful information exists outside the anchor event, but the gain is modest relative to the increase in input complexity.

Overall, this direct comparison provides the main proof required by the chapter. It shows that the event anchored peak-based route is not merely a simplified alternative, but a technically credible and highly competitive representation for cycle level quality classification. The full curve route remains the strongest reference in terms of overall discrimination, but the compact peak-anchored route retains much of the observed predictive discrimination with far lower input burden. This result justifies the use of peak-anchored modelling as a practical and scientifically grounded approach for cyclic manufacturing quality prediction, while also explaining why the full curve route remains an important benchmark in the study.

6.10 Discussion and implications for the governed real-time architecture

The results presented in this chapter provide a clear technical basis for the machine-learning layer used later in the governed real-time architecture. The main contribution of the chapter is not only that cycle-level quality classification is feasible, but that it can be achieved using two complementary representations with consistently strong performance. This is important because a manufacturing quality-monitoring architecture requires predictive models that are not only accurate, but also physically meaningful, computationally manageable, and defensible from an engineering perspective.

A key outcome of the study is that the event-anchored peak-based route is a technically credible modelling strategy rather than simply a reduced version of the full-curve approach. The maximum cavity-pressure event was selected as a physically meaningful anchor within the injection moulding cycle, and the results confirm that this event provides a useful reference for cycle-level representation. The strong performance of the best peak-anchored models shows that the synchronised process state at this event retains much of the discriminative information required for reliable cycle-level defect classification in the present dataset. This supports the broader idea that, in cyclic manufacturing, a carefully selected process event can provide a compact and interpretable representation of the production cycle.

The comparison with the full-curve route strengthens this conclusion. The full-curve models provide the highest overall discrimination, which confirms that information outside the anchor event still contributes to classification performance. However, the improvement is small relative to the substantial increase in input dimensionality. The full-curve representation contains 5,820 raw values per cycle, compared with six synchronised sensor values in the basic event-anchored representation. The full-curve route therefore acts as an important reference representation, while the peak-anchored route provides a substantially more compact alternative. The comparison demonstrates that, for the present dataset, much of the useful predictive information can be retained through physically informed event selection without requiring the complete cycle waveform as the primary operational input.

The dominance of tree-based ensemble models in both routes also has important implications. In the peak-anchored route, boosted tree models and random forests outperformed the linear, probabilistic, and neural-network baselines. In the full-curve route, boosted tree models trained on flattened curves also outperformed the deep sequence models. This suggests that the classification problem in this study is strongly influenced by nonlinear interactions and structured relationships that can be captured effectively by ensemble methods. It also indicates that model selection should be guided by the characteristics of the available manufacturing data and the intended deployment role rather than by model complexity alone.

These findings have direct implications for the governed real-time architecture developed in the next chapter. The architecture requires a close relationship between physical process behaviour, digital representation, and data-driven

prediction. The results of this chapter support that requirement in two ways. First, the event-anchored route is directly linked to a physically identifiable point within the production cycle, which improves interpretability and maintains a clear relationship between process behaviour and model input. Second, cycle-level classification aligns naturally with the structure of cyclic manufacturing, where each completed cycle can be linked to one produced part and one quality-related decision. The machine-learning model can therefore operate as a cycle-aware predictive component within the wider monitoring and decision-support architecture.

The chapter also supports a wider interpretation beyond injection moulding. Although injection moulding is the experimental case study used here, the modelling logic may be applicable to other cyclic manufacturing processes where repeatable process events can be identified and linked to product-quality outcomes. In such systems, an event-anchored representation may provide a compact way to connect process behaviour with predictive modelling. The full-curve route remains valuable as a richer reference representation, while the event-anchored route offers advantages in interpretability, input dimensionality, and connection to physical process understanding. The extent to which these advantages translate into storage, computation, or deployment benefits at larger production scales requires further direct evaluation.

At the same time, the results of this chapter should be interpreted within their experimental scope. As discussed in Section 6.6, the stratified five-fold cross-validation provides a consistent basis for comparing the evaluated models and cycle representations under the observed operating conditions, while generalisation to unseen future production regimes requires independent time-ordered validation. The model comparison was conducted using the 903-cycle dataset generated from the present injection moulding case study, and the reported performance therefore reflects the process conditions and defect definitions represented within that dataset. The results do not establish cross-dataset generalisation to data obtained from different machines, moulds, materials, products, sensor configurations, or operating regimes. Demonstrating such transferability would require independent validation using separately acquired datasets. The present chapter therefore establishes comparative predictive performance within the studied case, while the wider questions of runtime deployment, decision governance, drift awareness, evidence storage, and controlled model adaptation are considered separately in the subsequent system-

level evaluation. These system-level considerations are addressed in Chapter 7, where the selected modelling approaches are integrated into the governed real-time architecture for cycle-level quality monitoring and decision support. This conceptual bridge from offline validation to runtime integration is visualised in Figure 33.

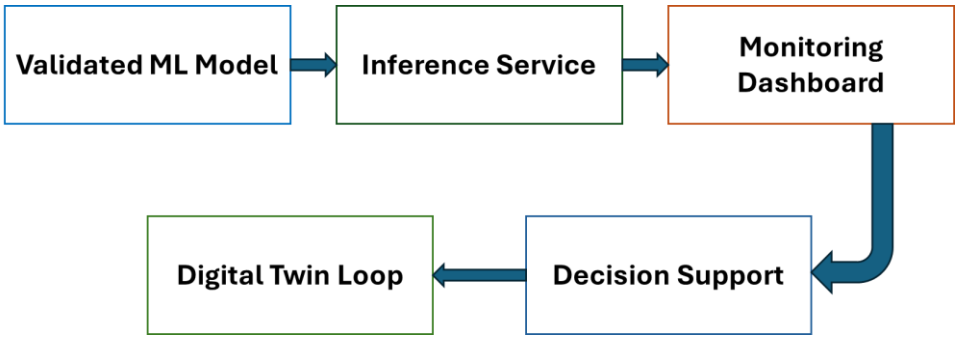


Figure 33. Conceptual bridge from validated machine-learning models to integration within the governed real-time architecture.

6.11 Summary

This chapter provides the analytical and predictive foundation required before machine learning can be integrated into the governed real-time architecture. It demonstrates that cycle-level quality prediction is achievable, that the maximum cavity-pressure event provides a meaningful anchor for compact representation, and that the peak-anchored route can approach the predictive performance of full-curve modelling with substantially lower input dimensionality. These findings justify the use of machine learning as a core predictive component of the architecture while preserving a clear engineering relationship between the physical production cycle, its digital representation, and the resulting model decision.

Chapter 7
Governed Real-Time Architecture for Cycle-Level Monitoring and Decision Support in Cyclic Manufacturing

7.1 Introduction

The previous chapters established the analytical and predictive basis of the work at the cycle level. They showed that the production cycle provides the appropriate decision unit, that the selected process signals contain sufficient physical information to support quality inference, and that two complementary representations can be used for prediction: a richer full-curve route and a compact event-anchored route centred on the maximum cavity-pressure event. These findings are necessary, but they remain incomplete unless they are carried into an operational environment. A model that performs well under offline evaluation is not, by itself, an industrially credible solution. For deployment in cyclic manufacturing, the predictive layer must be embedded within a system that can acquire process data, preserve cycle integrity, apply governed decision logic, support monitoring and review, and maintain a traceable record of what was executed. In this thesis, these functions are brought together within a governed real-time architecture for cycle-level digital representation, quality monitoring, and decision support.

The purpose of this chapter is therefore to define this operational architecture and to explain how validated predictive capability is transferred into governed runtime operation. The chapter does not introduce a new modelling method, nor does it repeat the comparative numerical findings that belong to the results chapter. Its role is to explain how the physical manufacturing cycle is represented inside a computational environment, how each completed cycle is converted into a valid digital object for inference, how complementary model paths are evaluated using that same cycle identity, how the released decision is constrained by an explicit operational policy, and how the important outputs are preserved as auditable run evidence. In this sense, Chapter 7 is the implementation and operational-governance chapter of the thesis.

The architecture is not treated as a passive representation of the machine. It provides an operational decision-support layer. The runtime system receives process signals through the machine-data interface, detects the completion of each production cycle, constructs the required cycle-level representations, scores the cycle through both the fixed Champion and governed Challenger, applies a locked Final policy, and records the resulting state within a governed evidence structure. The contribution is therefore not only that machine learning can be applied to manufacturing data in real time, but that prediction can be embedded within a cycle-resolved and policy-controlled architecture in which sensing,

digital representation, model outputs, released decisions, and later evidence remain connected through a common cycle identity.

The system does not provide autonomous machine control. No released quality decision is automatically returned to the injection moulding machine to modify an actuator or process setpoint. The operational scope of the architecture is therefore quality monitoring, decision support, evidence management, and governed model-management activity. This boundary is important because it distinguishes the implemented system from an automatic process-control architecture while preserving the real-time digital and decision-support functions demonstrated in this research.

The transition from offline model validation to governed runtime operation is conceptualised in Figure 34.

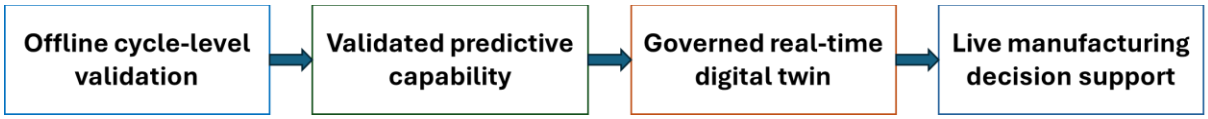


Figure 34. Conceptual transition from offline cycle-level model validation to the governed real-time architecture for manufacturing quality monitoring and decision support.

The architecture was examined through two distinct operational stages, each with a different purpose. The first stage operated without runtime retraining. Its purpose was to test the end-to-end stability of the system under governed conditions and, within that stable setting, to examine how the event-anchored Challenger behaved against the fixed full-curve Champion during runtime operation. This stage was necessary because it provided a clean operational reference for evaluating the system without model updates occurring during the run. It therefore allowed the thesis to investigate the comparative behaviour of the two predictive routes while the deployment policy remained fixed.

The second stage enabled the governed retraining pathway. Its purpose was not simply to obtain a second run, but to examine whether the architecture could admit and execute a controlled Challenger retraining transaction while drift awareness, reviewed correction evidence, retraining gates, policy control, and traceability remained in place. In the preserved evidence, one governed Challenger retraining transaction was completed at cycle 745 and produced a new versioned model artefact. The implementation evidence does not

conclusively establish that the running inference process loaded the replacement estimator for cycles 746–850. For this reason, the second stage is used as evidence of the operation of the governed retraining pathway and artefact-generation process, rather than as prospective performance evidence for the newly trained estimator.

This two-stage structure is essential to the interpretation of the later results. Stage 1 is the stable operational examination stage. It establishes how the governed system behaves when the deployed modelling and policy logic remain fixed and no runtime learning is permitted to modify the operational model path. Stage 2 is the retraining-enabled governance stage. It examines whether retraining can be admitted and completed without breaking cycle integrity, policy control, auditability, or the defined evidence structure. The difference between the two stages is therefore methodological as well as practical.

Stage 1 addresses the question of comparative runtime model behaviour under a locked deployment setting, especially the relationship between the Challenger and the Champion. Stage 2 addresses the question of whether the Challenger retraining pathway can be executed in a disciplined way once the required governance conditions are satisfied. This separation prevents the second stage from being interpreted as a simple repetition of the first run or as evidence that the replacement estimator necessarily became active immediately after retraining.

A further point that must be established at the beginning of this chapter is that the credibility of the architecture depends on more than model discrimination. Operational validity also depends on acquisition reliability, exact cycle accounting, disciplined runtime pacing, and preservation of the evidence needed for later interpretation. The system must maintain one committed record per accepted cycle, preserve run identity and configuration state, and retain the event history required to reconstruct later behaviour. It must also operate at a timing level that protects cycle capture from missed completion events.

For this reason, runtime reliability, persistence, and evidence preservation are treated as core architectural functions rather than as background software details. This is particularly important in manufacturing because an accurate predictive model has limited operational value if the acquisition layer loses cycles, duplicates decisions, associates outputs with the wrong cycle, or cannot support later audit.

The scope of the chapter follows the order of runtime execution. It begins by defining the operational architecture and the relationship between the physical manufacturing process and its cycle-resolved digital representation. It then explains how data acquisition and cycle execution are handled, how online feature construction supports the two predictive routes, and how the fixed Champion, governed Challenger, and locked Final policy interact during operation.

The chapter next explains how run-level persistence and evidence generation are managed, how operator-facing monitoring is incorporated, how drift awareness and human-reviewed evidence enter the runtime pathway, and how retraining is constrained so that model-management actions remain governed rather than unrestricted. It concludes by defining the safeguards that protect runtime validity and by preparing the transition to Chapter 8, where the measured behaviour of the two operational stages is examined in detail.

In this way, Chapter 7 provides the implementation definition of the governed real-time architecture that forms the operational core of the thesis.

Table 23 summarises the role of the two operational stages used to evaluate the architecture.

Table 23. Role of the two operational stages used to evaluate the governed real-time architecture.

Stage	Cycles analysed	Runtime retraining	Operational role	Primary purpose	Model relationship examined	Governance condition	Interpretive value for Chapter 7
Stage 1	850	No	Stable live examination	Evaluate end-to-end governed live operation under fixed deployment conditions.	Compare the event-anchored Challenger against the fixed full-curve Champion during live production.	Locked final policy; no runtime model updates permitted.	Establishes the reference operating condition of the governed real-time architecture before adaptation is introduced.
Stage 2	850	Yes	Controlled live adaptation	Test whether the same governed real-time platform can support disciplined model adaptation during runtime operation.	Assess governed Challenger updating while the Champion remains fixed as the reference path.	Locked final policy retained; one successful governed Challenger retrain occurred at cycle 745.	Shows that retraining can be introduced without breaking cycle integrity, auditability, or policy control.

7.2 Operational Architecture of the Governed Real-Time System

The system developed in this research is implemented as a governed real-time platform for cycle-level quality monitoring and decision support in cyclic manufacturing. Its role is not limited to online signal display, nor is it intended to function only as a prediction script. It is designed as an operational layer that maintains a cycle-resolved digital representation of the manufacturing process during runtime operation. Through this platform, process data are acquired through the defined data interface, organised into complete production cycles, transformed into valid analytical representations, evaluated through the defined model paths, resolved through a locked operational policy, and preserved within a governed runtime record. In this way, the platform connects the physical manufacturing process, its cycle-level digital representation, the predictive layer, and the operational decision layer within one real-time structure.

The architectural basis of the platform is that the manufacturing cycle is the appropriate unit of runtime interpretation and decision-making. In cyclic manufacturing, the quality of the produced part is not determined by a single isolated reading. It develops across the progression of the cycle, including relevant pressure, thermal, hydraulic, positional, and timing behaviour from the beginning of the event until its completion. For that reason, the platform does not treat individual signal samples as independent decision units. Instead, incoming process data are treated as part of a bounded physical event that must first be captured and confirmed as one complete cycle before inference is performed. This ensures that the digital representation remains physically meaningful and that each prediction is associated with one production cycle rather than with an isolated fragment of process behaviour.

Accordingly, the governed real-time platform is organised around a fixed operational path. Process signals are received continuously while the cycle is active. These signals are temporarily buffered so that the in-cycle behaviour is preserved during execution. Once the cycle boundary is confirmed, the buffered signal history is frozen as a completed cycle snapshot. That snapshot then becomes the digital object on which the analytical and predictive layers operate. From this point, the platform constructs the required analytical representations, evaluates the relevant model paths, applies the locked Final decision policy, and stores the resulting output together with the associated run and governance information.

The operational sequence therefore proceeds from continuous process-data acquisition and temporary cycle buffering, through confirmed cycle completion and snapshot formation, to cycle-representation construction, model inference, policy-based decision resolution, and persistent evidence storage. This sequence defines the operational logic of the architecture. The stream stage represents the changing process state during runtime operation. The buffering stage preserves the temporal continuity of the cycle while it remains incomplete. Cycle confirmation establishes the point at which the physical event can be treated as a complete manufacturing unit. The snapshot stage converts that completed event into a stable digital form. Feature construction transforms that representation into valid predictive inputs. Model inference generates the model outputs. The Final policy produces the released operational decision. Persistence records the cycle result and its governance state for later review. The platform therefore operates as a complete cycle-level decision-support structure rather than as a simple monitoring interface. This operational sequence is conceptually visualised in Figure 35.

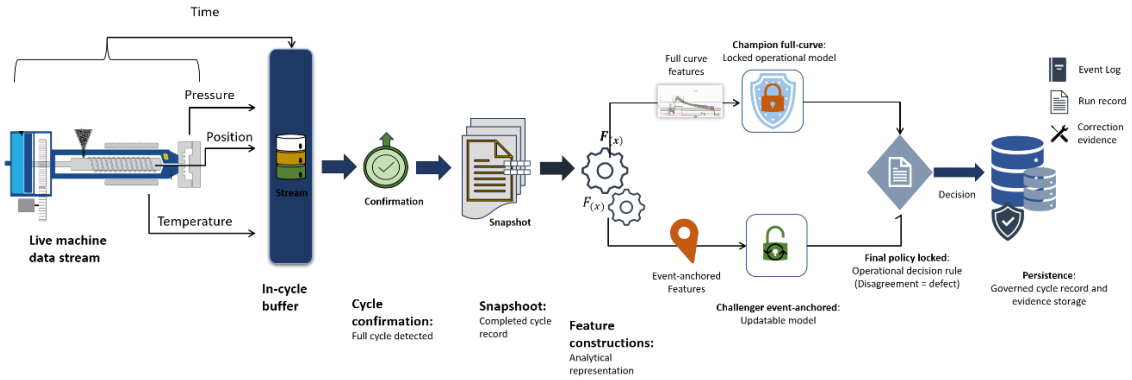


Figure 35. Operational structure of the governed real-time platform, showing the path from process-data acquisition to cycle-level governed decision and evidence storage.

The physical state of the manufacturing process is represented through the selected process variables. These include principal cavity pressure, cavity temperature, hydraulic pressure, screw position, nozzle temperature, hot-water temperature, cycle timing, cycle identification, and cycle-completion states. The use of these variables preserves the physical meaning of the manufacturing process within the digital layer. The platform therefore begins with process quantities that remain directly interpretable in manufacturing terms rather than with abstract numerical features detached from the production process. The

resulting computational state remains linked to the measured condition of the machine and the corresponding production cycle.

The process state observed at time t is represented by the time-varying state vector defined in Equation 11.

$$\mathbf{x}(t) = [\mathrm{MP1}(t), \mathrm{MT1}(t), \mathrm{HP}(t), \mathrm{SP}(t), \mathrm{NT}(t), \mathrm{HWT}(t), \mathrm{TIME}(t), \mathrm{CycleID}(t), \mathrm{CycleDone}(t)]$$

Equation 11. Live process state vector

Where $\mathbf{x}(t)$ denotes the live machine state observed by the governed real-time platform at time t. The vector contains the monitored process signals, cycle timing, cycle identifier and cycle-completion state. The role of the acquisition layer is to observe this evolving state without treating each observation as an independent decision unit. Instead, the platform groups the relevant sequence of observations into one bounded cycle window. If $C_i$represents the i-th accepted cycle, the completed cycle can be expressed as:

$$C_i = \{\mathbf{x}(t) \mid t_{\mathrm{start},i} \leq t \leq t_{\mathrm{end},i}\}$$

Equation 12. Completed cycle window

where $t_{\mathrm{start},i}$and $t_{\mathrm{end},i}$denote the start and end times of cycle i, respectively, and $\boldsymbol{x}(t)$is the live process state. In this form, C_i is not merely a set of raw samples. It is the complete digital record of a manufacturing cycle, preserved as the valid input object for subsequent decision stages. This cycle-based formalisation is important because it shows mathematically that the platform operates on completed physical events rather than on arbitrary signal points.

The next architectural step is the conversion of the completed cycle into one stable snapshot suitable for the analytical layer. If Si denotes the cycle snapshot associated with cycle Ci, then $S_i = \mathrm{Snapshot}(C_i)$

In practical terms, this means that once the cycle boundary is confirmed, the buffered in-cycle signal history is frozen and transferred into a stable data object. That object is then used for feature construction and model evaluation. The snapshot step is fundamental to the architecture because it separates the evolving streaming state from the completed decision state. Before snapshot formation, the process is still physically unfolding. After snapshot formation, the platform can treat the cycle as a complete operational unit. This separation protects the

validity of the later inference and ensures that every decision is linked to a completed cycle rather than to an unfinished process event.

From the accepted snapshot, the platform then constructs the required analytical inputs for the two defined predictive routes. The first route is the richer full-curve representation, which preserves the broader temporal structure of the cycle and supports the fixed reference model. The second route is the lighter event-anchored representation, which reduces the cycle to the most informative physically meaningful event and its aligned process state. If the full-curve feature mapping is denoted by $\phi_{\mathrm{fc}}(\cdot)$ and the event-anchored feature mapping by $\phi_{\mathrm{ea}}(\cdot)$, then the two cycle representations can be expressed as:

$$\mathbf{z}_{\mathrm{fc},i} = \phi_{\mathrm{fc}}(S_i)$$
$$\mathbf{z}_{\mathrm{ea},i} = \phi_{\mathrm{ea}}(S_i)$$

Equation 13. Dual cycle-representation mappings

where $S_i$is the accepted snapshot of cycle i, $\boldsymbol{z}_{\mathrm{fc},i}$is its full-curve representation, and $\boldsymbol{z}_{\mathrm{ea},i}$is its event-anchored representation. This dual representation is a central architectural feature of the governed real-time platform. It allows the same physical cycle to be analysed through two complementary routes without losing the connection between them. One route preserves richer cycle detail as the stable reference path, while the other provides a lighter operational path more suitable for efficient live use and governed updating.

This dual representation leads directly to the dual-model structure of the platform. The full-curve representation is passed to the fixed Champion, which remains unchanged during live operation and serves as the reference model path. The event-anchored representation is passed to the Challenger, which provides the lighter operational model path and is the only path permitted to change under governed retraining conditions. If $p_{\mathrm{champion},i}$ and $p_{\mathrm{challenger},i}$ denote the probability outputs of the Champion and Challenger for cycle i, then these can be written as:

$$p_{\mathrm{champion},i} = M_{\mathrm{champion}}\left(\mathbf{z}_{\mathrm{fc},i}\right)$$
$$p_{\mathrm{challenger},i} = M_{\mathrm{challenger}}\left(\mathbf{z}_{\mathrm{ea},i}\right)$$

Equation 14. Champion and Challenger probability outputs

where $M_{\text{champion}}(\cdot)$ and $M_{\text{challenger}}(\cdot)$ represent the respective model mappings. These two outputs are generated in parallel from the same accepted cycle, but they are not allowed to bypass the operational control structure. This is a defining feature of the architecture. The platform preserves the distinction between raw model output and deployed manufacturing decisions. The main layers of this platform are summarised in Table 24.

Table 24. Main layers of the governed real-time platform and the operational function of each layer.

Layer	Operational function	Role in the architecture
Live machine-data acquisition	Continuously receives the selected process variables from the manufacturing system during production.	Forms the real-time physical input layer of the architecture.
Cycle buffering and boundary confirmation	Buffers incoming samples and confirms when one physical production cycle has been completed.	Ensures that inference is based on complete cycles rather than fragmented signal samples.
Cycle snapshot formation	Freezes the accepted completed cycle as a stable digital object for analysis.	Separates the evolving live stream from the committed decision-ready cycle state.
Online feature construction	Transforms the accepted cycle into the two analytical representations required for live inference.	Creates the full-curve reference input and the event-anchored operational input.
Champion model path	Applies the fixed full-curve reference model to the accepted cycle.	Provides a stable benchmark prediction during live operation.
Challenger model path	Applies the lighter event-anchored model to the same accepted cycle.	Provides the operational prediction path and the only path eligible for governed retraining.
Locked final policy layer	Resolves the Champion and Challenger outputs into one deployed production decision.	Ensures that live deployment is policy-controlled rather than driven by raw model output alone.
Persistence and governed evidence storage	Stores cycle results, probabilities, labels, timings, events, and governance records.	Creates the auditable runtime memory required for later review and traceability.
Monitoring and operator-facing review	Presents runtime status, cycle behaviour, and stored outputs for human inspection.	Supports supervised interpretation of live operation rather than hidden automated execution.
Governed retraining control	Applies drift awareness, human correction input, retraining gates, and controlled model updating.	Allows adaptation to occur under explicit governance instead of uncontrolled automatic change.

The final deployed decision is produced only after the two model paths are resolved through the locked operational policy. This means that the platform does

not assume that the strongest raw model should automatically become the production decision. Instead, the model outputs are passed to the Final policy layer, which applies the predefined deployment rule. The final deployed decision is generated by applying the locked operational policy to the two model outputs. For cycle i, the general policy is expressed as:

$$y_{\text{final},i} = \Pi\left(p_{\text{champion},i}, p_{\text{challenger},i}\right)$$

Equation 15. General final-decision policy

where $y_{\text{final},i}$is the final deployed decision, and $\Pi(\cdot)$is the fixed operational policy that resolves the Champion and Challenger outputs into one production decision. In the present work, this policy is deliberately treated as an explicit part of the architecture rather than as a secondary reporting step. This is important because it shows that the governed real-time platform is policy-controlled, not performance-driven alone. The distinction between model output and deployed output is therefore built directly into the platform design.

A further important feature of the architecture is that persistence is integrated into the same operational path. Once the Final decision has been generated, the platform stores the cycle record together with its associated probabilities, labels, timing values, and governance information. Event records, correction evidence, and retraining records are also preserved within the run structure. The architecture therefore does not produce only transient runtime predictions. It creates a governed operational record in which every accepted cycle and associated decision can later be examined within the same execution history.

This persistence is essential for technical audit, later diagnosis, interpretation of monitoring behaviour, and evaluation of model-management actions. Without a persistent evidence layer, the platform would operate only as a temporary real-time prediction tool. By retaining the cycle identity, predictive outputs, decision reasons, monitoring state, corrections, retraining events, and configuration context, the system provides the traceability required for governed manufacturing decision support.

The same architecture remains in place across the two operational stages used in this study. In the first stage, runtime retraining is disabled so that the platform can be examined under stable governed conditions, particularly to assess the relationship between the Challenger and the fixed Champion while the model paths remain unchanged. In the second stage, the architecture itself remains

unchanged, but the governed retraining pathway is enabled so that controlled Challenger model management can be examined.

The second stage therefore does not represent a different architecture. It uses the same real-time platform with the retraining pathway enabled under the defined governance conditions. One governed Challenger retraining transaction was completed at cycle 745 and generated a new versioned model artefact. As stated in Section 7.1, the preserved implementation evidence does not conclusively demonstrate that the running inference process subsequently loaded the replacement estimator. The value of this stage is therefore the demonstrated execution and traceability of the governed retraining pathway rather than an unsupported claim of immediate post-retraining model deployment.

For these reasons, the governed real-time platform forms the implementation core of this chapter. It defines how the physical manufacturing process is represented digitally at the cycle level, how the analytical and predictive layers are executed, how released decisions are controlled, and how operational evidence is preserved. The next section moves from this system-level architecture to the detailed execution of runtime data acquisition, buffering, cycle-boundary confirmation, and snapshot release.

7.3 Real-time data acquisition and cycle execution

The correct operation of the governed real-time platform depends first on the reliability of its live acquisition layer. If the platform fails to capture the physical cycle correctly, then no later prediction, comparison, policy decision, or retraining step can be interpreted with confidence. For that reason, the acquisition stage in this work is designed around one strict principle: cycle integrity must be secured before inference is allowed to occur. The platform does not classify incoming signal samples one by one as they arrive. Instead, it observes the evolving process continuously, preserves the in-cycle signal history while the physical event is still active, and only releases the data to the analytical layer once the cycle has been confirmed as complete. This makes the acquisition layer part of the governing mechanism of the real-time architecture, not merely its input channel.

This design follows the physical nature of cyclic manufacturing. The produced part is formed across a complete moulding event, not at one isolated sample instant. Pressure, thermal response, screw motion, hydraulic behaviour, and

timing all contribute across the progression of the cycle. A real-time decision taken before the cycle has been properly completed would therefore risk acting on incomplete process information. To avoid this, the platform treats the incoming machine data as a time-ordered evolving process state that must be collected into one bounded cycle record. Only after that record has been validated does it become a legitimate unit for prediction and storage. In this way, the architecture preserves the physical meaning of the production event before applying the computational interpretation required for classification.

The live process state observed by the acquisition layer may be represented as the time-varying signal, as defined in Equation 11, the acquisition layer observes the time-varying process state $\mathbf{x}(t)$ throughout the active production cycle. The variables reflect the principal cavity pressure, cavity temperature, hydraulic pressure, screw position, nozzle temperature, hot-water temperature, cycle timing, cycle identifier, and cycle completion state. These signals are read continuously while the process is running. However, the platform does not interpret each x(t) as a separate decision input. The incoming observations are accumulated in a temporary cycle buffer to preserve the temporal continuity of the active production cycle. If $B_i(t)$denotes the buffer associated with the i-th active cycle, then:

$$B_i(t) = \{\mathbf{x}(\tau) \mid t_{\text{start},i} \leq \tau \leq t,\ \text{cycle } i \text{ is active}\}$$

Equation 16. Active cycle buffer

where τdenotes an observation time within the active cycle, $t_{\text{start},i}$is the accepted start time of cycle i, and $t$is the current acquisition time. The buffer is therefore a live temporary state, not yet a committed analytical object. This distinction is important because it separates the evolving machine process from the completed cycle record used later for decision-making. This acquisition and execution path is visualised in Figure 36.

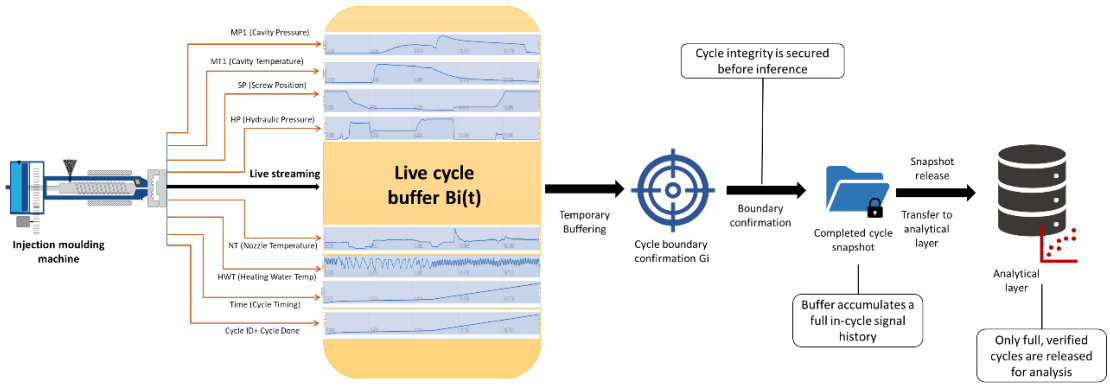


Figure 36. Real-time data acquisition and cycle execution path, showing live streaming, temporary buffering, boundary confirmation, snapshot release, and transfer to the analytical layer.

The central technical requirement is confirmation of the machine-defined cycle boundary. The physical cycle boundary is provided by the ComoNeo system through the cycle identifier and cycle-completion events. The governed real-time platform uses these signals and permits downstream processing only after completion has been confirmed. If $G_i$denotes the completion gate for cycle i, then:

$$G_i = \begin{cases} 1, & \text{if CycleDone confirms completion of cycle } i \\ 0, & \text{otherwise} \end{cases}$$

Equation 17. Machine-confirmed cycle-completion gate

Only when $G_i = 1$does the completion condition is satisfied does the platform permit the active buffer to proceed to the next execution stage. This gate prevents partial or incomplete cycles from entering the predictive pipeline. This rule is simple in form but essential in meaning. It ensures that the platform does not commit partial cycles, and it prevents the predictive layer from acting on incomplete physical events. The platform therefore uses boundary confirmation as the formal gate between observation and decision.

Once the machine-confirmed completion gate is satisfied, the active buffer is frozen and converted into a completed cycle record. If $C_i$denotes the completed record for cycle i, then:

$$C_i = B_i\left(t_{\text{end},i}\right) \qquad \text{subject to} \qquad G_i = 1$$

Equation 18. Formation of the completed cycle record

Where $B_i\left(t_{\text{end},i}\right)$ is the content of the active cycle buffer at the accepted cycle-end time. The condition $G_i = 1$ ensures that only a machine-confirmed cycle is

committed for downstream processing. Thus, the completed cycle C_i is created by freezing the buffered signal history at the accepted cycle boundary. Before confirmation, the process data remain provisional; after confirmation, the cycle becomes a complete digital record of one physical manufacturing event. This transition marks the point at which the live process data become a valid cycle-level decision unit.

The next execution step is the formation of a stable cycle snapshot. Although the completed cycle C_i contains the accepted in-cycle signal history, a stable operational representation is required for transfer to the analytical layer. The snapshot S_i is therefore formed as:

$$S_i = \text{Snapshot}(C_i)$$

Equation 19. Formation of the cycle snapshot

where Snapshot(·) denotes the operation that fixes the completed cycle as a stable input for downstream processing. The snapshot S_i separates the committed cycle record from the live acquisition state. Once the snapshot has been created, the acquisition buffer can be reset for the next physical cycle without affecting the analytical processing of the completed cycle. The main stages of this execution process are summarised in Table 25.

Table 25. Main stages of cycle execution within the governed real-time platform, from live signal acquisition to validated cycle snapshot release.

Execution stage	Operational meaning	Governance role	State label
Live signal acquisition	The platform continuously reads the selected machine variables while production is active.	Preserves the physical process state as the input to the governed real-time architecture.	q_0 to q_1
Temporary cycle buffering	Incoming observations are accumulated into an active in-cycle buffer while the cycle remains open.	Prevents early inference on incomplete process information.	q_1
Cycle-boundary confirmation	The platform checks the cycle identifier and completion signal to determine whether the physical cycle is complete.	Acts as the formal gate between observation and decision.	q_2
Committed cycle formation	Once completion is confirmed, the active buffer is frozen as one accepted completed cycle record.	Enforces the one-cycle, one-commit discipline and prevents duplicate or partial commits.	q_3
Snapshot release	The committed cycle is converted into a stable snapshot for downstream processing.	Separates the live acquisition state from the analytical decision state.	q_3

Execution stage	Operational meaning	Governance role	State label
Transfer to the analytical layer.	The stable snapshot is passed forward for feature construction, model scoring, policy resolution, and storage.	Ensures that only validated completed cycles enter inference and persistence.	q_3 to q_4
Buffer reset and readiness for next cycle	After a successful transfer, the acquisition buffer is cleared, and the system returns to the waiting state.	Supports reliable continuous operation without losing exact cycle accounting.	q_4 to q_0

A critical implementation issue in this section is the prevention of missed cycle commits, duplicate commits, and incomplete cycle releases. If the platform commits the same cycle more than once, the run record becomes unreliable. If it fails to commit a cycle, the operational history becomes incomplete. If it commits too early, the stored record no longer corresponds to a full physical production event. The execution logic is therefore designed to enforce a one-cycle, one-commit discipline. The active buffer continues to accumulate only while the cycle remains open. It is frozen only once the completion condition is satisfied. It is cleared only after the completed cycle has been successfully transferred forward. This sequencing ensures that the governed real-time platform maintains one unique, committed record for each accepted physical cycle.

This requirement became especially important in the final operational stage because live execution had to remain reliable even when cycle-completion signalling could be brief. The final platform behaviour therefore adopted a reliability-first pacing approach so that the acquisition layer had enough time to detect the true completion transition and commit the cycle correctly. This point is not a minor implementation detail. It is part of the engineering validity of the governed real-time architecture. A real-time platform cannot be considered operationally credible if the communication timing causes dropped cycles or unstable boundary detection. Reliable pacing was therefore necessary to ensure that cycle capture remained correct and that the platform preserved exact cycle accounting during runtime operation. The cycle-execution logic can also be expressed as a sequence of operational states. Let q_0 to q_4 denote the five successive runtime states. The execution path is:

$$q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_4$$

Equation 20. Cycle-execution state sequence

where:

q_0: waiting for cycle activity;

q_1: active acquisition and buffering;

q_2: machine-confirmed cycle-boundary acceptance;

q_3: completed-cycle commitment and snapshot release; and

q_4: buffer reset and return to the waiting state.

This state progression is useful because it shows that real-time execution is not a loose stream-handling process. It is a governed sequence with explicit operational states. The platform moves from waiting, to observing, to confirming, to committing, and then back to readiness for the next cycle. This structure supports stable, continuous operation and helps preserve interpretive clarity when later analysing timing, events, and retraining behaviour.

The same acquisition and execution logic is retained across both operational stages used in the study. In the first stage, runtime retraining is disabled so that the system can be examined under stable live conditions while the relationship between the Challenger and the fixed Champion is observed. In the second stage, governed retraining is enabled, but the cycle-execution path itself remains unchanged. The platform still reads, buffers, confirms, commits, and snapshots each cycle in the same disciplined way. This continuity is important because it means that the second stage does not alter the validity of the acquisition layer. Any later change in model behaviour arises from the governed analytical path, not from a different data-capture method.

A further strength of this arrangement is that it supports consistent persistence and later audit. Because each cycle is explicitly confirmed before it enters the analytical layer, the stored run history can be interpreted as a sequence of valid production events rather than as a mixture of incomplete signal fragments. This strengthens the meaning of the associated event logs, prediction records, human corrections, and retraining records, because all of them rest on the same confirmed cycle structure. The architecture therefore maintains a coherent operational record in which every later decision can be traced back to one accepted physical cycle.

For these reasons, the acquisition layer should be understood as one of the most important governing layers in the governed real-time platform. It establishes the

boundary of the production event, protects the validity of cycle-level execution, and determines when the physical process has become complete enough to enter the computational decision path. Only after that point can the platform construct the analytical representations required for the dual-model structure. The next section moves from cycle execution to the question of how the accepted cycle snapshot is transformed into the online representations used by the Champion and Challenger paths.

7.4 Online Cycle Representation and Feature Construction

Once a completed cycle has been accepted by the acquisition layer and fixed as a valid snapshot, the next task of the governed real-time platform is to convert that physical cycle record into analytical forms suitable for governed inference. This step is necessary because the accepted cycle snapshot, although complete and physically meaningful, is not yet the final structure used by the predictive layer. The platform must still transform the cycle into representations that preserve the essential process information while remaining consistent with the logic of the two model paths. For that reason, the representation stage is not a minor pre-processing step. It is the point at which the completed physical cycle becomes a usable digital object for real-time decision-making.

A central design principle in this work is that one accepted physical cycle should support more than one valid analytical view. The platform does not force the cycle into a single feature form. Instead, it constructs two linked representations from the same accepted snapshot. The first is a richer full-curve representation that preserves broader information from across the cycle and serves as the reference analytical route. The second is a lighter event-anchored representation centred on the first physically meaningful cavity-pressure peak and the aligned process state at that instant. These two forms are not competing descriptions of different data. They are two deliberate interpretations of the same completed cycle, designed to support two different operational roles within the governed architecture.

Let the accepted snapshot for the i-th cycle be denoted by Si. The representation layer applies two mapping functions to this snapshot. The first mapping generates the full-curve representation, while the second mapping generates the event-anchored representation. The full-curve and event-anchored

representations are generated using the mappings defined in Equation 13, where they define the two analytical routes through which the governed real-time platform interprets the accepted cycle before inference is performed. In this way, the representation stage preserves a strict connection between the physical cycle and the downstream decision paths.

The full-curve representation is intended to retain the wider temporal structure of the cycle. Its purpose is to preserve the broader process behaviour from which the fixed reference model can draw its judgement. In practical terms, this route may contain sampled, aggregated, or derived information taken across the cycle window, provided that the representation remains consistent with the trained reference path. The main value of this route is stability. It retains more of the cycle's temporal content and therefore acts as the richer structural description of the physical event. For this reason, it is the natural input to the fixed Champion. The Champion is not intended to be the most agile route in the live platform. It is intended to be the stable benchmark path against which the lighter route can be judged.

The event-anchored representation serves a different purpose. Rather than preserving a wider curve description, it compresses the cycle into a compact state centred on the maximum cavity-pressure event, referred to here as the MP1 event. This design is important because the thesis has already established that this event is physically meaningful in the moulding process and provides a reliable reference point for synchronising key process variables. Instead of asking the lighter live route to process a broader temporal pattern each time, the platform extracts the most informative event and the process state aligned to it. The result is a smaller but still physically grounded representation, better suited to responsive real-time operation and to controlled updating under governed retraining. Let $\mathrm{MP1}_i[k]$ denote the sampled cavity-pressure signal for cycle i, where k is the sample index. The index of the selected MP1 peak event is defined as:

$$k_{\mathrm{MP1},i} = \underset{k \in \mathcal{K}_i^{\mathrm{valid}}}{\arg\max}\ \mathrm{MP1}_i[k]$$

Equation 21. MP1 peak-event index

where $\mathcal{K}_i^{\mathrm{valid}}$ is the valid peak-search region for cycle i, and $k_{\mathrm{MP1},i}$ is the sample index at which the selected MP1 peak occurs. The valid search region is

constrained by the accepted cycle window and the defined peak-selection logic, preventing an arbitrary local fluctuation from being selected as the event anchor. In practical use, the valid peak-search region is constrained by the accepted cycle window and the event-detection logic defined for the process, so that the event corresponds to the physically meaningful cavity-pressure peak rather than to an arbitrary local fluctuation. The purpose of this expression is to show that the event-anchored route is not based on a vague qualitative notion of "the peak". It is based on a defined event location inside the accepted cycle record.

Once the MP1 event index has been identified, the platform aligns the selected process variables at the same sample. Let $v_{j,i}[k]$ denote the value of the j-th process variable at sample k of cycle i. Its value at the selected MP1 event is defined as:

$$a_{j,i} = v_{j,i}\left[k_{\mathrm{MP1},i}\right]$$

Equation 22. Process-variable alignment at the MP1 event

where $a_{j,i}$ is the aligned value of the j-th process variable in cycle i, and $k_{\mathrm{MP1},i}$ is the MP1 event index defined in Equation 21. This alignment step is a key part of the representation logic because it preserves the synchronised physical state of the cycle at the selected event. The platform does not use the peak value in isolation. It uses the peak as a reference point from which the corresponding thermal, hydraulic, positional, and timing state can be read consistently.

The event-anchored feature vector for cycle i contains one synchronised value from each of the six monitored process channels:

$$\mathbf{z}_{\mathrm{ea},i} = \left[\mathrm{HP}_{\mathrm{peak},i}, \mathrm{SP}_{\mathrm{peak},i}, \mathrm{NT}_{\mathrm{peak},i}, \mathrm{HWT}_{\mathrm{peak},i}, \mathrm{MT1}_{\mathrm{peak},i}, \mathrm{MP1}_{\mathrm{peak},i}\right]$$

Equation 23. Event-anchored feature vector

where each element is sampled at the MP1 event index defined in Equation 21. The corresponding event time $t_{\mathrm{MP},i}$ is retained as cycle-alignment and traceability information but is not included as an additional model-input feature. Consequently,

$N_E = 6$, consistent with the six-value representation defined in Equation 8. This extraction and feature-construction process is illustrated in Figure 37.

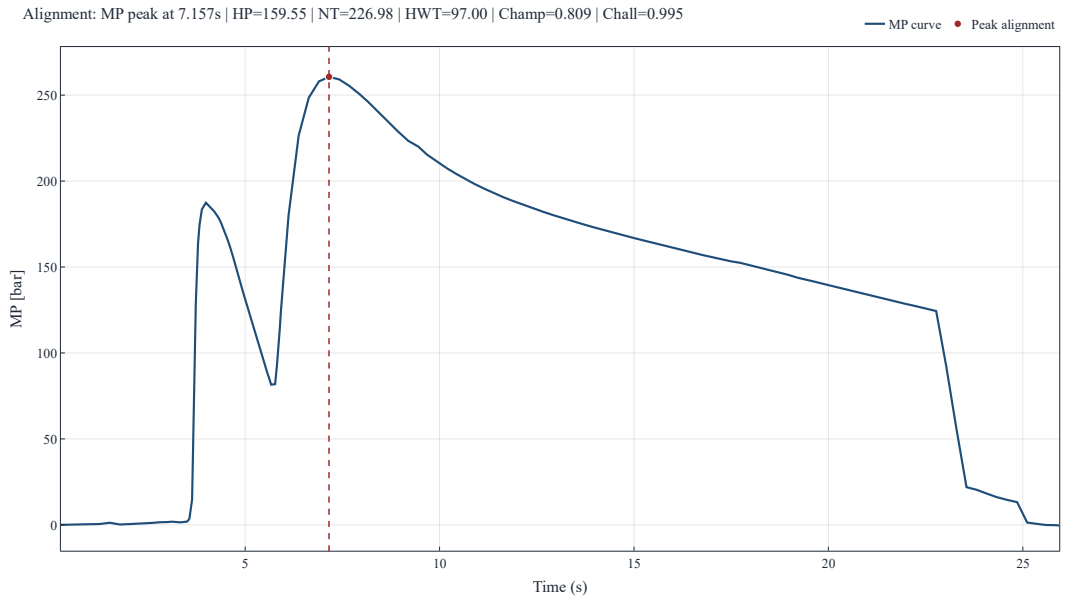


Figure 37. Online cycle representation and event-anchored feature construction, showing the accepted cycle snapshot, MP1 event location, and aligned variables used to form the compact live feature vector.

The full-curve and event-anchored routes should not be interpreted as redundant duplication. Their coexistence is one of the most important design features of the governed real-time platform. The full-curve route preserves a richer structural description and therefore supports a stable fixed reference path. The event-anchored route preserves a compact but physically meaningful state and therefore supports the more agile operational path. By maintaining both, the platform gains two important capabilities at the same time. First, it can compare a lighter live route with a richer benchmark during stable operation. Second, it can allow controlled adaptation only on the lighter route without losing the fixed structural reference provided by the fuller path. This is precisely the balance required by the two operational stages of the study. A comparison of these two representations is summarised in Table 26.

Table 26. Comparison of the two online cycle representations used in the governed real-time platform: full-curve reference representation and event-anchored operational representation.

Representation	Input basis	Main content	Operational role	Main strength	Governed updating
Full-curve reference representation	Accepted complete cycle snapshot	Broader temporal structure of the cycle, including wider in-cycle process behaviour.	Reference analytical route for the fixed Champion model.	Richer structural description and higher interpretive stability.	No. This route remains fixed during live operation.
Event-anchored operational representation	Accepted complete cycle snapshot	Compact feature vector centred on the first meaningful MP1 peak and the aligned process state at that event.	Operational analytical route for the Challenger model.	Lighter real-time use with preserved physical meaning.	Yes. This is the only path eligible for governed retraining.

The relationship between the two representations can also be described in terms of their role in the operational stages. In the first stage, where runtime retraining is disabled, the two representations allow the study to examine how closely the event-anchored route can match, or in some conditions exceed, the reference route under the same live cycle conditions. The purpose here is not simply to prove that two models can run in parallel. It is to test whether the lighter representation remains sufficient for live decision support while the richer route continues to provide the reference structure. In the second stage, where governed retraining is enabled, the same dual representation becomes even more important. The fixed full-curve route remains unchanged, while the event-anchored route becomes the only analytical path eligible for controlled update. This means that the architecture of the feature layer itself supports the governance logic of the governed real-time system.

A further important point is that feature construction in the live platform must remain consistent with the accepted cycle boundary defined earlier. Neither representation is created from partially observed data. Both $z_{fc,i}$ and $z_{ea,i}$ are derived only from the accepted snapshot Si. This ensures that feature generation remains linked to the same completed physical event that entered the governed real-time architecture through the acquisition layer. As a result, the representation stage preserves continuity between physical capture, analytical transformation, and later model inference. This continuity is essential for

interpretability. If the features were generated from partial or inconsistent cycle fragments, then the platform could no longer claim that its decision corresponds to one valid production cycle.

The representation stage, therefore, performs a critical bridging function inside the governed real-time platform. It takes the completed cycle produced by the acquisition layer and converts it into the precise analytical objects required for governed inference. At the same time, it preserves the physical meaning of the cycle by ensuring that both feature routes are derived from the same accepted production event. This makes the two-model structure logically possible and keeps the platform anchored to the actual process rather than to detached numerical abstractions.

For these reasons, online cycle representation and feature construction should be understood as a core architectural layer rather than as background preprocessing. It is here that the platform translates the physical cycle into the two analytical representations required for runtime operation: a richer, stable reference representation and a lighter event-centred operational representation. The next section, therefore, moves from representation to the model layer itself, where the roles of the Champion, the Challenger, and the locked Final policy are defined within the governed decision structure.

7.5 Champion, Challenger, and final decision policy

The predictive layer of the governed real-time platform is intentionally organised as a governed three-part structure rather than as a single-model decision path. This arrangement is necessary because the objective of the live system is not only to obtain a prediction, but to control how that prediction is produced, compared, and released during operation. The platform distinguishes between the Champion, the Challenger, and the Final deployed decision. Each of these has a different role. The Champion provides the fixed reference path, the Challenger provides the lighter operational path, and the Final layer provides the locked deployment decision after the model outputs have been resolved through the active policy rule. This separation is a central part of the thesis argument because it shows that live deployment is governed by explicit structure rather than by direct reliance on whichever model appears stronger at a given moment.

The Champion is the fixed full-curve reference model. It is supplied by the richer representation described in the previous section and remains unchanged during live operation. Its purpose is to preserve a stable analytical benchmark against which the lighter route can be judged. This stability is important for two reasons. First, it gives the platform a consistent reference path across both operational stages, which makes later comparison meaningful. Second, it protects the governed architecture from losing its baseline when the live operational path is being examined or, in the second stage, when retraining is enabled. The Champion is therefore not merely an alternative model running in parallel. It is the structural anchor of the predictive layer.

The Champion is therefore not retained because it is expected to be the most efficient deployed classifier under every live condition. Its role is to provide a fixed reference against which the behaviour of the lighter and updateable Challenger can be interpreted. This is important because, once retraining is enabled, the adaptive path and the operating environment may both change. The fixed Champion preserves a stable benchmark for comparison, traceability, and governance, even when the Challenger later proves stronger in raw live classification performance. Because the Champion remains fixed, this mapping does not change during either operational stage. Its behaviour therefore provides a stable reference against which the behaviour of the lighter live path can be interpreted.

The Challenger is the lighter event-anchored model path. It receives the compact cycle representation centred on the selected MP1 event and its aligned process state. Its role is different from that of the Champion. It is designed to represent the operational path that can be used more efficiently in live conditions while remaining physically grounded in the cycle. In the first operational stage, the Challenger is examined against the Champion under stable no-retrain conditions so that its live behaviour can be assessed without any model update occurring during the run. In the second operational stage, the Challenger becomes the only path eligible for governed retraining, which makes it the adaptive path of the platform. Its lighter representation and update eligibility are therefore directly linked. However, that possibility does not mean the Challenger is allowed to bypass the operational control structure. Even when it is the stronger raw path, it remains only one model component inside the governed decision architecture. The Champion and Challenger probability outputs are generated using the model mappings defined in Equation 14.

The coexistence of the Champion and Challenger is deliberate. The platform is not structured this way simply to run two models for comparison. The two-path design serves a deeper operational purpose. The fixed Champion preserves a stable benchmark based on the richer representation. At the same time, the Challenger provides the lighter live route whose behaviour can be examined under real operating conditions and, when permitted, updated in a controlled way. This allows the governed real-time platform to examine whether a compact, event-centred path can match or exceed the benchmark in practice without losing the reference path needed for governed deployment. The dual-model arrangement therefore supports both evaluation and operational control at the same time. This predictive decision structure is visualised in Figure 38.

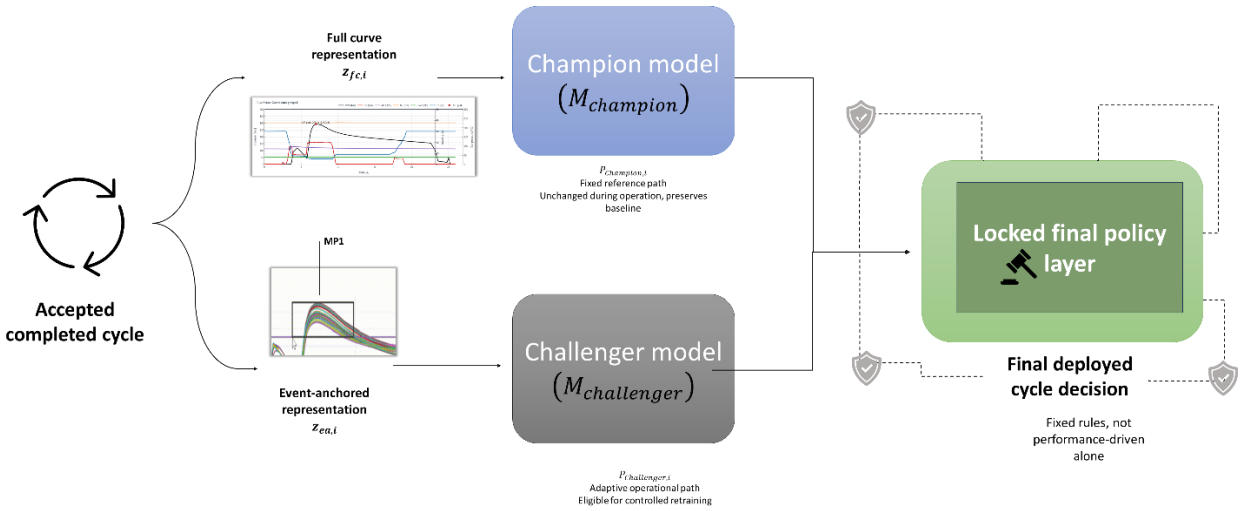


Figure 38. Predictive decision structure of the governed real-time platform, showing the fixed Champion path, the lighter Challenger path, and the locked Final policy layer.

The Final layer is the deployment decision layer. Its function is to convert the Champion and Challenger outputs into one explicit operational decision for the cycle. This is where the predictive architecture becomes a governed manufacturing decision structure. The Final layer is not another predictive model trained independently from the first two. It is a policy-controlled resolution layer that uses the two model outputs as inputs to a fixed operational rule. In this thesis, that rule is deliberately treated as a formal part of the architecture because the central argument is that live deployment must remain policy-controlled, not performance-driven alone. The general relationship between the two model outputs and the deployed decision is defined in Equation 15. This expression is

important because it separates the meaning of raw model output from the meaning of released operational output. A probability estimated by one model is not automatically the final decision. The final decision exists only after the policy layer has acted.

In this study, the active policy is a conservative disagreement-resolution rule. The final decision passes through the common class prediction when the Champion and Challenger agree. When they disagree, the cycle is classified as defective. For cycle i, the deployed rule is:

$$y_{\text{final},i} = \begin{cases} y_{\text{champion},i}, & y_{\text{champion},i} = y_{\text{challenger},i} \\ 1, & y_{\text{champion},i} \neq y_{\text{challenger},i} \end{cases}$$

Equation 24. Conservative disagreement-resolution policy

where $y_{\text{champion},i}$ and $y_{\text{challenger},i}$ are the thresholded Champion and Challenger class decisions, respectively, with 1 denoting a defective cycle and 0 denoting a good cycle. Because the class labels are binary, this rule is equivalent to

$y_{\text{final},i} = \max\left(y_{\text{champion},i}, y_{\text{challenger},i}\right)$. In this structure, agreement between the two model paths allows the common decision to pass through, while disagreement is resolved conservatively in favour of defect. This policy preserves a cautious deployment behaviour and reduces the risk of releasing an uncertain cycle as acceptable production.

The final policy intentionally adopts a conservative operational position. Its objective is not to minimise false alarms or maximise classification metrics alone. Instead, it prioritises the reduction of false negatives, where potentially defective products could otherwise be released as acceptable production. In manufacturing environments, the consequences of defect escape may include customer complaints, warranty costs, rework, process investigation, production disruption, or loss of confidence in the monitoring system. The policy represents a risk-management decision rather than a purely statistical optimisation strategy. Any increase in false-positive classifications is treated as an accepted operational trade-off in exchange for greater protection against defect escape.

This conservative rule is especially important in the first operational stage. The purpose of that stage is to examine how the Challenger compares with the fixed Champion during live use without allowing runtime learning to change the active path. The disagreement-based Final rule ensures that this examination takes

place inside a controlled deployment setting. The platform can therefore observe when the Challenger matches the Champion, when it diverges from it, and how often the final operational result becomes more conservative than either path alone. This gives the first stage a clear methodological role. It is not only a performance check. It is an operational examination of the model relationship under fixed deployment governance.

In the second operational stage, the same Final policy remains in place even when retraining is enabled. This continuity is essential. It means that the introduction of governed retraining does not remove or weaken the deployment rule. The Challenger may be updated if the required conditions are satisfied, but the deployed cycle decision still passes through the same locked Final structure. This preserves interpretive continuity across the two stages and prevents the retraining stage from being misread as unrestricted model substitution. The architecture, therefore, allows adaptation only within a persistent policy-controlled shell.

The Champion and Challenger probability outputs are converted into binary class decisions before the final policy is applied. Let θ_{C} and θ_{H} denote the active Champion and Challenger thresholds, respectively. The class decisions are defined as:

$$y_{\mathrm{champion},i} = \begin{cases} 1, & p_{\mathrm{champion},i} \geq \theta_{\mathrm{C}} \\ 0, & p_{\mathrm{champion},i} < \theta_{\mathrm{C}} \end{cases}$$
$$y_{\mathrm{challenger},i} = \begin{cases} 1, & p_{\mathrm{challenger},i} \geq \theta_{\mathrm{H}} \\ 0, & p_{\mathrm{challenger},i} < \theta_{\mathrm{H}} \end{cases}$$

Equation 25. Thresholding of the model probability outputs

where class 1 denotes a defective cycle and class 0 denotes a good cycle. This thresholding stage converts the probabilistic model outputs into the discrete Champion and Challenger decisions used by the final policy in Equation 24. This thresholding step is necessary because the Final policy operates on discrete decision outcomes, not on raw probability values alone. The threshold layer, therefore, acts as the bridge between probabilistic inference and governed operational release. The functional roles of these three decision layers are summarised in Table 27.

Table 27. Functional roles of the Champion, Challenger, and Final layers within the governed decision structure of the real-time platform.

Layer	Input representation	Primary function	Operational character	Update status during live operation
Champion	Full-curve reference representation	Provides the fixed benchmark probability for each accepted cycle.	Stable reference path during both stages.	Not updated
Challenger	Event-anchored operational representation	Provides the lighter live operational probability for each accepted cycle.	Agile operational path examined in Stage 1 and eligible for governed retraining in Stage 2.	May be updated under governed retraining conditions
Final policy layer	Champion and Challenger class decisions/probabilities after thresholding	Resolves the two model outputs into one deployed production decision.	Locked deployment-governance layer that preserves conservative operational release.	Policy remains fixed across the two stages.

The policy structure also supports a cost-sensitive interpretation of classification errors. This analysis does not assign measured financial values to individual cycle decisions. Instead, the false-positive cost is treated as one reference unit, while a false negative is assigned a relative penalty r. For a truth-labelled run containing N cycles, the normalised decision loss is defined as:

$$L_r = \frac{\mathrm{FP} + r\,\mathrm{FN}}{N}$$

Equation 26. Normalised decision loss under a relative error-cost ratio

where FP and FN are the numbers of false positives and false negatives, respectively, r is the assumed false-negative to false-positive cost ratio, and N is the number of truth-labelled cycles. Here, normalisation refers specifically to division by N. The numerator $FP + rFN$ represents the total weighted classification-error burden under the assumed relative penalty r, while division by N expresses this burden on a per-cycle basis. The resulting L_r therefore represents a normalised relative decision loss per evaluated cycle rather than a measured monetary cost. When $r = 1$, the expression reduces to the conventional misclassification rate, $L_r = (FP + rFN)/N$. Because r may be greater than 1, the term normalised does not imply that L_r is necessarily bounded between 0 and 1. This measure is used as a sensitivity analysis rather than as a complete economic

model. It allows the Champion, Challenger, and Final outputs to be compared not only by statistical metrics, but also by how their errors would affect deployment decisions under different assumed penalty ratios.

One of the most important outcomes of this structure is that it allows one to make a stronger claim than a simple online prediction. The live platform does not merely show that two model paths can run in real time. It shows that a manufacturing decision can be released only after passing through a controlled architecture in which benchmark stability, operational efficiency, and deployment caution are all represented separately. This is precisely why the Champion, Challenger, and Final layers must remain distinct in the chapter. Without that distinction, the architecture would lose its governed character and would reduce to a direct model-serving system.

The next section moves from the predictive decision structure to the runtime memory that supports it. Once the Champion, Challenger, and Final outputs have been produced for each accepted cycle, the platform must preserve them in a stable run-level record together with the events, timing information, corrections, and governance traces needed for later review.

7.6 Run-Level Persistence, Traceability, and Evidence Structure

A governed real-time architecture cannot be regarded as a credible operational system if its decisions disappear once they are produced. For that reason, the real-time platform developed in this work includes a dedicated persistence and evidence layer as part of its core architecture. The purpose of this layer is to preserve the full runtime history of the system in a structured and traceable form so that each accepted cycle, each generated decision, and each governance-related event can be examined after execution. This is essential for two reasons. First, it ensures that the live platform remains auditable rather than transient. Second, it allows the later analytical chapters to interpret system behaviour on the basis of stored operational evidence rather than on temporary screen output or informal observation.

The persistence layer begins at the point where the accepted cycle has already passed through feature construction, model inference, and final policy resolution. At that stage, the cycle is no longer only a physical event mirrored in digital form.

It has become a governed operational record containing the cycle identifier, the predictive outputs of the Champion and Challenger, the deployed Final decision, timing information, and the associated runtime state. The system then commits this record into the run structure as one traceable cycle entry. In this way, each completed production cycle produces not only a decision, but also a permanent digital record of how that decision was formed during live operation.

Let R_i denote the committed operational record for cycle i. Its general structure is defined as:

$$R_i = \left[\text{cycle_id}_i, \mathbf{z}_{\text{fc},i}, \mathbf{z}_{\text{ea},i}, p_{\text{champion},i}, p_{\text{challenger},i}, y_{\text{final},i}, t_{\text{proc},i}, \text{meta}_i\right]$$

Equation 27. Committed operational cycle record

where cycle_id_i identifies the accepted cycle; $\mathbf{z}_{\text{fc},i}$ and $\mathbf{z}_{\text{ea},i}$ are the full-curve and event-anchored representations; $p_{\text{champion},i}$ and $p_{\text{challenger},i}$ are the Champion and Challenger probability outputs; $y_{\text{final},i}$ is the deployed decision; $t_{\text{proc},i}$ is the processing time; and meta_i contains the associated governance and runtime metadata. The implementation may extend these fields, but each accepted cycle is retained as one traceable governed record.

The persistence layer is organised at the run level rather than as an unstructured accumulation of independent records. This means that all stored cycle records belong to one defined execution run with a distinct identity, configuration state, and runtime boundary. The run, therefore, acts as the governing container within which all live activity is interpreted. This is a major part of the thesis logic because the meaning of a cycle decision depends not only on the cycle itself, but also on the conditions under which the full run was executed. Configuration settings, active policy mode, enabled or disabled retraining state, event logs, and stored evidence all belong to the same run-level context. The platform, therefore, preserves not only per-cycle behaviour but also the operational identity of the stage in which that behaviour occurred.

This run-level organisation is especially important because the study uses two operational stages, each with a different purpose. In the first stage, runtime retraining is disabled so that the governed real-time platform can be examined under stable operating conditions and the relationship between the Challenger and the fixed Champion can be assessed without model update during execution. In the second stage, the same real-time platform is operated with governed

retraining enabled so that controlled adaptation can be evaluated. Because both stages are stored through the same run-based evidence structure, the two stages remain comparable without confusion. The persistence layer, therefore, helps preserve methodological clarity as well as technical traceability. This structure is visualised in Figure 39.

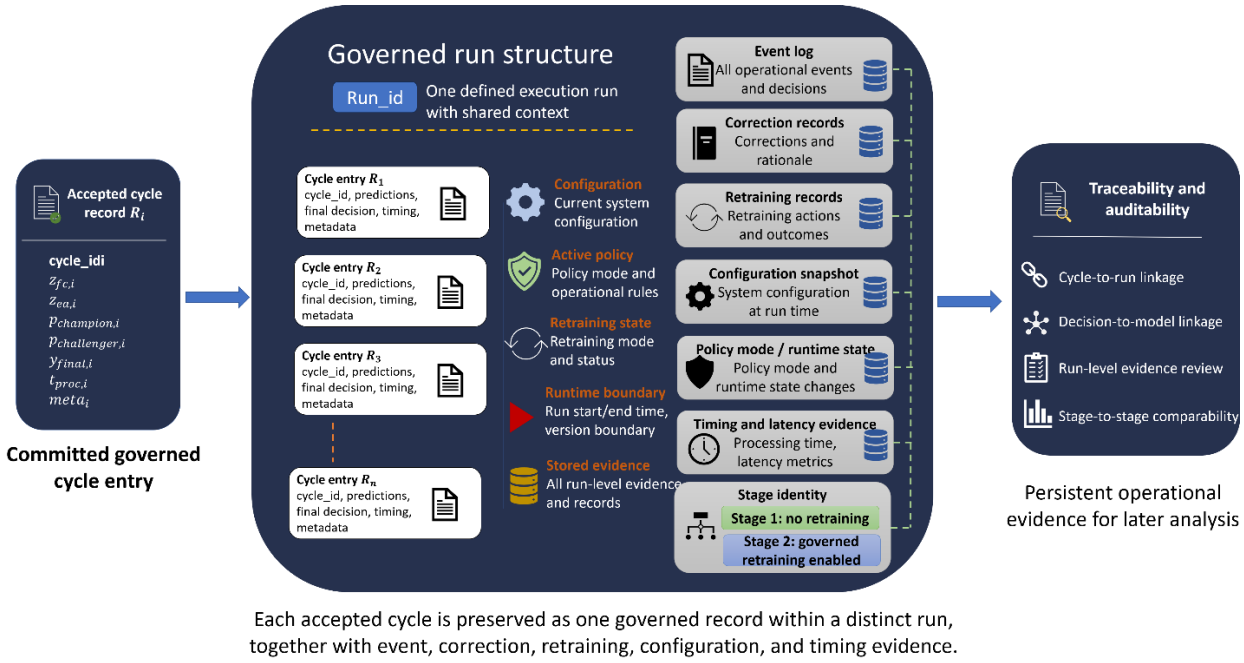


Figure 39. Run-level persistence and evidence structure of the governed real-time platform, showing cycle records, event logs, correction records, retraining records, and configuration snapshots within one governed run.

A key requirement of this layer is traceability. Traceability means that a later reviewer must be able to follow the path from the stored run record back to the accepted cycle and, from there, to the specific model and policy conditions that produced the deployed decision. In practical terms, this requires that the system preserve the cycle identifier, the active run identity, the predictive outputs, the decision outputs, and the important runtime conditions associated with the record. The persistence layer, therefore, ensures that the live decision is not detached from its origin. It remains tied to the exact production cycle and the exact runtime context in which it was produced.

The traceability structure links each committed cycle record to its governed run context and associated evidence. If G_{run} denotes the governed run context, the complete traceable record for cycle i is represented as:

$$T_i = [G_{\text{run}}, R_i, E_i, \text{Corr}_i, U_i]$$

Equation 28. Complete traceability record

where R_i is the committed operational cycle record, E_i is the associated event history, Corr_i contains any human-reviewed correction associated with the cycle, and U_i is the corresponding update-related record. This representation does not prescribe a specific database schema; it defines the evidence components required to reconstruct the decision and its governance context. A cycle decision becomes fully interpretable only when it can be read together with the events and controls that surrounded it.

The evidence structure, therefore, extends beyond the main cycle records. It also includes event logs, which preserve important runtime occurrences during execution. These may include operational state changes, monitoring triggers, stage transitions, or other events relevant to the interpretation of the live run. They provide a chronological layer that helps explain what was happening around the cycle decisions at different points in time. Without such logs, the cycle records would still exist, but the behavioural context of the run would be much harder to reconstruct.

A further component of the evidence structure is the preservation of human correction records where they are applicable. This is especially important in the second operational stage, where governed retraining is evaluated. Human corrections are not treated as detached annotations outside the system. They are preserved as part of the run evidence so that the relationship between automated prediction and reviewed intervention can be examined directly within the same operational history. This allows the platform to support later interpretation of how controlled adaptation was enabled and under what reviewed conditions it was allowed to proceed.

The persistence layer also preserves retraining records when governed retraining is activated. These records capture the fact that a retraining event occurred, the point in the run at which it occurred, and the operational context associated with it. Their value is not limited to proving that an update took place. They also make it possible to interpret later system behaviour in relation to the accepted model transition. In a controlled adaptation setting, this is essential. Without a dedicated retraining record, the platform would show a change in behaviour without

preserving a formal account of when and why the adaptive event occurred. The main components of this persistent evidence structure are summarised in Table 28.

Table 28. Main persistent evidence components stored by the governed real-time platform and the interpretive purpose of each component.

Persistent evidence component	Stored content	Interpretive purpose
Cycle records	Cycle identifiers, model outputs, final decisions, timing values, and linked runtime fields for each accepted cycle.	Preserves the per-cycle operational history of the governed real-time system in traceable form.
Run identity and run context	Run identifier, stage identity, runtime boundary, and stage-level operating context.	Ensures that each cycle record is interpreted within the correct governed run rather than as an isolated event.
Event logs	Chronological records of important runtime events, state changes, triggers, and operational occurrences.	Provides the behavioural context needed to understand what was happening around stored cycle decisions.
Human correction records	Reviewed intervention records linked to the relevant run and cycle context.	Supports later interpretation of how reviewed human input entered the governed operational history.
Retraining records	Accepted retraining events, their timing, and the associated governed update context.	Allows later system behaviour to be interpreted in relation to controlled model transition events.
Configuration snapshots and metadata	Policy mode, thresholds, runtime settings, and related configuration state.	Ensures that stored outcomes can be read together with the conditions under which they were produced.
Run-level summaries and exported evidence	Run summaries, model summaries, phase summaries, timing summaries, cost summaries, and evidence packages.	Connects detailed cycle-level history to higher-level interpretation of the operational stage as a whole.

Another important aspect of this section is the preservation of configuration state and runtime metadata. A stored prediction has limited value if the later reader cannot tell under which policy mode, threshold conditions, or operational stage it was produced. The persistence layer, therefore, preserves the relevant configuration snapshot and associated metadata within the same run evidence package. This allows the live behaviour of the platform to be interpreted in relation to its active settings rather than being read as if it were generated under unknown conditions. In a thesis concerned with governed operation, this is

especially important because the meaning of the results depends directly on the deployed rules and runtime mode.

The run-level structure also supports the production of summary outputs that describe the behaviour of the stage as a whole. These may include run summaries, model summaries, timing summaries, phase summaries, cost summaries, and retraining reports. Although these summaries are not replacements for the detailed per-cycle records, they provide an important higher-level view of what occurred during the run. Their presence inside the same evidence structure strengthens the interpretability of the platform because they connect detailed cycle behaviour to stage-level patterns without leaving the governed runtime context.

7.7 Live monitoring and operator-facing decision support

A governed real-time platform must not only execute decisions correctly but also make its runtime state visible during operation. For that reason, the system developed in this work includes a live monitoring layer that operates alongside the acquisition, inference, and persistence layers. The purpose of this monitoring layer is to provide runtime visibility of the evolving production process, the predictive behaviour of the models, and the operational condition of the platform itself. This is important because a live manufacturing environment requires more than correct background computation. It requires a transparent operational view through which system behaviour can be observed, interpreted, and, where necessary, reviewed while the run is still active.

The monitoring layer is therefore not treated as a cosmetic dashboard. It is an operational support function of the governed real-time architecture. It presents the live computational state in a form that remains meaningful to both technical review and production oversight. The monitored outputs include the evolving process behaviour of the accepted cycles, the prediction state of the Champion and Challenger paths, the deployed Final decision, the timing characteristics of the runtime path, and the key events that indicate changes in system status. In this way, the monitoring layer allows the governed real-time system to remain observable during operation rather than functioning as an internal process whose actions can only be examined after the run has ended.

One important function of the monitoring layer is the display of the live process state in relation to the cycle. Because the architecture is organised around cycle-

level execution, the monitoring view must reflect the cycle structure rather than reducing the process to isolated status indicators. The system therefore supports visualisation of cycle-resolved process behaviour, allowing the accepted cycle to be viewed as a coherent production event. This helps preserve the connection between the physical manufacturing process and its cycle-level digital representation. The operator or reviewer is able to see that the system is not generating decisions from detached statistical abstractions, but from a live process whose pressure, thermal, and motion behaviour remain visible in relation to the cycle boundary and the selected analytical representation.

A second major function of the monitoring layer is the presentation of runtime predictive visibility. During runtime execution, the platform produces outputs from the Champion and Challenger paths and resolves them through the Final policy. Monitoring these outputs is valuable because it makes the predictive structure of the platform visible while the run is progressing. This is especially important in the first operational stage, where the purpose is to examine how the Challenger behaves against the fixed Champion under stable live conditions. By making both paths visible in runtime, the system allows the relationship between the two models to be observed as part of the active operational state rather than only as a later static summary.

Let M_i denote the live monitoring state associated with cycle i. The monitored state is represented as:

$$M_i = \left[\text{cycle_id}_i, \text{process_view}_i, p_{\text{champion},i}, p_{\text{challenger},i}, y_{\text{final},i}, \text{status}_i\right]$$

Equation 29. Live monitoring state

where cycle_id_i identifies the accepted cycle, process_view_i denotes the displayed process information, $p_{\text{champion},i}$ and $p_{\text{challenger},i}$ are the Champion and Challenger probability outputs, $y_{\text{final},i}$ is the deployed decision, and status_i is the runtime status presented by the monitoring layer. This representation specifies the monitored information rather than prescribing a particular user-interface layout. This expression is not intended to define a particular user interface format. It simply makes clear that the monitoring layer combines process state, prediction state, and operational status in one live view.

A third major function of the monitoring layer is the display of runtime timing behaviour. Because the system is intended for live cycle-level use, it is necessary

to observe whether the computational path remains compatible with the cycle rhythm of the physical process. Runtime latency, therefore, becomes a directly relevant operational characteristic. The platform records the processing time associated with cycle handling and uses the monitoring layer to make the timing behaviour visible at the stage level. This allows the operator or reviewer to see whether the live computational path remains within a practical operating range while the run is active. In the final operational stages, timing visibility was particularly important because the platform had to preserve reliable cycle capture while maintaining acceptable cycle-level processing behaviour.

Let $t_{\mathrm{proc},i}$ denote the processing latency associated with cycle i. For a run containing N accepted cycles, the ordered processing-latency sequence is defined as:

$$\mathbf{t}_{\mathrm{proc}} = \left[t_{\mathrm{proc},1}, t_{\mathrm{proc},2}, \ldots, t_{\mathrm{proc},N}\right]$$

Equation 30. Run-level processing-latency sequence

where N is the number of accepted cycles in the run. This sequence supports the calculation of run-level timing summaries, including the mean and 95th-percentile latency. Timing is therefore treated as an observable operational characteristic rather than as a hidden implementation detail. The relationship between the user and the live computational state is illustrated in Figure 40.

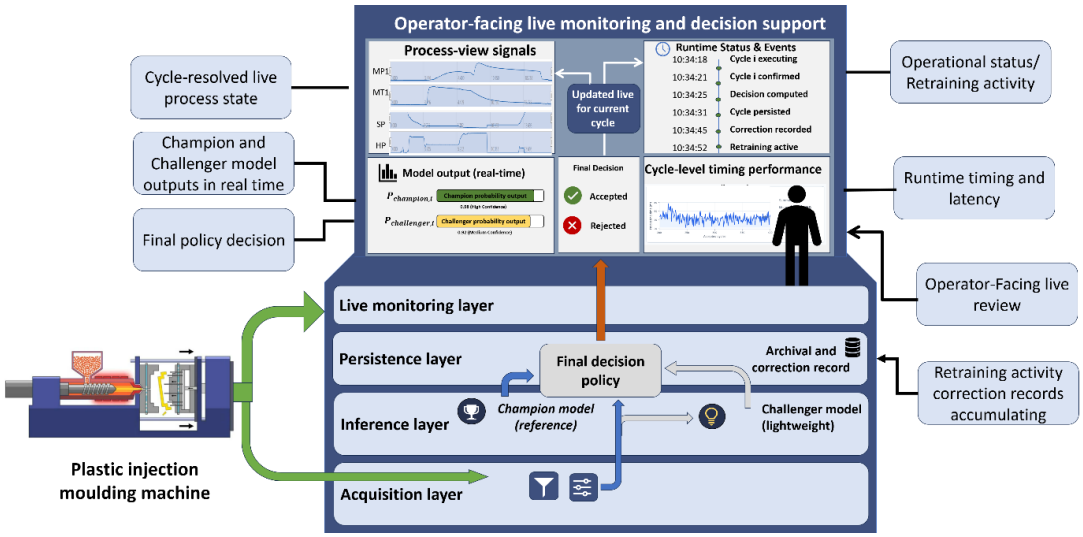


Figure 40. Live monitoring view of the governed real-time platform, showing runtime visibility of process behaviour, prediction state, and timing performance.

The monitoring layer also plays an important role in exposing runtime events and system state changes. During live operation, the platform may encounter status transitions related to cycle execution, stage progression, correction input, or retraining activity. These transitions are not left hidden in the background. The monitoring layer allows them to be presented as part of the live operational condition of the platform. This makes the governed real-time platform easier to interpret because the reviewer can see not only the cycle decisions themselves, but also the status changes that may affect the later interpretation of the run.

This is particularly relevant in the second operational stage, where governed retraining is enabled. In that stage, live monitoring is not limited to normal cycle execution. It also becomes part of the visibility mechanism through which the platform shows that the system remains under controlled operation while correction records accumulate and the retraining pathway becomes active. The monitoring layer supports confidence that the adaptive stage is not occurring as a hidden background change. It remains visible as part of the governed runtime state of the platform.

Another important contribution of live monitoring is support for operator-facing review. The purpose of the architecture is not to remove human interpretation entirely, especially when the platform is operating in a governed adaptation setting. Instead, the platform is designed to maintain a live view that helps the human reviewer understand what the system is doing, how the cycle decisions are evolving, and whether the operational condition remains stable. This is a significant practical requirement because a live manufacturing system must remain interpretable to the people supervising it. Runtime visibility, therefore, supports not only technical analysis but also responsible deployment practice. The main outputs and operational values of this monitoring layer are summarised in Table 29.

Table 29. Main monitored outputs of the governed real-time platform and the operational value of each during runtime execution.

Monitored output	What is monitored	Operational value during live execution
Cycle-resolved process view	Live or recently accepted cycle behaviour in the main process variables.	Maintains visibility of the physical production event and preserves the link between machine behaviour and digital interpretation.
Champion prediction state	Runtime probability or decision output of the fixed Champion path.	Provides a stable benchmark view against which the lighter operational path can be interpreted.
Challenger prediction state	Runtime probability or decision output of the Challenger path.	Supports live comparison with the Champion and reveals the behaviour of the lighter operational route during the run.
Final deployed decision	The policy-resolved operational decision was released for the accepted cycle.	Makes the deployed manufacturing decision visible rather than leaving it hidden behind raw model outputs.
Runtime timing behaviour	Cycle-level processing time and stage-level timing summaries.	Shows whether the live computational path remains compatible with the rhythm of the physical process.
Operational status and events	State changes, stage progression, correction activity, and retraining-related status.	Provides context for interpreting what the platform is doing while the run is active.
Adaptive-stage visibility	Reviewed correction accumulation and retraining-path activity during governed adaptation.	Supports confidence that adaptation remains visible and supervised rather than hidden in the background.

The relation between monitoring and persistence is important, but the two layers are not identical. Persistence stores the full runtime evidence so that behaviour can be studied after the run. Monitoring, by contrast, exposes the current operational state while the run is still in progress. The governed real-time architecture requires both. Without persistence, the system would be difficult to audit later. Without monitoring, the system would remain difficult to interpret while it is operating. The combination of the two layers allows the platform to remain both observable and traceable.

The monitoring layer also supports the methodological distinction between the two operational stages. In the first stage, where runtime retraining is disabled, the live view helps examine the relationship between the Challenger and the fixed Champion under stable governed operation. In the second stage, where governed retraining is enabled, the same monitoring layer provides continuity while also exposing the active adaptation context. This continuity is valuable because it allows the second stage to be interpreted as an extension of the same operational platform rather than as a separate experimental system.

7.8 Drift awareness, human oversight, and governed retraining

A governed real-time system intended for manufacturing use cannot assume that the operating conditions observed at the start of a run will remain unchanged indefinitely. Process behaviour may shift, data balance may change, and the relationship between the incoming cycle state and the trained model response may gradually move away from the conditions under which the active model path was last calibrated. For that reason, the governed real-time platform developed in this work includes an explicit layer for drift awareness and controlled adaptation. Its role is not to allow unrestricted online learning. Its role is to detect when the live operating state is changing, preserve visibility of that change, allow human review to enter the decision path where required, and permit retraining only when the predefined governance conditions have been satisfied.

does not treat the live model path as if it were operating in a permanently stable environment. Instead, it monitors the evolving behaviour of the live stage and preserves indicators that can reveal whether the current operating state is moving away from the earlier expected condition. This function is important because it creates an intermediate position between fixed deployment and uncontrolled adaptation. The system is therefore able to remain stable in normal operation while still recognising that the incoming live context may not remain identical across the full duration of the stage.

Drift awareness operates as an observational mechanism and does not independently trigger model retraining. Let d_i denote the drift-monitoring state recorded for cycle i. For a stage containing N accepted cycles, the ordered drift-state sequence is defined as:

$$\mathbf{d} = [d_1, d_2, \dots, d_i, \dots, d_N]$$

Equation 31. Ordered drift-monitoring state sequence

where N is the number of accepted cycles in the operational stage. The sequence $\mathbf{d}$ preserves the progression of the monitored drift state throughout the run, allowing changes in operating behaviour to be reviewed in their correct cycle order.

The presence of drift awareness does not mean that the system is designed to retrain automatically whenever any variation is detected. That would be too weak from a governance perspective and too risky in a live production setting. Instead, the platform separates awareness from action. The detection or observation of changed behaviour is treated as a reason for closer review, not as a direct instruction to overwrite the active operational path. This distinction is essential to the logic of the chapter because it preserves the difference between an intelligent live system and an uncontrolled self-modifying system. This governed pathway is visualised in Figure 41.

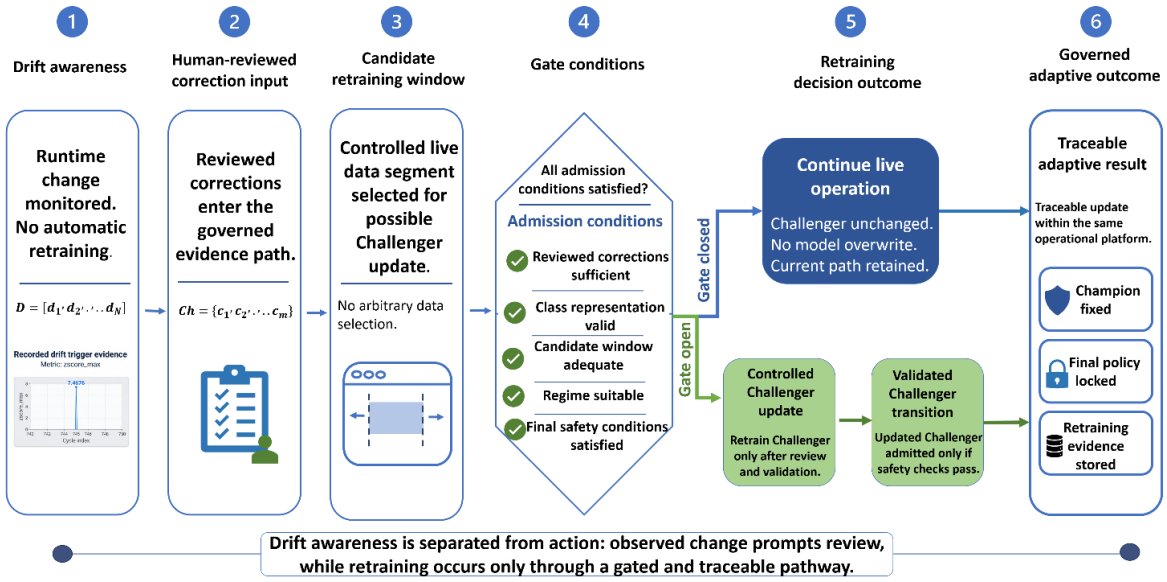


Figure 41. Drift awareness and governed retraining pathway within the governed real-time platform, showing monitored change, human review, gated retraining, and controlled model update.

The second part of this layer is human oversight. Human oversight is necessary because the platform is designed for governed industrial use rather than unrestricted autonomous self-adjustment. When the live stage enters a condition in which adaptation may become relevant, the system preserves a path for introducing reviewed corrections and controlled judgment. This means that the governed real-time platform remains operationally supervised even when the adaptive branch is active. The live platform, therefore, does not treat model updating as a purely internal machine process. It preserves the role of the reviewed intervention as part of the adaptation logic.

Human-reviewed corrections are retained as governed runtime evidence. Let $\mathcal{C}_{\mathrm{h}}$ denote the set of reviewed corrections accumulated within the relevant operational-stage window. This set is defined as:

$$\mathcal{C}_{\mathrm{h}} = \{c_1, c_2, \dots, c_m\}$$

Equation 32. Set of human-reviewed corrections

where c_j is the j-th reviewed correction entry and m is the number of reviewed corrections accumulated within the relevant stage window. These corrections are retained within the governed evidence structure rather than being treated as informal adjustments outside the platform. They become part of the operational evidence used when the retraining pathway is being considered.

The retraining gate forms the boundary between adaptive readiness and permission to update the Challenger. Drift observation or the availability of reviewed corrections does not independently authorise retraining. Let G_{retrain} denote the retraining-admission decision at a given review point. The general gate is defined as:

$$G_{\mathrm{retrain}} = \begin{cases} 1, & \text{if all governed retraining conditions are satisfied} \\ 0, & \text{otherwise} \end{cases}$$

Equation 33. General retraining-admission gate

The update path remains closed when $G_{\mathrm{retrain}} = 0$ and becomes eligible to proceed only when $G_{\mathrm{retrain}} = 1$. This binary gate ensures that model adaptation remains conditional, traceable and governed.

The general gate in Equation 33 can be expressed as the conjunction of four validation conditions:

$$G_{\mathrm{retrain}} = \mathbb{I}(H_{\mathrm{valid}})\mathbb{I}(C_{\mathrm{valid}})\mathbb{I}(B_{\mathrm{valid}})\mathbb{I}(R_{\mathrm{valid}})$$

Equation 34. Conjunctive retraining-eligibility gate

where $\mathbb{I}(\cdot)$ is an indicator function that equals 1 when its stated condition is satisfied and 0 otherwise. The four validation conditions are:

H_{valid}: validity of the human-reviewed correction requirement;

C_{valid}: validity of the candidate retraining-data window;

B_{valid}: validity of the class-balance and data-adequacy requirements; and

R_{valid}: validity of the remaining safety and governance requirements.

Therefore, $G_{\text{retrain}} = 1$ only when all four conditions are satisfied. If any indicator equals 0, the product becomes 0 and the Challenger update remains blocked.

In the implemented governed retraining path, retraining was therefore permitted only when all admission conditions were satisfied within the supervised adaptive branch of the platform. This gated structure is what distinguishes the adaptive layer of the platform from a conventional online learning loop. The platform does not move directly from detected change to retraining. Retraining must pass through an explicit review and acceptance boundary. This is especially important because the second operational stage of the study is not intended merely to show that a model can be retrained during runtime. Its purpose is to show that retraining can occur correctly, that is, under controlled conditions, without breaking the live governed structure established in the earlier sections.

In the second operational stage, this adaptive path was activated in a controlled form. The retraining event was not introduced as a free automatic overwrite of the live model path. Operationally, this means that the platform first admits retraining through governed review conditions and then confirms final training-set safety before the Challenger model state is allowed to change. The retraining transaction occurred only after the required reviewed and governed conditions had been satisfied within the same governed real-time platform. The significance of this stage is therefore not simply that retraining was attempted, but that the governed retraining pathway completed successfully and produced a new versioned Challenger model artefact while preserving the cycle-execution discipline, policy control, and run-level traceability defined earlier in the chapter. The preserved implementation evidence does not conclusively establish that the running inference process loaded the replacement estimator for subsequent cycles. Therefore, this stage is interpreted as evidence of governed retraining execution and artefact generation rather than as prospective performance evidence for the newly trained model.

Let $M^{\text{new}}_{\text{challenger}}(\cdot)$ denote the updated Challenger mapping. The governed model transition is permitted only after the retraining gate has been satisfied:

$$M_{\text{challenger}}(\cdot) \longrightarrow M^{\text{new}}_{\text{challenger}}(\cdot) \qquad \text{if} \qquad G_{\text{retrain}} = 1$$

Equation 35. Governed Challenger model transition

Equation 35 makes clear that the Challenger does not change continuously or implicitly. The transition to $M_{\text{challenger}}^{\text{new}}(\cdot)$ occurs only after gate acceptance. The Champion remains unchanged and continues to provide the fixed reference path.

The fixed role of the Champion remains especially important in this section. Even when the retraining branch is enabled, the Champion does not change. It continues to provide the stable full-curve reference path throughout the stage. This stability is necessary because it preserves a constant analytical anchor while the adaptive route is being evaluated. The architecture therefore avoids the risk of losing all reference structure during governed model adaptation. One path remains fixed, while the other is permitted to change only under governance. This separation supports both interpretability and safety.

Human oversight also remains relevant after the retraining event itself. The significance of the reviewed correction pathway is not exhausted once the gate has been satisfied. It remains part of the interpretive basis on which the adaptive stage is understood. The governed real-time platform therefore preserves not only the fact that retraining occurred, but also the reviewed context under which it occurred. This makes the resulting model transition technically accountable. The update can be examined not only as a change in model state, but as a governed action tied to visible operational conditions. The main elements of this governed retraining pathway are summarised in Table 30.

Table 30. Main elements of the governed retraining pathway in the governed real-time platform, including drift awareness, human review, gate conditions, and controlled Challenger updating.

Element of the governed retraining pathway	Function in the platform	Governance value
Drift awareness	Monitors whether the live operating state is moving away from the earlier expected condition.	Creates awareness of change without triggering uncontrolled automatic updating.
Human-reviewed correction input	Preserves reviewed corrections as governed by operational evidence within the run.	Ensures that adaptation remains supervised rather than purely internal to the model path.
Candidate retraining window	Defines the live data segment considered for possible Challenger updating.	Prevents arbitrary or uncontrolled data selection for model change.
Gate conditions	Checks reviewed correction evidence, candidate-window adequacy, class representation and composition requirements, regime suitability, and final safety conditions before Challenger update.	Keeps the update path closed unless all required conditions are satisfied.

Element of the governed retraining pathway	Function in the platform	Governance value
Controlled Challenger update	Permits a model transition only after the governed gate has been accepted.	Ensures that adaptation occurs as a formal, managed action rather than an unrestricted overwrite.
Fixed Champion reference	Maintains an unchanged full-curve benchmark while the Challenger path is being considered for update.	Preserves interpretability and reference stability during the adaptive stage.

The relationship between the two operational stages becomes especially clear at this point. In the first stage, runtime retraining is disabled, so the platform operates as a stable, governed inference environment in which the Challenger can be examined against the fixed Champion under live conditions. Drift awareness may remain relevant as part of the monitored operational state, but the update path is not open. In the second stage, the same platform retains its existing cycle execution, model structure, final policy, and persistence design, but the governed retraining branch is now enabled. This means that the second stage is not a different system. This means that the second stage is not a different system. It is the same governed real-time platform operating with the retraining pathway enabled. That continuity is critical to the thesis logic because it allows the adaptive stage to be interpreted as a controlled extension of the stable live stage rather than as an unrelated experiment.

7.9 Deployment Robustness, Safeguards, and Runtime Stability

A governed real-time platform intended for manufacturing use must remain reliable not only when conditions are favourable, but also when the operating environment becomes imperfect. For that reason, the live system developed in this study includes a set of runtime safeguards designed to preserve cycle integrity, decision integrity, and operational continuity during execution. These safeguards are not separate from the platform architecture. They are embedded within it because the value of a live decision system depends not only on prediction quality, but also on whether the system can continue to operate in a stable, interpretable, and controlled manner while the physical process is running.

The first requirement of robustness is the protection of cycle integrity. Since the platform is built around one accepted cycle as the fundamental decision unit, any failure to preserve the uniqueness and completeness of that cycle immediately weakens the credibility of the live system. A robust runtime design must therefore prevent incomplete cycles from entering the inference layer, prevent the same cycle from being committed more than once, and prevent a missed cycle from silently disappearing from the operational history. This is why the earlier acquisition discipline is also a safeguard. The buffering, confirmation, snapshot, and reset sequence is not only a method of data handling. It is the first line of protection against invalid live decisions.

The accepted-cycle condition combines completeness and uniqueness. A cycle may enter the downstream decision path only when machine-confirmed completion has been received and the cycle has not already been committed. Let V_i denote the acceptance-validity condition for cycle i. The rule is defined as:

$$V_i = \begin{cases} 1, & \text{Complete}_i = 1 \ \wedge \ \text{Committed}_i = 0 \\ 0, & \text{otherwise} \end{cases}$$

Equation 36. Accepted-cycle validity condition

where $\text{Complete}_i = 1$ indicates that machine-confirmed cycle completion has been received, and $\text{Committed}_i = 0$ indicates that the cycle has not previously been stored. Therefore, $V_i = 1$ enforces the one-cycle, one-commit rule and prevents incomplete or duplicate cycle records from entering the decision path.

A second requirement is the protection of decision integrity. Even if the cycle is captured correctly, the governed real-time architecture still requires safeguards to ensure that the produced decision remains governed. For this reason, neither the Champion nor the Challenger is allowed to bypass the locked Final policy. The deployed output is always generated inside the controlled decision layer described earlier. This architectural separation acts as a safeguard because it prevents a direct raw model output from becoming the plant-facing decision without passing through the agreed deployment rule. In effect, the final policy is not only a decision structure. It is also a runtime protection layer.

A third requirement is the protection of reference stability. The platform achieves this by keeping the Champion fixed throughout both operational stages. This safeguard is important because it ensures that live comparison and controlled

adaptation do not destroy the baseline analytical path. Even when the Challenger is examined as the lighter operational model, and even when retraining is enabled in the second stage, the system retains a stable full-curve reference route. This prevents the platform from becoming fully self-modifying and preserves a permanent benchmark within the live environment. The fixed Champion acts as a structural safeguard as well as a modelling reference.

The retraining branch is protected by the binary admission gate defined in Equation 33 and the conjunctive validation conditions defined in Equation 34. When $G_{\text{retrain}} = 0$, the model-update path remains closed. When $G_{\text{retrain}} = 1$, the governed Challenger transition defined in Equation 35 may proceed. Consequently, neither observed drift nor the existence of a candidate dataset can independently authorise a model update. This gate protects the live platform against uncontrolled adaptation. The principal runtime safeguards are illustrated in Figure 42.

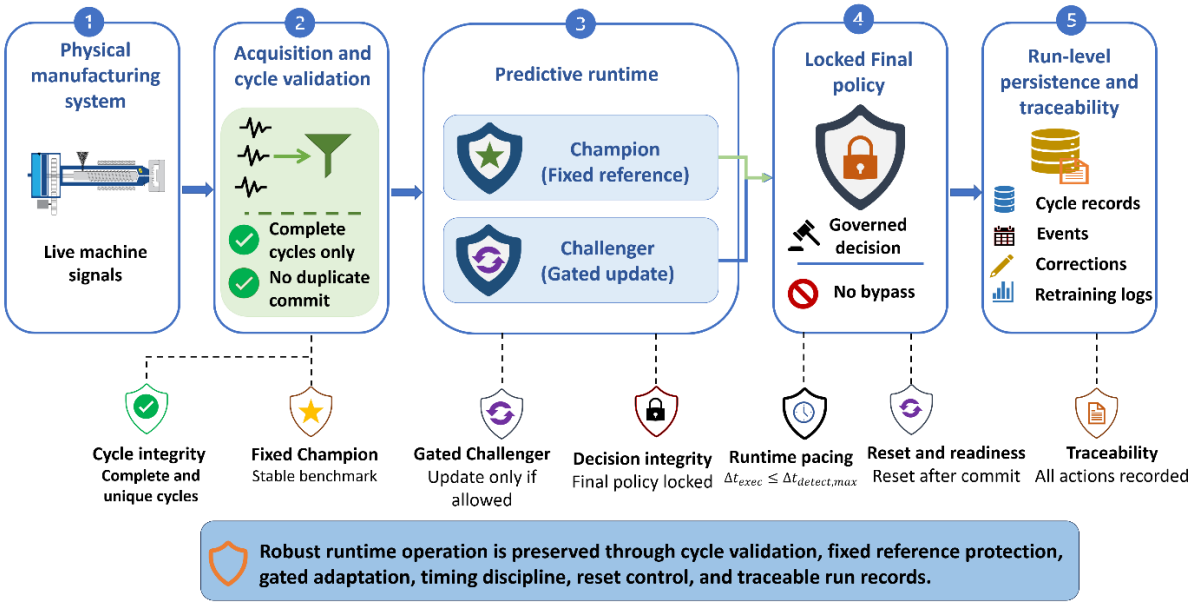


Figure 42. Main runtime safeguards of the governed real-time platform, showing cycle validation, fixed reference protection, gated adaptation, and preserved policy control.

Another important aspect of runtime robustness is timing stability. In a live manufacturing environment, the computational path must operate in step with the physical process. A decision system that is analytically correct but mistimed can still fail in practice if it misses cycle-completion signals or cannot process the accepted cycle within a practical operational window. For that reason, timing behaviour is treated as a live operational concern, not as a secondary

performance statistic. The platform, therefore, preserves processing-time visibility and adopts an execution pace that protects reliable acquisition as well as practical cycle-level inference.

This became especially important in the final governed stages because the cycle-completion signal could be brief and therefore vulnerable to overly aggressive polling or execution timing. To address this, the runtime behaviour was adjusted according to a reliability-first pacing principle. The purpose of this adjustment was not to make the system artificially slower. Its purpose was to align the runtime platform with the timing behaviour of the machine-data interface so that cycle completion could be captured correctly and the operational record could remain complete. This is an important engineering outcome because it shows that runtime robustness depends on respecting the physical and communication characteristics of the manufacturing environment, not simply on computational speed.

Let Δt_{exec} denote the effective runtime execution interval and $\Delta t_{\text{detect,max}}$ the maximum interval that still permits reliable observation of the machine-reported cycle-completion transition. The runtime pacing constraint is:

$$\Delta t_{\text{exec}} \leq \Delta t_{\text{detect,max}}$$

Equation 37. Runtime pacing constraint for reliable signal capture

Equation 37 requires the execution interval to remain sufficiently short to observe the required completion signal reliably. If Δt_{exec} exceeds $\Delta t_{\text{detect,max}}$, the platform may fail to observe the machine-reported transition and consequently compromise cycle accounting. The pacing constraint is therefore a correctness safeguard rather than merely a performance preference.

A further safeguard concerns operational continuity after each committed cycle. Once a cycle has been accepted, evaluated and stored, the execution sequence proceeds to state q_4, which performs the buffer and runtime-state reset defined in Equation 20. The platform then returns to the waiting condition associated with q_0 before acquisition of the next physical cycle begins. This controlled reset prevents temporary buffers, stale values or incomplete state transitions from affecting the subsequent cycle.

The safeguard structure also protects the platform from interpretive instability. A live system can be technically active but still be difficult to trust if its behaviour cannot be explained afterwards. For that reason, the persistence and evidence structure described earlier also functions as a robustness mechanism. By preserving cycle records, events, correction entries, retraining records, and configuration state, the platform protects itself from becoming an opaque black box. If an unexpected result appears during execution, the run history remains available for review. This ability to reconstruct what happened is a practical safeguard because it reduces the operational risk of untraceable behaviour. The complete set of these runtime safeguards is summarised in Table 31.

Table 31. Main safeguards used to preserve runtime stability, decision control, and operational reliability in the governed real-time platform.

Safeguard	Protective function	Operational effect
Cycle validation rule	Allows a cycle to enter the decision path only when it is complete and not already committed.	Protects one-cycle, one-commit discipline and prevents incomplete or duplicate records.
Locked Final policy	Prevents raw model output from bypassing the governed decision layer.	Maintains decision control even when Champion and Challenger differ.
Fixed Champion reference path	Keeps the benchmark analytical route unchanged across both stages.	Preserves reference stability while the lighter path is examined or updated.
Gated retraining	Keeps the adaptive branch closed unless governance conditions are satisfied.	Prevents uncontrolled live model change.
Reliability-first runtime pacing	Keeps execution aligned with the signal-detection requirements of the machine interface.	Reduces the risk of missed cycle-completion signals and unstable live capture.
Reset and readiness discipline.	Returns the platform to a clean post-cycle state before the next cycle begins.	Prevents stale state or buffer carry-over from contaminating the following cycle.
Run-level persistence and traceability	Preserves cycle records, events, corrections, retraining records, and configuration state.	Protects the platform from becoming opaque or operationally untraceable.

The meaning of this section differs slightly across the two operational stages, but the safeguard structure remains present in both. In the first stage, where runtime retraining is disabled, the safeguards maintain stable governed live operation so that the system can be examined under fixed conditions and the relationship between the Challenger and the Champion can be observed without model change. In the second stage, the same safeguards remain in force while governed

retraining is enabled. This is important because it shows that adaptation is added inside the same protected runtime structure rather than by removing the safeguards established in the stable stage. The platform, therefore, supports a stronger claim: it can accommodate controlled adaptation while retaining the same protective architecture that governs ordinary live operation.

7.10 Summary

This chapter has defined the implementation basis of the governed real-time platform developed in this research and has shown how the predictive capability established earlier is carried into governed runtime operation. The chapter began by framing the platform as a cycle-resolved operational layer rather than as a simple online classifier. It then explained how the live system captures the physical process at the cycle level, how each accepted cycle is converted into valid analytical representations, how the fixed Champion and governed Challenger operate in parallel, how the deployed decision is produced through the locked Final policy, and how the resulting operational state is preserved through run-level persistence, monitoring, and controlled adaptation. Taken together, these elements establish the architecture as a governed manufacturing monitoring and decision-support environment rather than as a standalone modelling tool.

A central outcome of the chapter is that the real-time platform is governed at every critical stage of operation. The cycle must first be captured correctly. The accepted cycle must then be converted into a valid analytical form. The Champion and Challenger are allowed to produce model outputs, but those outputs do not become the deployed decision until they pass through the Final policy. The resulting cycle decision is then stored within the run-level evidence structure, where it remains linked to its operational context. When adaptation becomes relevant, the platform does not retrain automatically. It preserves drift awareness, allows human oversight to enter the process, and permits a Challenger update only after the required governance conditions have been satisfied. The chapter shows that live prediction, decision control, and model updating are all embedded inside one controlled operational structure.

The two operational stages used in the study are also clarified by the chapter architecture. The first stage operates with runtime retraining disabled so that the governed real-time platform can be examined under stable operating conditions. This stage is used to assess how the lighter Challenger behaves relative to the

fixed Champion while the deployment rule remains unchanged. The second stage retains the same platform architecture but activates governed retraining so that the adaptive behaviour of the system can be evaluated under controlled live conditions. This distinction is essential because it defines the methodological purpose of the later comparative analysis. The second stage is not simply a repeated run. It uses the same governed real-time platform, with the governed retraining pathway enabled.

The chapter has also established that live industrial credibility depends on more than raw model performance. Reliable cycle acquisition, correct cycle-boundary handling, stable timing behaviour, one-cycle one-commit discipline, preserved runtime evidence, and safe control of model change are all necessary conditions for a usable live system. This is particularly important in a manufacturing context, where a model that appears accurate in isolation may still be operationally unreliable if the platform loses cycles, mismanages timing, or cannot provide traceable evidence after execution. By defining the platform in these terms, the chapter places operational validity at the centre of the governed architecture design. This complete operational logic is visualised in Figure 43.

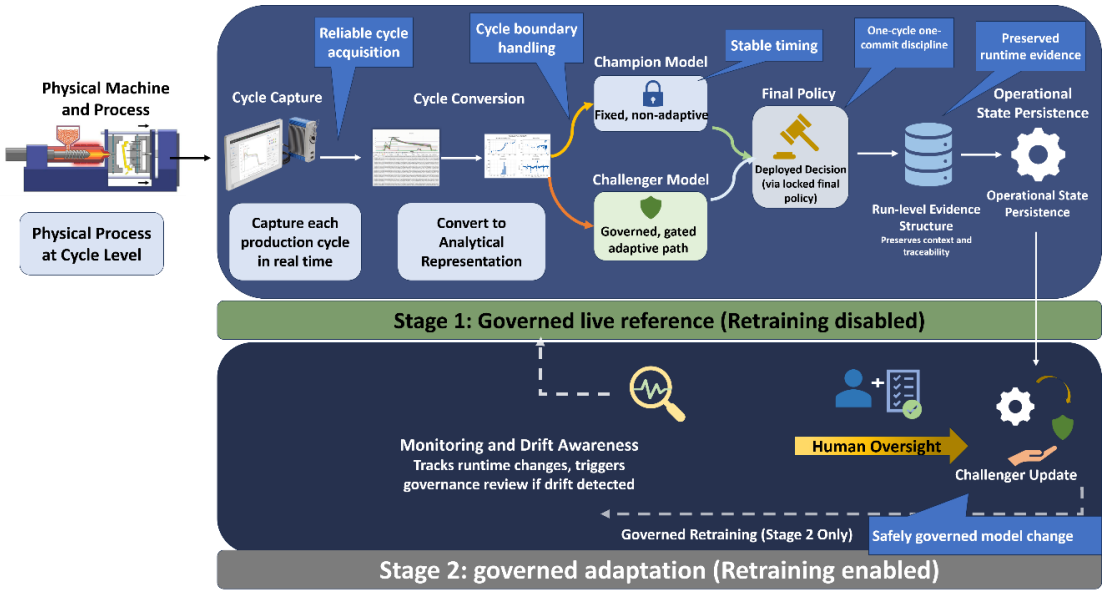


Figure 43. Overall logic of the governed real-time platform, showing the complete path from cycle-level process-data capture to governed runtime evaluation across the two operational stages.

This chapter establishes the implementation basis of the governed real-time architecture by integrating cycle handling, complementary Champion–Challenger

roles, explicit Final decision authority, drift-aware monitoring, human-supported evidence, controlled model adaptation, and traceable evidence storage within a common cycle-resolved structure. Its contribution lies in keeping prediction, decision release, monitoring, and model adaptation functionally distinct while preserving their connection through a shared cycle identity. In doing so, the chapter addresses Research Question 3 and Objective 3 by establishing the integrated, governed real-time architecture, while providing the implementation basis for evaluating Research Question 4 and Objective 4 through the two operational stages examined in Chapter 8.

The two operational stages provide the basis for the subsequent evaluation. The first represents fixed-condition live operation, in which the Champion and Challenger are compared without runtime model updating, while the second uses the same platform with governed Challenger retraining enabled. The following chapter therefore evaluates the measured behaviour of these stages, including predictive performance, model disagreement, Final-policy effects, timing, drift behaviour, and governed retraining. Because the architecture and stage definitions have been established here, these results can be interpreted within a clearly defined operational context rather than as isolated numerical outcomes. The main implementation outcomes supporting this evaluation are summarised in Table 32.

Table 32. Main implementation outcomes established in Chapter 7 and their role in supporting the measured evaluation presented in Chapter 8.

Implementation outcome established in Chapter 7	Role for Chapter 8 evaluation
Cycle-resolved governed real-time architecture	Provides the operational definition needed to interpret measured live behaviour at the cycle level.
Reliable live acquisition and cycle execution discipline	Ensures that later measured results can be read as valid cycle-level evidence rather than unstable capture artefacts.
Dual representation and dual-model structure	Supports later comparison of the fixed Champion and lighter Challenger under the same live conditions.
Locked Final policy layer	Allows later results to distinguish raw model behaviour from deployed operational release.
Run-level persistence and evidence structure	Provides the stored operational record on which comparative measured analysis can be based.
Live monitoring and operator-facing visibility	Supports interpretation of runtime behaviour as an observable operational state rather than a hidden process.

Implementation outcome established in Chapter 7	Role for Chapter 8 evaluation
Governed retraining pathway	Makes it possible to evaluate controlled adaptation as part of the same live platform rather than as a separate system.
Runtime safeguards and stability controls	Ensures that later numerical findings are interpreted within a protected and operationally credible live environment.

With the governed real-time platform now defined as a manufacturing monitoring and decision-support system, the thesis can proceed to the detailed examination of how that system performed across the two operational stages.

Chapter 8
Live Evaluation and Discussion of the Governed Cycle-Level Architecture

8.1 Introduction

The previous chapter defined the governed real-time architecture as a cycle-level manufacturing monitoring and decision-support system rather than as a standalone prediction tool. It established the operational structure through which cycle data are acquired, validated, represented, evaluated by the two predictive paths, resolved through a locked final policy, and preserved as traceable run-level evidence. The purpose of the present chapter is different. This chapter examines the measured behaviour of that governed system under live operating conditions and interprets what the recorded evidence shows when the architecture is executed in practice. The emphasis, therefore, moves from implementation logic to results and discussion.

The live-run records evaluated in this chapter should not be read as a replacement for the offline 903-cycle modelling dataset used in Chapter 6. They represent governed runtime evidence generated by the implemented real-time platform. Each live run contains the cycles that were accepted by the runtime capture logic, stored under the run record, and successfully linked to truth labels for evaluation. For this reason, the 850-cycle live-run evidence is used to assess deployment behaviour, timing, drift, policy effects, and governed retraining, whereas the 903-cycle dataset is used for offline model development and representation comparison.

The chapter is organised around two governed live runs, each serving a distinct evaluative purpose. The first is the baseline live run before runtime retraining, which provides the non-adaptive reference state. In this run, the governed real-time platform operated under fixed predictive conditions, allowing the relationship between the full-curve Champion, the lighter event-anchored Challenger, and the Final decision to be examined without the added complication of an in-run model update. The second is the retraining-enabled live run, in which the same governed platform was operated again with the adaptive branch active. This second run is not treated as a simple repetition of the first. Its purpose is to evaluate whether the same live system can support controlled adaptation correctly while preserving operational continuity, timing suitability, and decision traceability.

This two-run structure is methodologically important because the chapter is not intended to report model accuracy in isolation. The central objective is to

determine whether the governed live system behaves credibly as a cycle-level manufacturing decision layer. For that reason, the discussion is carried out at the system level rather than at the classifier level alone. The chapter considers the comparative behaviour of the fixed Champion and the lighter Challenger, the behaviour of the final deployed decision under the locked policy, the effect of disagreement between the two model paths, the visible evolution of drift across the run, the meaning of governed retraining in live operation, the timing burden of cycle-level inference, and the practical cost consequences of the resulting error structure.

The baseline run provides the essential starting point for that interpretation. It establishes how the two predictive paths behave when the live system is operating under fixed model conditions and before any runtime update is admitted. This is the clearest state in which to judge whether the lighter event-anchored path is merely viable or genuinely stronger than the fixed full-curve reference. The retraining-enabled run then extends that interpretation by moving into a more demanding operational condition in which the same governed platform remains live while controlled Challenger updating becomes active. The chapter uses the two runs not as unrelated experiments, but as two connected views of the same decision architecture: first under fixed live operation, then under governed adaptive operation.

The measured evidence also requires a deliberate separation between raw model behaviour, deployed decision behaviour, and adaptive behaviour. This distinction is central to the interpretation of the chapter. The Champion and Challenger are examined as two predictive paths with different representational and operational roles. The Champion remains the fixed full-curve reference, while the Challenger represents the lighter event-anchored route intended for efficient live use. However, the final deployed decision is not simply the output of whichever model appears strongest. It is produced through the locked final policy defined earlier. This means that the results cannot be interpreted as though the strongest classifier automatically becomes the operational decision. The chapter must therefore distinguish clearly between the performance of the raw predictive paths and the behaviour of the policy-controlled deployed output.

The evidence used in this chapter also shows that the meaning of the second run must be handled carefully. The retraining-enabled run included one successful governed Challenger retraining event at cycle 745, supported by reviewed correction input and completed within the live operating record. This is a major

result, but it should not be misread as meaning that the second run exists only to show aggregate improvement over the baseline. Its deeper purpose is to demonstrate that the platform can admit a controlled in-run update without breaking the governed execution structure. For that reason, the adaptive-stage evidence is interpreted not only in terms of classification outcomes but also in terms of whether the system preserved control, continuity, and traceability while the adaptive branch became active.

A second reason the chapter must be read at system level is that the contribution being evaluated is broader than predictive strength alone. Confusion matrices are used to show how the three decision paths behave in practical classification terms. ROC and precision-recall plots are used to assess ranking strength beyond one operating threshold. The alignment figure is used to connect the event-anchored route back to a physically meaningful cycle transformation. The drift timeline is used to interpret how the operating condition evolved across cycle progression. The latency evidence is used to test whether the live system remained compatible with real-time cycle-level use. The cost summaries are used to translate error patterns into a practical operational burden. These figures and tables are therefore not treated as decorative summaries. They are used as operational evidence that explains how the system behaved and why the resulting behaviour matters.

The chapter is also structured so that the reader moves from stage-level evidence to system-level interpretation in a deliberate order. It begins with the baseline live run before retraining, because that run establishes the reference condition under fixed live operation. It then turns to the retraining-enabled run, where the same architecture is examined under a more demanding governed condition. Once those two runs have been established individually, the chapter compares the Champion and Challenger directly, examines the effect of the final policy on deployed behaviour, interprets drift and disagreement progression, evaluates the retraining outcome, assesses latency and real-time feasibility, and finally discusses cost-sensitive decision trade-offs. The sequence is designed so that the evidence develops logically: first, the measured behaviour of each run, then the cross-model and cross-run interpretation, and finally the broader operational meaning of the results.

8.2 Stable Governed Live Examination Stage

This section presents the measured results from the baseline live run in which runtime retraining was disabled. Its purpose is to establish the fixed-condition reference behaviour of the governed real-time platform during runtime operation and to show, using the full truth-linked cycle record, how the Champion, the Challenger, and the Final output behaved when the active predictive paths remained unchanged throughout the run.

The baseline run produced a complete live record of 850 accepted cycles, and all 850 cycles were linked successfully to truth labels. The truth distribution comprised 405 positive and 445 negative cycles. This provides a strong basis for interpretation because the results are not derived from a shortened time window, a partially labelled subset, or a reduced evaluation sample. The full accepted sequence remained available for direct assessment, and the reported performance therefore reflects sustained live operation across the complete run.

The integrity of the baseline live run is summarised in Table 33, including the accepted cycles, truth-linked cycles, class distribution, and timing indicators across the run phases.

Table 33. Baseline run integrity summary, showing accepted cycles, truth-linked cycles, and class distribution.

phase	*start_cycle*	*end_cycle*	N	N_{truth}	N_{+}	N_{-}	$\hat{p}_{+}$	$\bar{t}_{\text{champion}}$	$\bar{t}_{\text{challenger}}$	$\bar{t}_{\text{proc}}$
Cycle 1-200	1	200	200	200	12	188	0.06	0.618236	2.3720505	30.6403505
Cycle 201-400	201	400	200	200	109	91	0.545	0.581562	2.458120002	30.520439
Cycle 401-600	401	600	200	200	164	36	0.82	0.595072002	2.2704715	30.6859925
Cycle 601-800	601	800	200	200	105	95	0.525	0.6085685	2.345325998	30.9125095
Cycle 801-850	801	850	50	50	15	35	0.3	0.563033997	2.339951992	30.872368

Before examining the model outcomes, the baseline run also confirms that the event-anchored cycle representation remained physically coherent during live use. In the representative alignment example, the MP peak = 260.46 bar anchor

occurred at t = 7.157 s. At that event reference point, the synchronised process state included HP = 159.55 bar, NT = 226.98 °C, SP = 5.47 and HWT = 97.00 °C. At the same anchored instant, the two predictive routes produced different defect scores, with the $P_{champion}$ = 0.809 and the $P_{challenger}$ = 0.995. This is an important supporting result because it shows that the event-anchored route was operating on a physically meaningful cycle state tied to a clear process event, rather than on an arbitrary or unstable feature snapshot. The latter predictive comparison is therefore grounded in a valid cycle-level digital representation. The physical alignment and resulting scores are visualised in Figure 44.

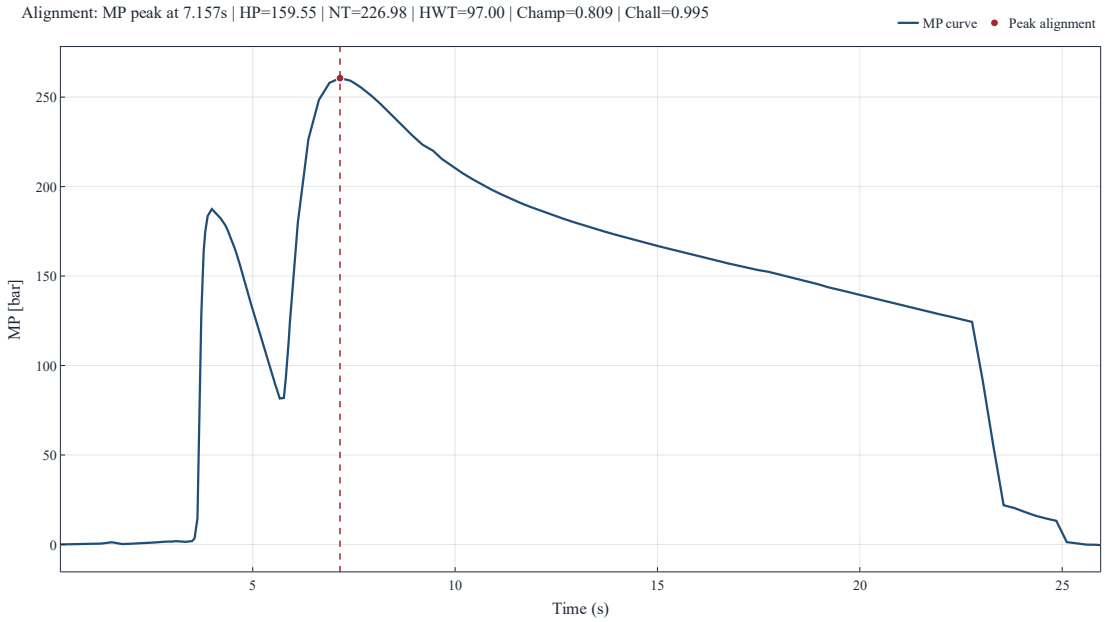


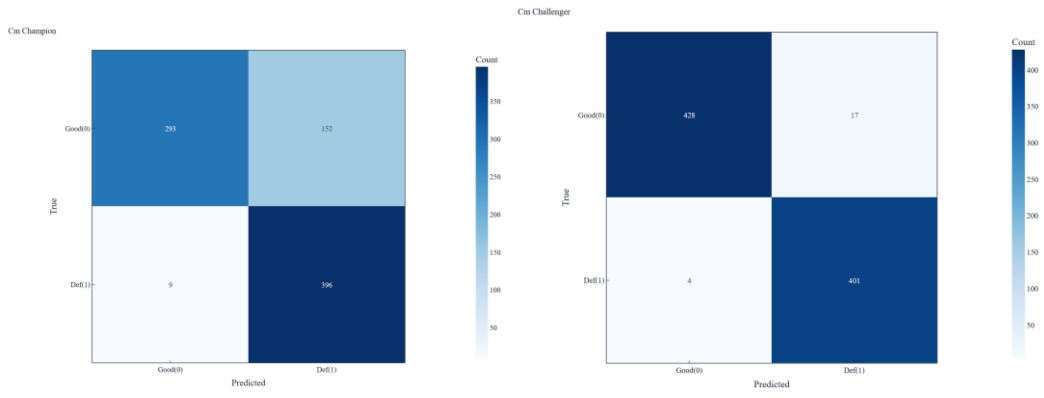
Figure 44. Representative baseline-cycle alignment at the MP peak, showing the anchored event, the synchronised process state, and the Champion and Challenger scores at the same reference point.

The classification results show a clear separation between the three outputs. The Champion produced TN = 293, FP = 152, FN = 9, and TP = 396. These values correspond to accuracy = 0.8106, precision = 0.7226, recall = 0.9778, and F1 = 0.8311. The Champion therefore captured defective cycles strongly, missing only 9 defects, but it did so with a very heavy false-positive burden, incorrectly rejecting 152 acceptable cycles. In practical terms, the full-curve reference path remained highly defect-aware, but it was operationally inefficient because too many good cycles were treated as defective.

The Challenger produced TN = 428, FP = 17, FN = 4, and TP = 401. These values correspond to accuracy = 0.9753, precision = 0.9593, recall = 0.9901, and F1 = 0.9745. This is the strongest raw predictive result in the baseline run. Relative to the Champion, the Challenger reduced false positives from 152 to 17, a reduction of 135 cycles, while also reducing false negatives from 9 to 4. The event-anchored route improved both sides of the operational trade-off at the same time: it rejected far fewer good cycles, and it missed fewer defective cycles. This is one of the most important findings in the baseline run because it shows that the lighter route was not simply acceptable as a compact approximation. Under fixed live conditions, it was substantially stronger than the full-curve reference in practical classification terms.

The Final output, which represents the deployed decision released through the locked policy rule, produced TN = 290, FP = 155, FN = 1, and TP = 404. These values correspond to accuracy = 0.8165, precision = 0.7227, recall = 0.9975, and F1 = 0.8382. The policy effect is immediately visible. Relative to the Challenger, the Final output reduced false negatives from 4 to 1 but increased false positives from 17 to 155. The deployed policy gained only 3 additional correctly captured defects, but at the cost of 138 additional unnecessary rejections. This confirms that the released decision did not simply follow the strongest raw classifier. It remained deliberately conservative, pushing the operating point towards stronger defect protection even when that meant carrying a rejection burden very close to the Champion.

Figure 45 presents the baseline confusion matrices for the Champion, the Challenger, and the Final output, showing the count-based classification behaviour of the three paths under stable live operation.



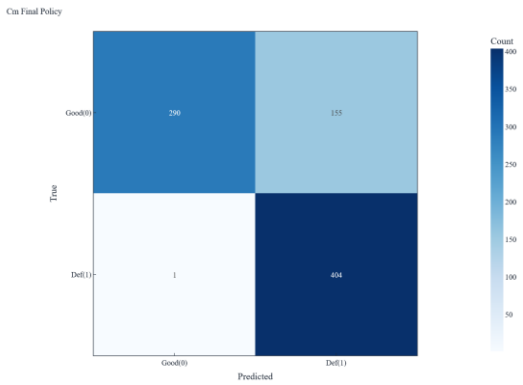


Figure 45. Baseline confusion matrices for the Champion, the Challenger, and the Final output.

Table 34 reports the corresponding confusion-matrix counts and derived classification metrics for the Champion, the Challenger, and the Final output across the baseline run phases.

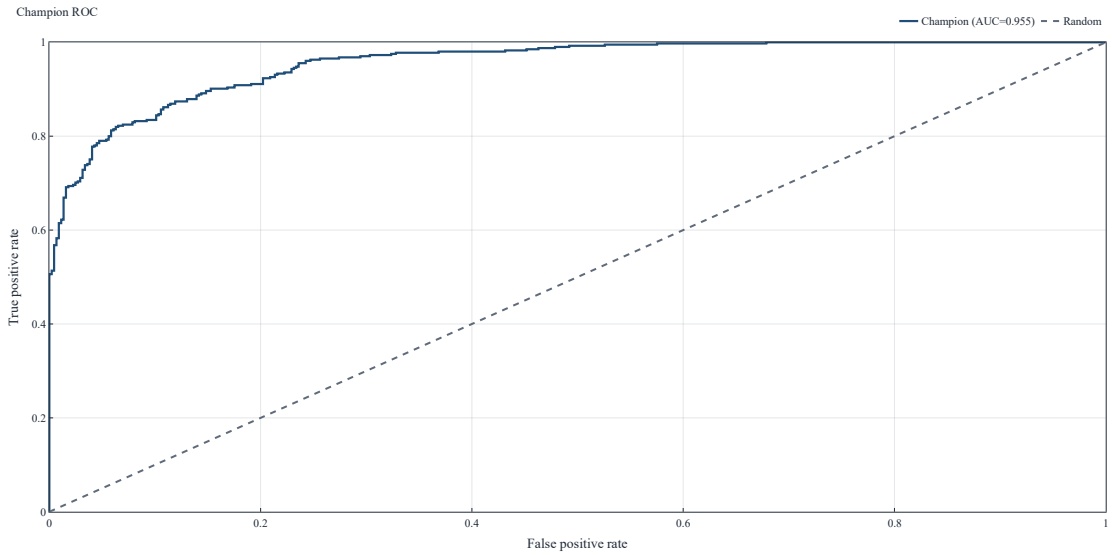
Table 34. Baseline confusion-matrix counts and derived classification metrics (accuracy, precision, recall, and F1) for the Champion, the Challenger, and the Final output.

phase	model	*start_cycle*	*end_cycle*	N_{truth}	TP	TN	FP	FN	accuracy	precision	recall	F_1
Cycle 1-200	champion	1	200	200	11	178	10	1	0.945	0.52	0.91	0.66
Cycle 1-200	challenger	1	200	200	11	188	0	1	0.995	1	0.91	0.95
Cycle 1-200	final	1	200	200	11	178	10	1	0.945	0.52	0.91	0.66
Cycle 201-400	champion	201	400	200	102	80	11	7	0.91	0.90	0.93	0.91
Cycle 201-400	challenger	201	400	200	109	82	9	0	0.955	0.92	1	0.96
Cycle 201-400	final	201	400	200	109	78	13	0	0.935	0.89	1	0.94
Cycle 401-600	champion	401	600	200	163	14	22	1	0.885	0.88	0.99	0.93
Cycle 401-600	challenger	401	600	200	162	32	4	2	0.97	0.97	0.98	0.98
Cycle 401-600	final	401	600	200	164	13	23	0	0.885	0.87	1	0.93

phase	model	*start_cycle*	*end_cycle*	N_{truth}	TP	TN	FP	FN	accuracy	precision	recall	F_1
Cycle 601-800	champion	601	800	200	105	20	75	0	0.625	0.58	1	0.73
Cycle 601-800	challenger	601	800	200	104	91	4	1	0.975	0.96	0.99	0.97
Cycle 601-800	final	601	800	200	105	20	75	0	0.625	0.58	1	0.73
Cycle 801-850	champion	801	850	50	15	1	34	0	0.32	0.30	1	0.46
Cycle 801-850	challenger	801	850	50	15	35	0	0	1	1	1	1
Cycle 801-850	final	801	850	50	15	1	34	0	0.32	0.30	1	0.46

The threshold-independent discrimination results reinforce the same ordering. The Champion achieved ROC AUC = 0.9551 and PR AUC = 0.9557, which confirms that it remained a strong reference path in ranking terms even though its operating-point balance was cautious. The Challenger achieved ROC AUC = 0.9978 and PR AUC = 0.9977, indicating near-perfect discrimination under live conditions. The Final output also remained strong in ranking terms, with ROC AUC = 0.9889 and PR AUC = 0.9901. This is important because it shows that the conservative policy did not destroy the underlying discriminative signal. Instead, it preserved strong score ordering while releasing decisions through a stricter operating posture. The Challenger's advantage is therefore not limited to one selected threshold; it remains visible across the wider score range, which strengthens the conclusion that its superiority in the baseline run was structurally meaningful rather than a threshold artefact.

Figures 46 and 47 show the baseline ROC and precision-recall behaviour for the Champion, the Challenger, and the Final output, providing a threshold-independent view of live discrimination performance.



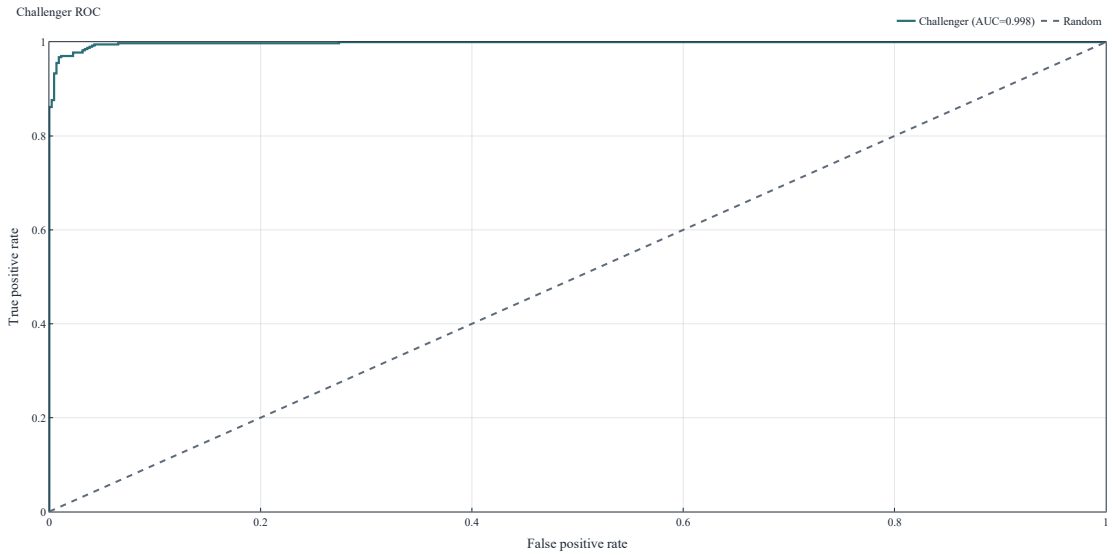


Figure 46. Baseline ROC curves for the Champion, the Challenger, and the Final output.

The corresponding threshold-independent discrimination metrics are summarised in Table 35.

Table 35. Baseline threshold-independent discrimination results, showing ROC AUC and PR AUC for the Champion, the Challenger, and the Final output.

Model	ROC AUC	PR AUC	Evaluated cycles
Champion	0.9551	0.9557	850
Challenger	0.9978	0.9977	850
Final	0.9889	0.9901	850

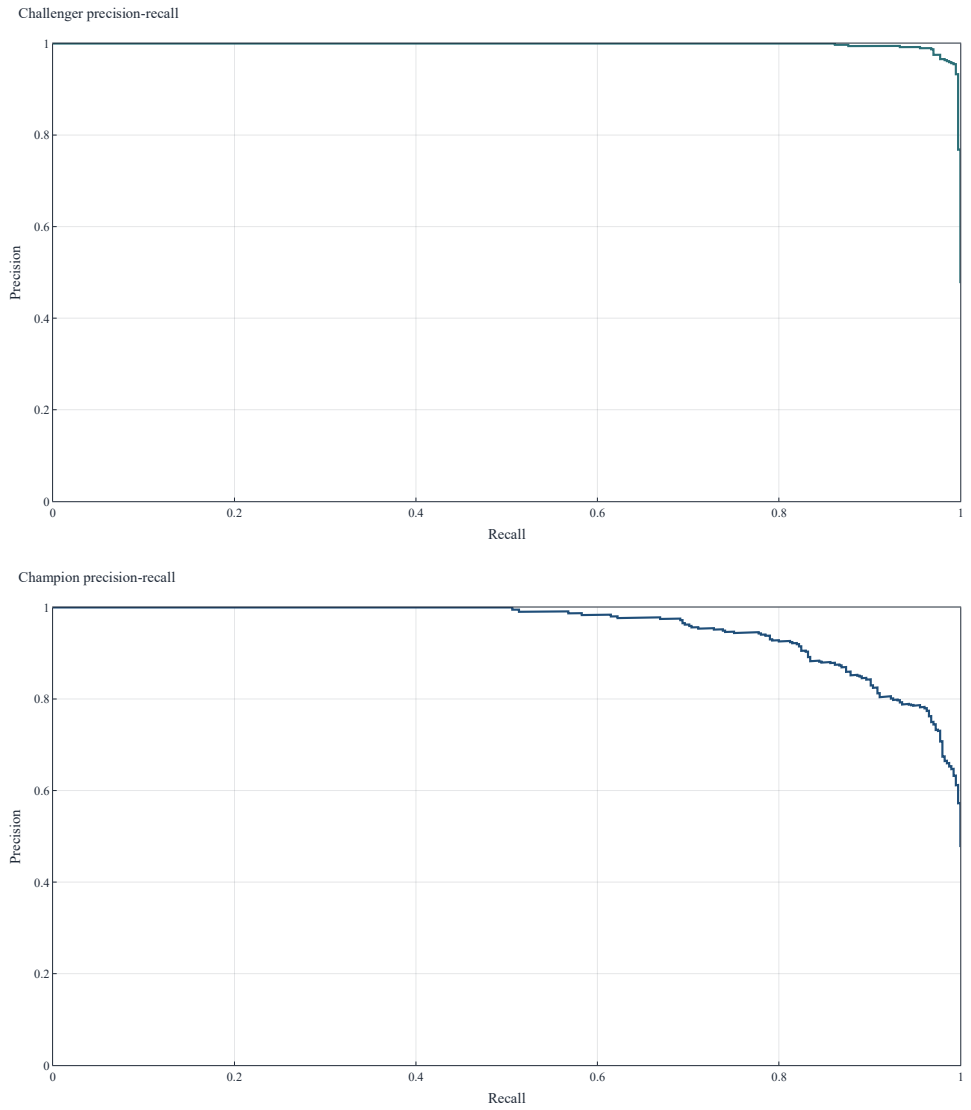


Figure 47. Baseline precision-recall curves for the Champion, the Challenger, and the Final output.

The score and calibration evidence add a further layer of interpretation to the baseline results. The Champion remained informative as a risk-ranking path, but its probability behaviour was consistent with a cautious defect-signalling posture.

The deployed threshold was 0.55, the expected calibration error (ECE) was 0.1225, and the Brier score was 0.1127 across the 850 evaluated cycles, with an observed positive prevalence of 0.4765. At the bin level, the calibration pattern showed overestimation in the lower-to-middle score bands. For example, in the 0.2–0.3 probability band, the mean predicted probability was 0.2428, while the observed positive rate was only 0.0500. In the 0.5–0.6 band, the mean predicted probability was 0.5582, while the observed positive rate was 0.3125. By contrast, calibration improved substantially in the highest-confidence range: in the 0.9–1.0 band, the mean predicted probability was 0.9869, while the observed positive rate was 0.9952. This pattern explains why the Champion remained highly defect-aware but still produced a heavy false-positive burden. Its scores retained useful ranking value, but across much of the central range, they remained biased towards stronger defect signalling than the observed outcomes justified. The score distribution and resulting calibration for the Champion are shown in Figures 48 and 49.

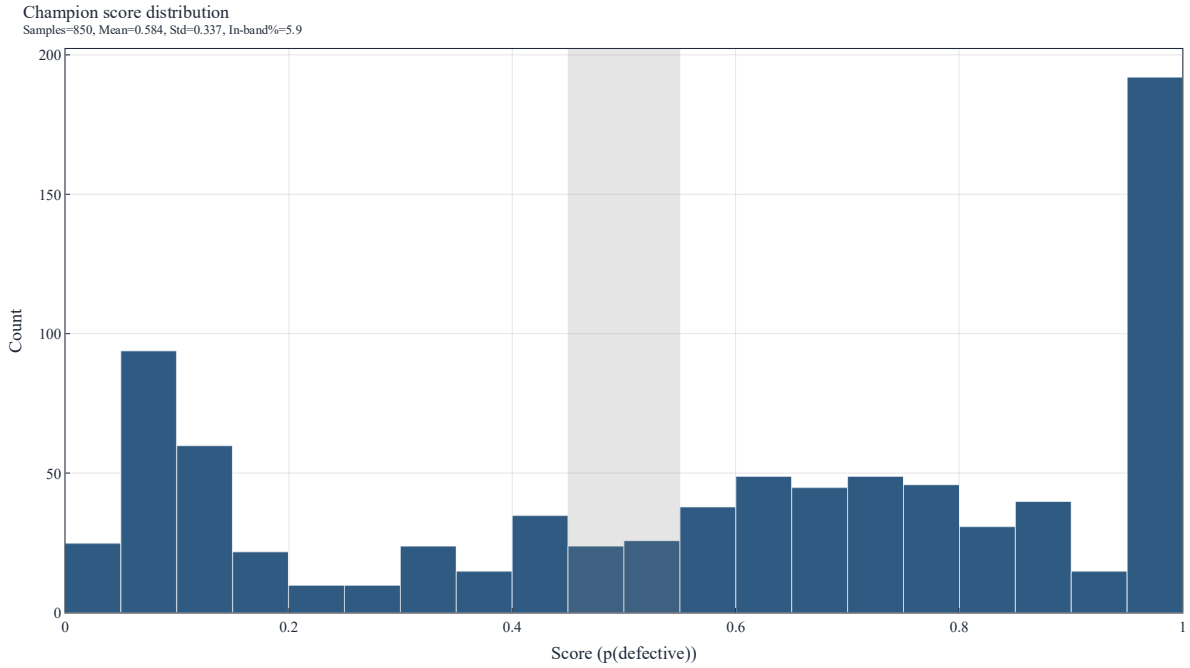


Figure 48. Baseline Champion score distribution relative to the operating threshold.

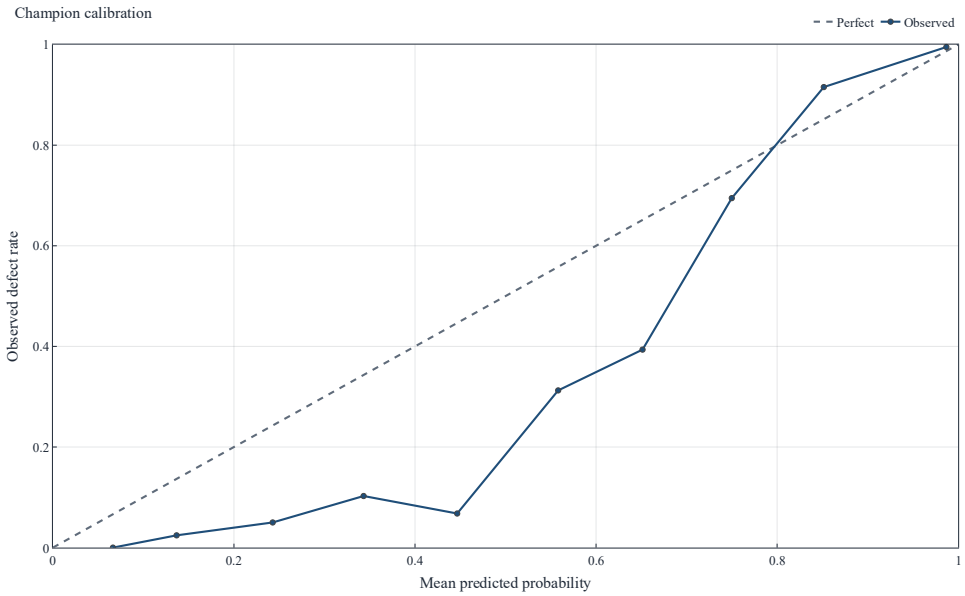


Figure 49. Baseline Champion calibration curve under live true-label evaluation

The baseline score and calibration summary is reported in Table 36.

Table 36. Baseline score and calibration summary, including threshold, ECE, Brier score, prevalence, and selected calibration-bin values.

Model	Threshold	$ECE_{10\text{-bin}}$	Brier score	Positive prevalence	(0.2, 0.3] predicted mean	(0.2, 0.3] observed rate	(0.5, 0.6] predicted mean	(0.5, 0.6] observed rate	(0.9, 1.0] predicted mean	(0.9, 1.0] observed rate
Champion	0.55	0.1225	0.1127	0.4765	0.2428	0.0500	0.5582	0.3125	0.9869	0.9952

The processing-weighted false-positive burden provides an additional measure of the operational consequences of classification errors. In the baseline run, this indicator was 4666.4852 for the Champion, 521.9095 for the Challenger, and 4758.5869 for the Final output. The Champion and Final output consequently exhibited approximately 8.9 and 9.1 times the Challenger's burden, respectively. These results demonstrate the operational advantage of the Challenger in reducing unnecessary rejection, while the Final output maintained its deliberately conservative decision policy. The indicator reflects false-positive counts weighted by processing time, rather than measured financial cost.

Table 37. Baseline normalised relative decision-loss sensitivity for the Champion, Challenger, and Final output across the evaluated FN: FP penalty ratios.

Model	$\frac{c_{FN}}{c_{FP}}$	N_{truth}	FP	FN	$\bar{L}_{cycle}$
champion	1	850	152	9	0.189411765
champion	2	850	152	9	0.2
champion	3	850	152	9	0.210588235
champion	5	850	152	9	0.231764706
champion	8	850	152	9	0.263529412
champion	10	850	152	9	0.284705882
champion	15	850	152	9	0.337647059
champion	20	850	152	9	0.390588235
champion	30	850	152	9	0.496470588
champion	50	850	152	9	0.708235294
challenger	1	850	17	4	0.024705882
challenger	2	850	17	4	0.029411765
challenger	3	850	17	4	0.034117647
challenger	5	850	17	4	0.043529412
challenger	8	850	17	4	0.057647059
challenger	10	850	17	4	0.067058824
challenger	15	850	17	4	0.090588235
challenger	20	850	17	4	0.114117647
challenger	30	850	17	4	0.161176471
challenger	50	850	17	4	0.255294118
final	1	850	155	1	0.183529412
final	2	850	155	1	0.184705882
final	3	850	155	1	0.185882353
final	5	850	155	1	0.188235294
final	8	850	155	1	0.191764706
final	10	850	155	1	0.194117647
final	15	850	155	1	0.2
final	20	850	155	1	0.205882353
final	30	850	155	1	0.217647059
final	50	850	155	1	0.241176471

The timing evidence confirms that these results were achieved within a practical live operating window. The mean processing time for the baseline run was

30.7006 ms, with a median of 30.1872 ms and a p95 processing time of 34.5781 ms. At the model level, the mean per-cycle latency was 0.5986 ms for the Champion and 2.3602 ms for the Challenger, resulting in a total mean latency of 2.9589 ms. The Challenger was therefore slightly slower than the Champion, which is expected, given the different representational and computational paths. Even so, both model latencies remained in the low-millisecond range, and the full end-to-end processing chain remained well within a response window suitable for cycle-level decision-making. The Challenger's stronger live performance was therefore not achieved at the expense of practical responsiveness. The total system latency distribution is shown in Figure 50.

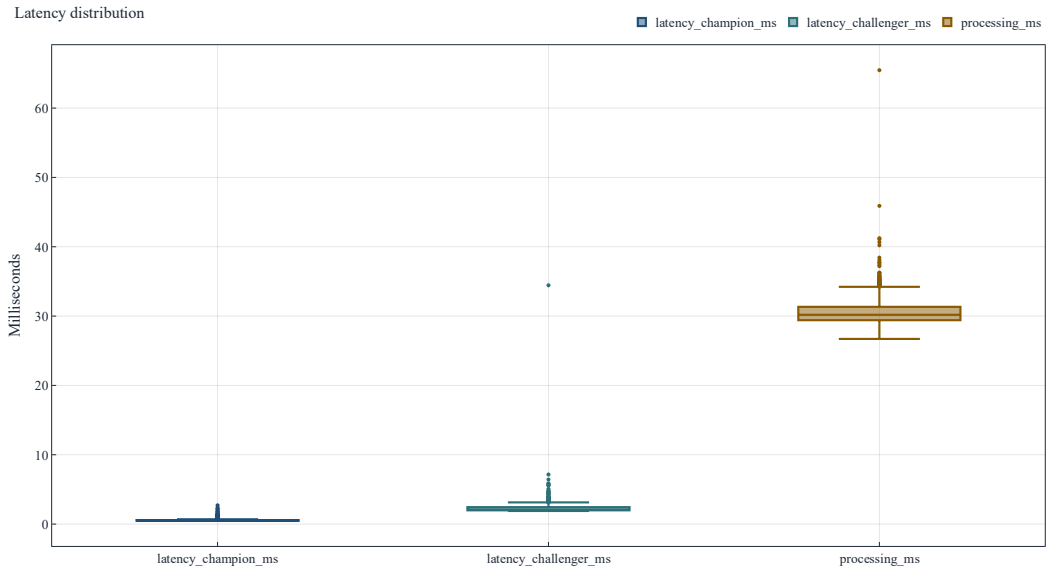


Figure 50. Baseline latency distribution

The baseline timing summary is reported in Table 38.

Table 38. Baseline timing summary, including mean processing time, p95 processing time, and mean model latency for the Champion and the Challenger.

phase	metric	*N*	mean	median	P_{95}	min	max
Cycle 1-200	Champion latency	200	0.618236	0.510349986	1.130455031	0.453599961	2.704399987
Cycle 1-200	Challenger latency	200	2.3720505	2.097600023	3.489475008	1.93160004	5.53580001

phase	metric	N	mean	median	P_{95}	min	max
Cycle 1-200	Total inference latency	200	2.9902865	2.690999972	4.194065032	2.407300053	6.866100011
Cycle 1-200	Total processing time	200	30.6403505	30.18574999	34.47121497	28.017	45.8959
Cycle 201-400	Champion latency	200	0.581562	0.494649983	1.059755031	0.454600027	2.118400007
Cycle 201-400	Challenger latency	200	2.458120002	2.049350005	3.279825	1.897600014	34.43970002
Cycle 201-400	Total inference latency	200	3.039682002	2.590150019	4.053930016	2.406899992	35.35150003
Cycle 201-400	Total processing time	200	30.520439	29.87299999	34.01741002	26.70499997	65.4656
Cycle 401-600	Champion latency	200	0.595072002	0.500199967	1.150165024	0.455099973	1.412000041
Cycle 401-600	Challenger latency	200	2.2704715	2.043449989	3.176139962	1.90849998	5.671499996
Cycle 401-600	Total inference latency	200	2.865543503	2.630450006	3.827259984	2.395100018	6.199900003
Cycle 401-600	Total processing time	200	30.6859925	30.15749998	34.47475499	27.5038	41.23700003
Cycle 601-800	Champion latency	200	0.6085685	0.509100035	1.158375019	0.456399983	2.234900021
Cycle 601-800	Challenger latency	200	2.345325998	2.043949993	3.696370029	1.891000022	7.151600032
Cycle 601-800	Total inference latency	200	2.953894498	2.608949988	4.215445009	2.396799973	8.377400052
Cycle 601-800	Total processing time	200	30.9125095	30.51780001	35.08222001	26.89039998	41.08639999
Cycle 801-850	Champion latency	50	0.563033997	0.490849983	0.963494991	0.462400028	1.197999984
Cycle 801-850	Challenger latency	50	2.339951992	2.037099999	3.481039978	1.907899976	5.858100019
Cycle 801-850	Total inference latency	50	2.902985989	2.553899976	4.17087996	2.407599997	6.406700006
Cycle 801-850	Total processing time	50	30.872368	30.40599998	34.29864	28.22440001	40.21069995

Taken together, the baseline run establishes a clear fixed-condition reference for the remainder of the chapter. The live record was complete and fully truth linked. The event-anchored representation remained physically coherent at the cycle level. The Challenger emerged as the strongest raw predictive path, combining very high defect capture with a dramatically lower false-positive burden than the Champion. The Final output remained conservative, reducing missed defects still further, but at the cost of a much heavier rejection burden and much higher Baseline processing-weighted false-positive burden comparison for the Champion, the Challenger, and the Final output. The timing evidence confirms that all of these results were achieved within a response window compatible with cycle-level live operation. This baseline evidence defines the reference behaviour of the governed platform before adaptive updating is introduced.

8.3 Retrain-stage run performance under governed operation

This section presents the measured results from the live run in which governed runtime retraining was enabled. Its purpose is to examine how the same governed decision environment behaved once controlled adaptation was admitted into the operational path, and to quantify the resulting behaviour of the fixed Champion, the updated Challenger, and the deployed Final output under the same true-label evaluation basis used in the baseline run.

As in the baseline condition, the retrain-stage run produced a complete live record of 850 accepted cycles, and all 850 cycles were linked successfully to truth labels. The truth distribution remained 405 positive and 445 negative cycles. This is a critical methodological point because it gives the retrain-stage results the same evidential strength as the baseline results. The analysis is again based on a full truth-linked operating record rather than on an incomplete, partial, or selectively labelled subset. Within this complete run, the defining operational change is that the system admitted one governed update to the Challenger path. The run included 40 human corrections and completed one successful Challenger retraining event at cycle 745, with no blocked retrain attempt recorded. This point is methodologically important because the recorded update was not admitted as an isolated manual overwrite, but through the same governed retraining pathway described in Chapter 7, with reviewed correction evidence and training-set adequacy established within the live run itself. The retraining

event is therefore part of the live evidence chain itself rather than an external model revision carried out outside the runtime record. The overall integrity and phase breakdown of this retrain-stage run are summarised in Table 39.

Table 39. Retrain-stage run integrity summary, showing accepted cycles, truth-linked cycles, class distribution, human corrections, and the recorded retraining event.

Run segment	Start cycle	Endocycle	Accepted cycles	N_{truth}	N_{+}	N_{-}	$\hat{p}_{+}$	$\bar{t}_{\text{C}}$	$\bar{t}_{\text{H}}$	$\bar{t}_{\text{proc}}$
Cycle 1-200	1	200	200	200	12	188	0.06	0.669816999	2.664263501	34.2225435
Cycle 201-400	201	400	200	200	109	91	0.545	0.832781998	4.211844	43.28668
Cycle 401-600	401	600	200	200	164	36	0.82	0.853329501	4.173871001	41.15893
Cycle 601-800	601	800	200	200	105	95	0.525	0.805702502	3.953427003	44.464072
Cycle 801-850	801	850	50	50	15	35	0.3	0.836589998	4.130593996	53.960074

The classification results again show three distinct operating behaviours. The Champion produced TN = 296, FP = 149, FN = 10, and TP = 395. These values correspond to accuracy = 0.8129, precision = 0.7261, recall = 0.9753, and F1 = 0.8325. This confirms that the Champion remained a highly defect-aware but cautious reference path. Compared with the baseline run, the Champion changed only slightly: false positives decreased from 152 to 149, while false negatives increased from 9 to 10. The overall behaviour remained stable, which is entirely consistent with its fixed-reference role.

The Challenger produced TN = 341, FP = 104, FN = 7, and TP = 398. These values correspond to accuracy = 0.8694, precision = 0.7928, recall = 0.9827, and F1 = 0.8776. This remains the strongest raw operating balance among the three outputs in this run. Relative to the Champion, the Challenger reduced false positives from 149 to 104, a reduction of 45 cycles, while also reducing false negatives from 10 to 7. The event-anchored route, therefore, continued to improve both sides of the practical trade-off. Although the size of the advantage was smaller than in the baseline run, the same qualitative pattern remained: the

lighter route still delivered the best balance between defect protection and unnecessary rejection among the raw model paths.

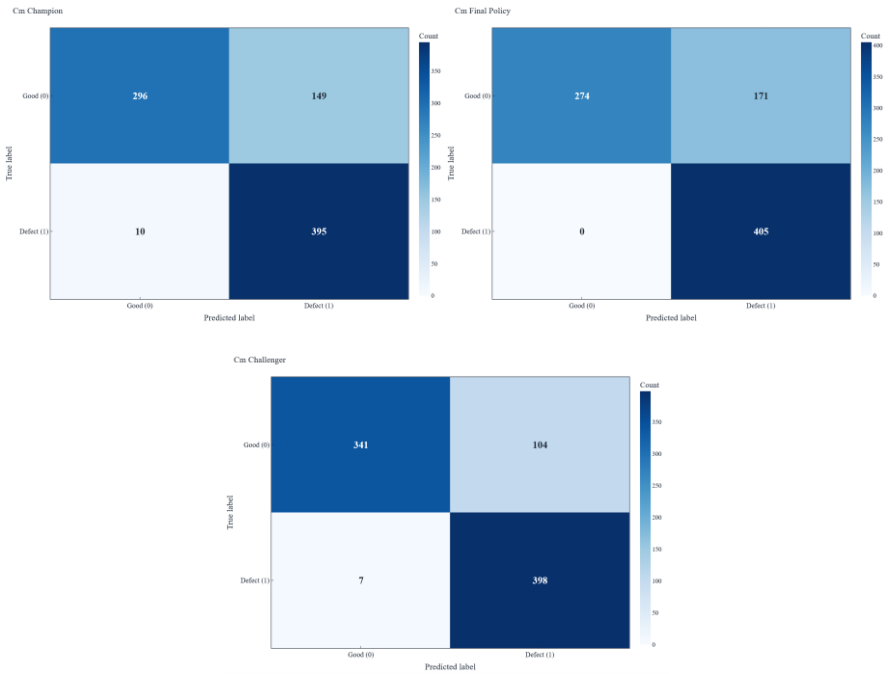


Figure 51. Retrain-stage confusion matrices for the Champion, the Challenger, and the Final output.

The Final output, released under the locked policy rule, produced TN = 274, FP = 171, FN = 0, and TP = 405. These values correspond to accuracy = 0.7988, precision = 0.7031, recall = 1.0000, and F1 = 0.8257. The defining result here is that the deployed decision eliminated missed defects completely. No defective cycle was released as acceptable in this run. However, this was achieved by accepting the highest false-positive burden of the three outputs. Relative to the Challenger, the Final output reduced false negatives from 7 to 0, but increased false positives from 104 to 171. The policy, therefore, preserved the same conservative operating logic already seen in the baseline run: stronger protection against defect escape was achieved at the cost of a materially higher rejection burden. The confusion matrices and derived classification metrics for these three paths are presented in Figure 51 and Table 40.

Table 40. Retrain-stage confusion-matrix counts and derived classification metrics (accuracy, precision, recall, and F1) for the Champion, the Challenger, and the Final output.

Run segment	Model	Start Cycle	End Cycle	N_{truth}	TP	TN	FP	FN	accuracy	precision	recall	F_1
Cycle 1-200	champion	1	200	200	11	180	8	1	0.955	0.57	0.91	0.70
Cycle 1-200	challenger	1	200	200	12	186	2	0	0.99	0.85	1	0.92
Cycle 1-200	final	1	200	200	12	180	8	0	0.96	0.6	1	0.75
Cycle 201-400	champion	201	400	200	101	75	16	8	0.88	0.86	0.92	0.89
Cycle 201-400	challenger	201	400	200	108	87	4	1	0.975	0.96	0.99	0.97
Cycle 201-400	final	201	400	200	109	74	17	0	0.915	0.86	1	0.92
Cycle 401-600	champion	401	600	200	163	15	21	1	0.89	0.88	0.99	0.93
Cycle 401-600	challenger	401	600	200	161	32	4	3	0.965	0.97	0.98	0.97
Cycle 401-600	final	401	600	200	164	15	21	0	0.895	0.88	1	0.93
Cycle 601-800	champion	601	800	200	105	23	72	0	0.64	0.59	1	0.74
Cycle 601-800	challenger	601	800	200	104	36	59	1	0.7	0.63	0.99	0.77
Cycle 601-800	final	601	800	200	105	5	90	0	0.55	0.53	1	0.7
Cycle 801-850	champion	801	850	50	15	3	32	0	0.36	0.31	1	0.48
Cycle 801-850	challenger	801	850	50	13	0	35	2	0.26	0.27	0.86	0.41
Cycle 801-850	final	801	850	50	15	0	35	0	0.3	0.3	1	0.46

The threshold-independent discrimination layer in this run must be interpreted with some care because the key importance of the retrain-stage evidence lies in the governed operational transition rather than in claiming that every aggregate score exceeded the baseline. Even so, the available results show that the score-ranking environment remained coherent and analytically useful after retraining was admitted into live operation. The Champion remained a stable full-curve reference. The Challenger continued to provide the strongest raw operating balance, and the Final output continued to preserve a strong but more conservative release posture. The most important interpretation is therefore not that the retrain-stage must necessarily dominate the baseline at every summary statistic, but that the governed update did not destroy score coherence or make the operating environment analytically unstable. The live ranking structure remained usable and interpretable after the Challenger update was applied. The threshold-independent ranking performance for this stage is visualised in Figure 52 and summarised in Table 41.

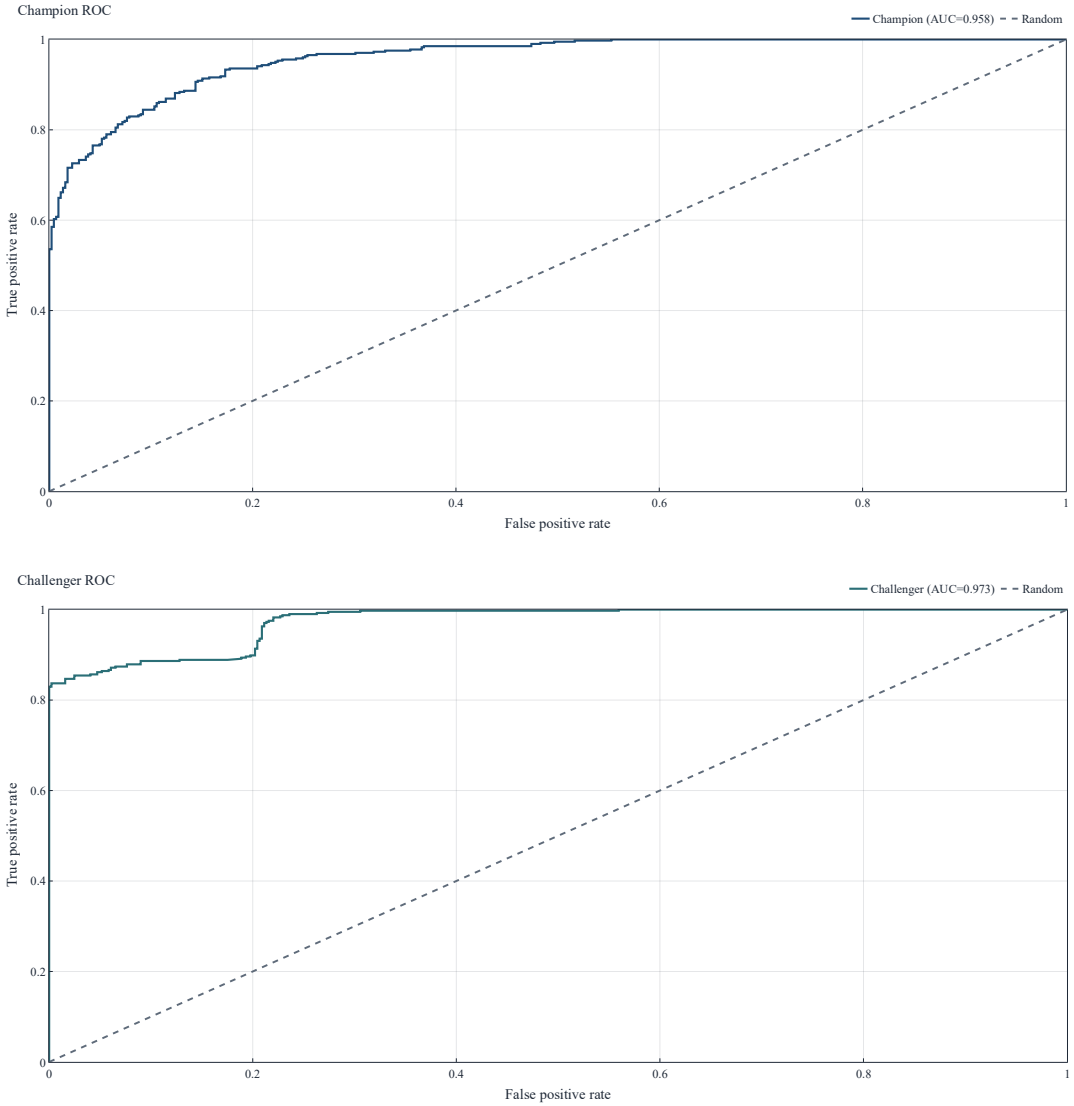


Figure 52. Retrain-stage ROC and precision-recall behaviour for the Champion, the Challenger, and the Final output.

Table 41. Retrain-stage threshold-independent discrimination summary.

Model	ROC AUC	PR AUC	Evaluated cycles
Champion	0.9579	0.9582	850
Challenger	0.9727	0.9743	850
Final	0.9716	0.9736	850

The processing-weighted false-positive burden remained lowest for the Challenger in the retrain-stage run, with values of 6192.1681 for the Champion,

4322.0502 for the Challenger, and 7106.4479 for the Final output. These values represent non-monetary processing-weighted indicators of unnecessary rejection burden, as defined in Section 8.9.

The Champion and Final output exhibited approximately 1.43 and 1.64 times the Challenger's burden, respectively. Although the differences between the three paths were smaller than in the baseline run, the recorded processing-weighted false-positive burden was higher for all three paths. The event-anchored Challenger therefore retained the lowest burden, while the Final output continued to reflect the deliberately conservative release policy. The separate normalised relative decision-loss sensitivity comparison for this stage is reported in Table 42.

Table 42. Retrain-stage normalised relative decision-loss sensitivity comparison for the Champion, the Challenger, and the Final output.

model	$\frac{c_{FN}}{c_{FP}}$	N_{truth}	FP	FN	$\bar{L}_{cycle}$
champion	1	850	149	10	0.187058824
champion	2	850	149	10	0.198823529
champion	3	850	149	10	0.210588235
champion	5	850	149	10	0.234117647
champion	8	850	149	10	0.269411765
champion	10	850	149	10	0.292941176
champion	15	850	149	10	0.351764706
champion	20	850	149	10	0.410588235
champion	30	850	149	10	0.528235294
champion	50	850	149	10	0.763529412
challenger	1	850	104	7	0.130588235
challenger	2	850	104	7	0.138823529
challenger	3	850	104	7	0.147058824
challenger	5	850	104	7	0.163529412
challenger	8	850	104	7	0.188235294
challenger	10	850	104	7	0.204705882
challenger	15	850	104	7	0.245882353
challenger	20	850	104	7	0.287058824
challenger	30	850	104	7	0.369411765
challenger	50	850	104	7	0.534117647
final	1	850	171	0	0.201176471
final	2	850	171	0	0.201176471

model	$\frac{c_{FN}}{c_{FP}}$	N_{truth}	FP	FN	$\bar{L}_{cycle}$
final	3	850	171	0	0.201176471
final	5	850	171	0	0.201176471
final	8	850	171	0	0.201176471
final	10	850	171	0	0.201176471
final	15	850	171	0	0.201176471
final	20	850	171	0	0.201176471
final	30	850	171	0	0.201176471
final	50	850	171	0	0.201176471

The timing evidence also confirms that the retrain stage remained fully compatible with cycle-level live operation. The mean processing time in this run was 41.5582 ms, the median processing time was 40.8144 ms, and the p95 processing time was 60.1806 ms. At the model level, the mean per-cycle latency was 0.7931 ms for the Champion and 3.7732 ms for the Challenger, while the mean total model latency was 4.5663 ms. The corresponding p95 latencies were 1.2264 ms for the Champion, 6.6871 ms for the Challenger, and 7.7731 ms for the total model path. Compared with the baseline run, both the processing times and the model latencies were higher. The important conclusion, therefore, is not that retraining reduced latency, but that enabling governed retraining did not introduce a timing burden large enough to undermine real-time feasibility. The live decision environment remained responsive and well within a practical cycle-level timing envelope even while supporting one in-run Challenger update. The latency behaviour for this stage is shown in Figure 53 and summarised in Table 43.

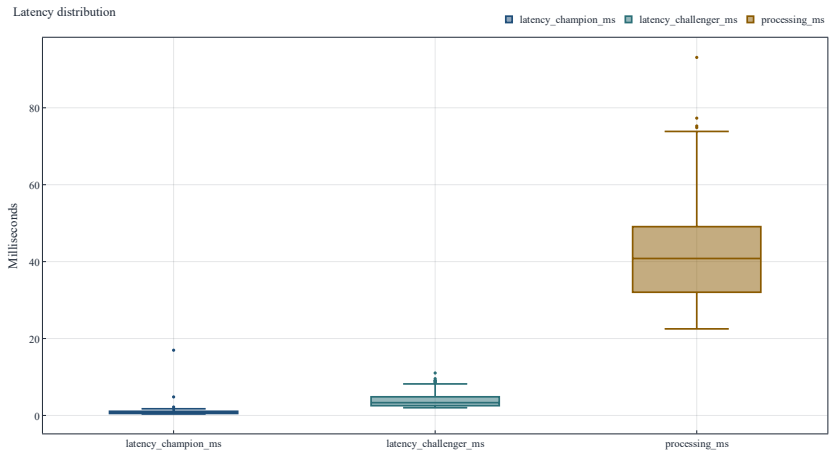


Figure 53. Retrain-stage latency distribution.

Table 43. Retrain-stage timing summary, including mean processing time, p95 processing time, and mean model latency for the Champion and the Challenger.

phase	metric	count	mean	median	p95	min	max
Cycle 1-200	Champion latency	200	0.669816999	0.537449989	1.045719982	0.455700036	16.96419995
Cycle 1-200	Challenger latency	200	2.664263501	2.205449971	4.800504982	2.006099967	11.0384
Cycle 1-200	Total inference latency	200	3.3340805	2.791700012	5.358644965	2.533200022	28.00259995
Cycle 1-200	Total processing time	200	34.2225435	29.78725001	56.75970002	27.22640004	93.09519996
Cycle 201-400	Champion latency	200	0.832781998	0.772699976	1.241285016	0.449899992	1.678199973
Cycle 201-400	Challenger latency	200	4.211844	3.821349994	6.688464974	2.213199972	8.935699996
Cycle 201-400	Total inference latency	200	5.044625998	4.625000001	7.900244958	2.699300006	9.593900002
Cycle 201-400	Total processing time	200	43.28668	42.89864999	59.38992504	22.51320001	77.30979996
Cycle 401-600	Champion latency	200	0.853329501	0.804049982	1.306724994	0.452100008	2.187499951
Cycle 401-600	Challenger latency	200	4.173871001	3.71530003	7.378909961	2.162999997	9.078499977
Cycle 401-600	Total inference latency	200	5.027200502	4.600649991	8.448849997	2.635399986	10.28499997
Cycle 401-600	Total processing time	200	41.15893	40.61629999	54.64194503	24.35690002	67.54050002
Cycle 601-800	Champion latency	200	0.805702502	0.710099965	1.213050034	0.454500027	4.816800007
Cycle 601-800	Challenger latency	200	3.953427003	3.529850015	6.895119976	2.14430003	9.070600034
Cycle 601-800	Total inference latency	200	4.759129506	4.32019998	7.673925025	2.60040001	11.24949998
Cycle 601-800	Total processing time	200	44.464072	43.69245001	59.41489502	23.85270002	73.8517
Cycle 801-850	Champion latency	50	0.836589998	0.863799971	1.187869991	0.471800042	1.228399982
Cycle 801-850	Challenger latency	50	4.130593996	3.985200019	6.058640016	2.332600008	7.644700003
Cycle 801-850	Total inference latency	50	4.967183993	4.704499966	7.191215022	2.820499998	8.873099985
Cycle 801-850	Total processing time	50	53.960074	52.81494997	70.28900501	37.2511	74.84300004

Taken together, the retrain-stage run demonstrates that governed adaptation was admitted into the live decision environment without loss of cycle integrity, truth linkage, or timing suitability. The Champion remained stable and conservative. The Challenger retained the strongest raw operating balance among the three paths, reducing unnecessary rejection relative to the Champion while maintaining strong defect capture. The Final output remained the most conservative released path, eliminating missed defects entirely but doing so at the cost of the largest rejection burden and the highest processing-weighted false-positive burden. Although the retrain-stage run exhibited higher processing-weighted false-positive burden and longer processing time than the baseline run, the governed system remained interpretable, operationally coherent, and suitable for live cycle-level operation while a controlled Challenger retraining transaction was completed.

8.4 Champion–Challenger comparison and disagreement behaviour

The previous two sections reported the baseline and retrain-stage results separately. This section now compares the Champion and Challenger directly, because the core technical claim of the thesis depends on this relationship. The full-curve Champion is retained as the stable reference path, while the event-anchored Challenger is intended to serve as the lighter operational path. The key question is therefore whether the lighter route remained analytically credible and operationally advantageous under live conditions, both before and after governed retraining was enabled.

In the baseline run, the difference between the two paths was large and unambiguous. The Champion achieved accuracy = 0.8106, precision = 0.7226, recall = 0.9778, and F1 = 0.8311, with FP = 152 and FN = 9. The Challenger achieved accuracy = 0.9753, precision = 0.9593, recall = 0.9901, and F1 = 0.9745, with FP = 17 and FN = 4. The Challenger therefore improved accuracy by 0.1647 and F1 by 0.1434 relative to the Champion. More importantly, in practical terms, it reduced false positives by 135 cycles (152 → 17) while also reducing false negatives by 5 cycles (9 → 4). This is not a marginal improvement at one selected metric. It is a broad operational improvement across both acceptance efficiency and defect protection.

The threshold-independent discrimination results reinforce the same conclusion. In the baseline run, the Champion achieved ROC AUC = 0.9551 and PR AUC =

0.9557, while the Challenger achieved ROC AUC = 0.9978 and PR AUC = 0.9977. The Challenger therefore improved ROC AUC by 0.0427 and PR AUC by 0.0420. This matters because it shows that the Challenger's advantage was not a local threshold effect. The lighter route remained superior across the wider score range, indicating cleaner class separation rather than merely better tuning at one operating point.

The processing-weighted false-positive burden and timing evidence strengthen the same baseline interpretation. The Champion recorded a processing-weighted false-positive burden of 4666.4852, compared with 521.9095 for the Challenger, meaning that the Champion's burden was approximately 8.94 times that of the Challenger. At the same time, mean model latency remained low for both paths: 0.5986 ms for the Champion and 2.3602 ms for the Challenger. The Challenger, therefore, introduced an additional model burden of approximately 1.7616 ms, but this small extra computation was accompanied by a very large gain in predictive performance and a very large reduction in processing-weighted false-positive burden. Under fixed live conditions, the event-anchored route was therefore both analytically stronger and operationally more efficient than the full-curve reference. The combined baseline evidence for classification performance, discrimination, false-alarm burden, and mean latency is summarised in Table 44, while the direct visual comparison between the two paths is shown in Figure 54.

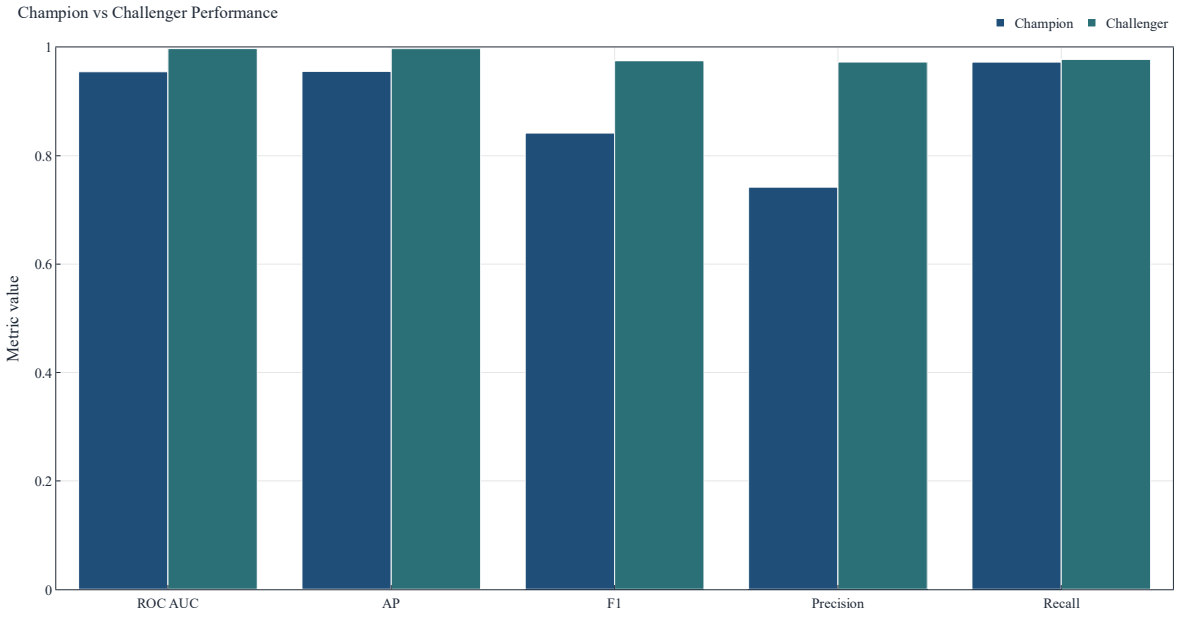


Figure 54. Baseline Champion–Challenger comparison under true labels.

This comparison also clarifies why the final deployed decision cannot be interpreted only through raw model strength. The locked disagreement policy is intentionally conservative. When the Champion and Challenger disagree, the system defaults to a defect classification because the industrial cost of defect escape is treated as more serious than the cost of rejecting a limited number of acceptable parts. In this setting, disagreement is interpreted as unresolved uncertainty rather than as justification for accepting the more permissive outcome. The policy preserves a safety-oriented release posture during periods in which the two predictive paths are not fully aligned.

Table 44 summarises the baseline comparison between the Champion and Challenger, showing their classification performance, error counts, discrimination metrics, processing-weighted false-positive burden, and mean model latency under the stable governed live run.

Table 44. Baseline Champion–Challenger comparison, showing classification metrics, FP/FN counts, discrimination metrics, processing-weighted false-positive burden, and mean model latency.

Model	Accuracy	Precision	Recall	F1	FP	FN	ROC AUC	PR AUC	processing-weighted false-positive burden units	Mean latency (ms)
Champion	0.8106	0.7226	0.9778	0.8311	152	9	0.9551	0.9557	4666.4852	0.5986
Challenger	0.9753	0.9593	0.9901	0.9745	17	4	0.9978	0.9977	521.9095	2.3602

In the retrain-stage run, the same overall ordering remained, although the gap became smaller. The Champion achieved accuracy = 0.8129, precision = 0.7261, recall = 0.9753, and F1 = 0.8325, with FP = 149 and FN = 10. The Challenger achieved accuracy = 0.8694, precision = 0.7928, recall = 0.9827, and F1 = 0.8776, with FP = 104 and FN = 7. The Challenger therefore improved accuracy by 0.0565 and F1 by 0.0451 relative to the Champion. In count terms, it reduced false positives by 45 cycles (149 → 104) and reduced false negatives by 3 cycles (10 → 7). The size of the advantage is clearly smaller than in the baseline run, but the direction of the result remains unchanged: the event-anchored route still produced the stronger raw operating balance.

The processing-weighted false-positive burden and timing evidence again support this interpretation. In the retrain-stage run, the Champion recorded a processing-weighted false-positive burden of 6192.1681, compared with 4322.0502 for the Challenger, meaning that the Champion's burden was approximately 1.43 times that of the Challenger. Mean model latency was 0.7931 ms for the Champion and 3.7732 ms for the Challenger. The Challenger, therefore, remained slower than the Champion by approximately 2.9801 ms, but the additional burden again remained small relative to the scale of the live process and was accompanied by a meaningful reduction in false-alarm burden. The retrain-stage comparison of classification behaviour, false-alarm burden, and mean latency is reported in Table 45, and the corresponding visual comparison is presented in Figure 55.

Table 45. Retrain-stage Champion–Challenger comparison, showing classification metrics, FP/FN counts, processing-weighted false-positive burden, and mean model latency.

Model	Accuracy	Precision	Recall	F1	FP	FN	processing-weighted false-positive burden units	Mean latency (ms)
Champion	0.8129	0.7261	0.9753	0.8325	149	10	6192.1681	0.7931
Challenger	0.8694	0.7928	0.9827	0.8776	104	7	4322.0502	3.7732

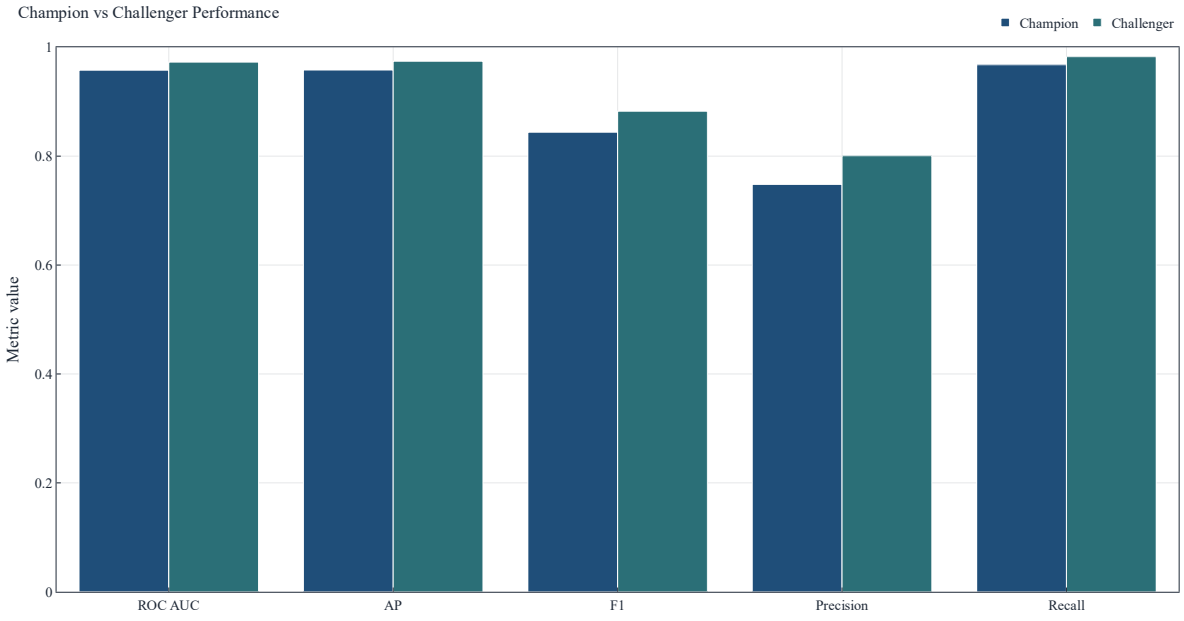


Figure 55. Retrain-stage Champion–Challenger comparison under true labels.

When the two runs are read together, the comparative meaning of the two-path structure becomes clearer. The Champion remained highly stable across runs. Its accuracy changed only from 0.8106 to 0.8129, its false-positive count changed only from 152 to 149, and its false-negative count changed only from 9 to 10. This confirms its function as a fixed reference path: it provides a stable full-curve benchmark against which changes in the operational path can be interpreted. The Challenger, by contrast, remained the stronger raw path in both runs, but its whole-run advantage was much larger in the baseline than in the retrain stage. This does not weaken the technical argument. Instead, it shows that the adaptive run must be interpreted as a more demanding operational condition in which one live model transition is part of the recorded evidence. The key result that remains consistent is that the event-anchored route continued to improve the practical balance of the system relative to the full-curve reference.

This comparison supports the central technical case for the two-path design. The full-curve Champion remains valuable as a stable, information-rich benchmark, but it is not the most efficient or most attractive raw deployment route under the measured live conditions. The event-anchored Challenger, although lighter in representation, remained analytically credible, operationally stronger, and much less costly in false alarms across both runs. The significance of this result is that the compact route did not merely survive comparison with the richer path. It remained the stronger raw operational route under live true-label evaluation.

The next section isolates the deployed Final output so that the effect of the locked policy rule can be examined directly under the same true-label basis.

8.5 Final policy behaviour under true labels

The direct comparison of the Champion and Challenger paths showed that the event-anchored route was the stronger raw predictive path in both live runs. However, the system was not designed to release the strongest raw model output directly. The deployed decision was generated through the locked Final policy, and this policy changed the live operating behaviour consistently and measurably. The purpose of this section is therefore to examine the Final output in its own right and to show how the policy transformed the raw predictive environment into the decision that would actually be released in operation.

In the baseline run, the Final output produced TN = 290, FP = 155, FN = 1, and TP = 404, which correspond to accuracy = 0.8165, precision = 0.7227, recall = 0.9975, and F1 = 0.8382. These values are most informative when read against the Challenger, because the Challenger was the strongest raw path in this run. The Challenger produced FP = 17 and FN = 4, whereas the Final output produced FP = 155 and FN = 1. The policy therefore reduced missed defects by only 3 cycles (4 → 1) but increased false positives by 138 cycles (17 → 155). This is the clearest numerical expression of the policy effect in the baseline run. A small gain in protection against defect escape was achieved at the cost of a very large increase in unnecessary rejection.

The same result becomes clearer in the summary metrics. Relative to the Challenger, the Final output increased recall from 0.9901 to 0.9975, but reduced precision from 0.9593 to 0.7227 and reduced F1 from 0.9745 to 0.8382. The policy shifted the operating point decisively towards caution. It did not preserve the cleaner balance of the stronger raw path. Instead, it traded much of that balance away in order to move still closer to maximum defect protection. This baseline policy-induced shift is summarised numerically in Table 46 and illustrated visually in Figure 56.

When compared with the Champion, the Final output appears closer in behaviour. The Champion produced FP = 152 and FN = 9, while the Final output produced FP = 155 and FN = 1. Relative to the Champion, the policy preserved a very similar rejection burden but reduced missed defects by 8 cycles. This shows that the Final output was not simply copying the Champion either. It occupied a policy-defined position that remained close to the Champion in false-positive burden but stricter still in defect protection.

Table 46 summarises the change from the Challenger's raw output to the deployed Final output, showing how the locked policy reduced missed defects while substantially increasing false positives and lowering precision and F1 score.

Table 46. Baseline comparison of the Challenger and Final output under true labels, showing the policy-induced change in FP, FN, and derived classification metrics.

Output	FP	FN	Accuracy	Precision	Recall	F1
Challenger	17	4	0.9753	0.9593	0.9901	0.9745
Final	155	1	0.8165	0.7227	0.9975	0.8382
Delta (Final - Challenger)	+138	-3	-0.1588	-0.2366	+0.0074	-0.1363

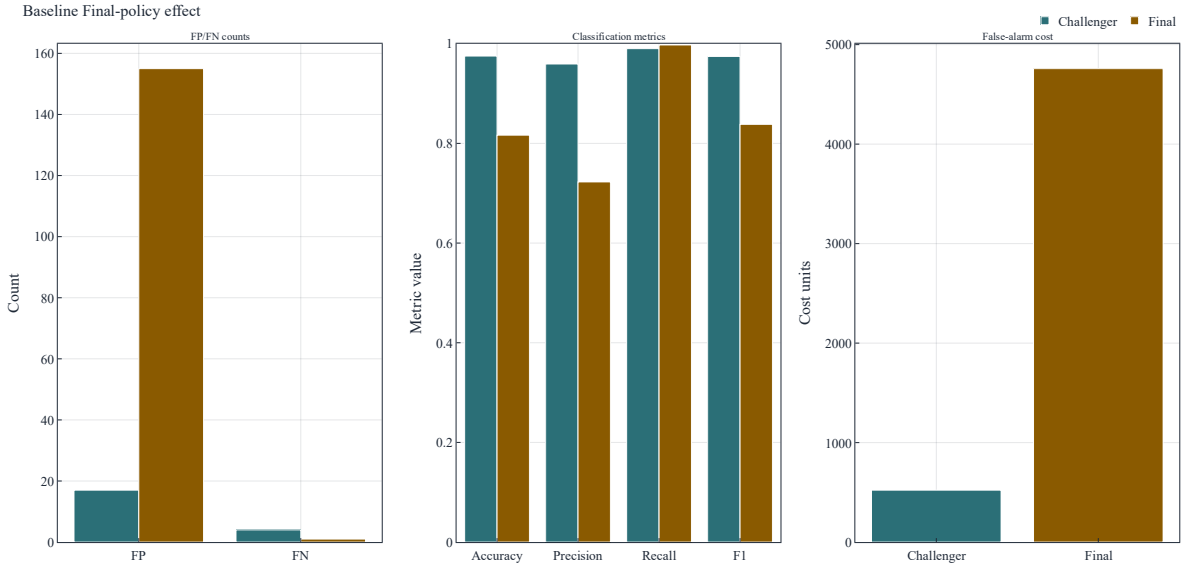


Figure 56. Baseline operating effect of the Final policy, showing the shift from the Challenger's raw output to the deployed Final output.

In the retrain-stage run, the same policy logic remained visible. The Final output produced TN = 274, FP = 171, FN = 0, and TP = 405, corresponding to accuracy = 0.7988, precision = 0.7031, recall = 1.0000, and F1 = 0.8257. The most striking outcome is that the deployed decision eliminated missed defects completely. However, this perfect recall was achieved by accepting the largest false-positive burden of the three outputs.

Relative to the Challenger, the Final output reduced false negatives from 7 to 0, but increased false positives from 104 to 171. The policy removed 7 missed defects at the cost of 67 additional unnecessary rejections. In terms of the summary metrics, recall rose from 0.9827 to 1.0000, but precision fell from 0.7928 to 0.7031, and F1 fell from 0.8776 to 0.8257. The same structural trade-off therefore remained present under retraining-enabled live operation: the policy pushed the system towards stronger protection and lower tolerance of uncertain acceptance, but it did so by sacrificing classification balance. The retrain-stage policy effect is summarised in Table 47 and visualised in Figure 57.

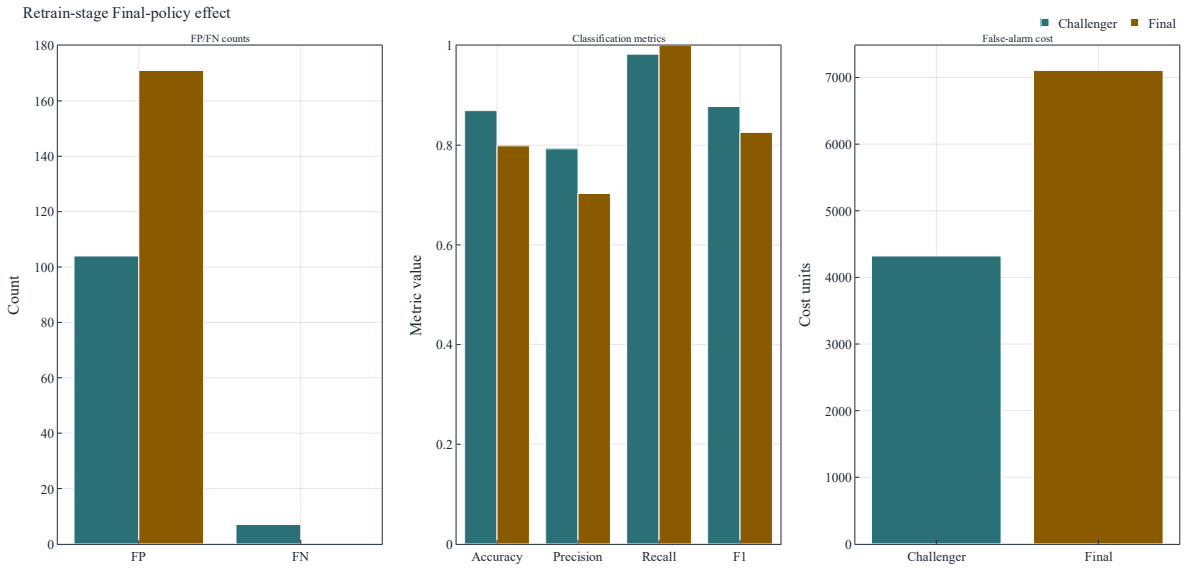


Figure 57. Retrain-stage operating effect of the Final policy, showing the shift from the Challenger's raw output to the deployed Final output.

Relative to the Champion, the Final output again remained closer in rejection behaviour than in overall balance. The Champion produced FP = 149 and FN = 10, while the Final output produced FP = 171 and FN = 0. The policy, therefore, added 22 false positives relative to the Champion, but removed all 10 missed defects. This confirms that the Final output was not a passive compromise between the two models. It was an active release rule that moved the operating point toward maximum protection when the policy judged caution preferable.

Table 47 summarises the retrain-stage change from the Challenger's raw output to the deployed Final output, showing how the locked policy eliminated missed defects while increasing false positives and reducing overall balance.

Table 47. Retrain-stage comparison of the Challenger and Final output under true labels, showing the policy-induced change in FP, FN, and derived classification metrics.

Output	FP	FN	Accuracy	Precision	Recall	F1
Challenger	104	7	0.8694	0.7928	0.9827	0.8776
Final	171	0	0.7988	0.7031	1.0000	0.8257
Delta (Final - Challenger)	+67	-7	-0.0706	-0.0897	+0.0173	-0.0519

When the two runs are read together, the Final policy shows a highly consistent role. In the baseline run, it reduced false negatives from 4 to 1 relative to the Challenger but increased false positives from 17 to 155. In the retrain-stage run, it reduced false negatives from 7 to 0 but increased false positives from 104 to 171. In both cases, the policy shifted the released decision towards stronger defect protection and heavier rejection. The numerical scale of the shift changed between runs, but its direction remained the same.

This consistency is important because it shows that the deployed behaviour did not drift unpredictably when retraining was enabled. The policy remained active in both live conditions and continued to impose a distinct operating posture that was clearly separate from the strongest raw model path. That is one of the key governance results in the chapter: the Final output did not become dominated by the Challenger even when the Challenger remained the strongest raw route. The deployed decision retained its own identity as a policy-controlled release layer.

This behaviour should not be interpreted as a failure of the Final policy. The policy was not designed to reproduce the best raw model metrics, but to enforce a conservative release rule under uncertainty. Its value lies in making the operational trade-off explicit, such as fewer missed defects are obtained by accepting a heavier false-positive and inspection burden. This distinction is important because the governed real-time system is evaluated not only in terms of classification performance, but also as a controlled decision-support architecture whose released output must remain stable, interpretable, and aligned with the chosen risk posture.

The processing-weighted false-positive burden makes the same conclusion visible in operational terms. In the baseline run, the Final processing-weighted false-positive burden was 4758.5869, compared with 521.9095 for the Challenger. In the retrain-stage run, the Final processing-weighted false-positive burden was 7106.4479, compared with 4322.0502 for the Challenger. The Final output therefore had approximately 9.12 times the Challenger's burden in the baseline run and approximately 1.64 times its burden in the retrain-stage run. This is not a minor side effect. It is a measurable operational consequence of conservative deployment. The policy provided additional protection against defect escape, but it did so by preserving a materially heavier burden of unnecessary rejection than the Challenger. The cross-run policy impact on error balance and false-alarm burden is summarised in Table 48 and illustrated in Figure 58.

Table 48. Cross-run summary of Final policy impact relative to the Challenger, showing changes in FP, FN, precision, recall, F1, and processing-weighted false-positive burden.

Run	Delta FP	Delta FN	Delta precision	Delta recall	Delta F1	Δ Processing-weighted false-positive burden
Baseline	+138	-3	-0.2366	+0.0074	-0.1363	+4236.6774
Retrain-stage	+67	-7	-0.0897	+0.0173	-0.0519	+2784.3977

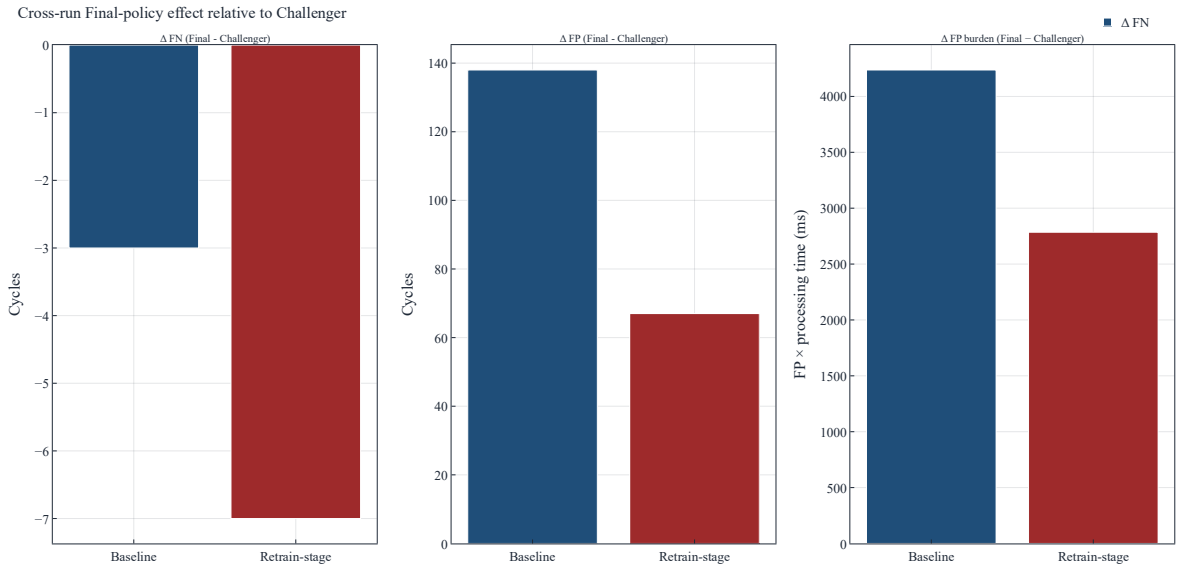


Figure 58. Cross-run comparison of the final policy effect on false negatives, false positives, and false-alarm burden.

The overall conclusion is clear. The Final output was not designed to maximise statistical classification performance or minimise processing-weighted false-positive burden. Its function was to preserve a cautious release posture under disagreement, and the measured results show that it did so consistently in both live runs. It reduced or eliminated missed defects, but only by accepting a substantially higher false-positive burden and a substantially higher processing-weighted false-positive burden than the Challenger. The significance of this result is therefore not that the policy was universally optimal, but that its operating effect was stable, interpretable, and fully visible in the live evidence.

8.6 Drift behaviour across the live run

This section examines the distributional drift evidence associated with the retrain-stage run and explains how it should be interpreted alongside the true-label and retraining results. The intention is not to claim that distributional change explains every performance difference or independently determines when model adaptation should occur, but to characterise how the monitored process distributions evolved across the run and to distinguish this statistical evidence from the separate governance conditions controlling retraining.

The PSI and KS timeline in Figure 59 was produced as a post-run distributional analysis of the accepted cycle sequence. The earliest 283 cycles of the 850-cycle run were used as the reference distribution, corresponding to the first third of the available sequence, and subsequent behaviour was evaluated using a 200-cycle rolling window with a 33-cycle step. PSI and KS were calculated separately for the six monitored event-aligned features and then combined using their arithmetic mean at each plotted window. PSI was used to quantify changes in the relative occupancy of the reference and comparison distributions, while KS provided a complementary non-parametric measure based on the maximum separation between their empirical cumulative distributions. The two measures were therefore used as complementary descriptions of distributional movement rather than interchangeable quantities. No PSI or KS threshold was used to authorise retraining, and the reference length, rolling-window size and sampling step were analytical configuration choices rather than values established through statistical optimisation.

The runtime drift monitor was separate from this post-run PSI/KS analysis. It compared the mean of the most recent 30 cycles with the first 50 cycles used as a reference and expressed the difference relative to the reference standard deviation. The configured criterion was an absolute z-score of at least 3.0 in at least two monitored features, sustained across three consecutive checks, with an 80-cycle cooldown following retraining. These values formed the preconfigured runtime monitoring logic and were engineering configuration choices rather than empirically optimised thresholds. The recorded *zscore_max* value of 7.4676 at cycle 745 was therefore an observed maximum standardised shift, not the configured drift threshold. Moreover, the evaluated run used the governed manual retraining pathway, while automatic retraining was disabled. Statistical drift evidence and retraining authority are consequently treated as distinct: drift

indicators identify distributional change, whereas model updating requires a separate governed admission and execution process.

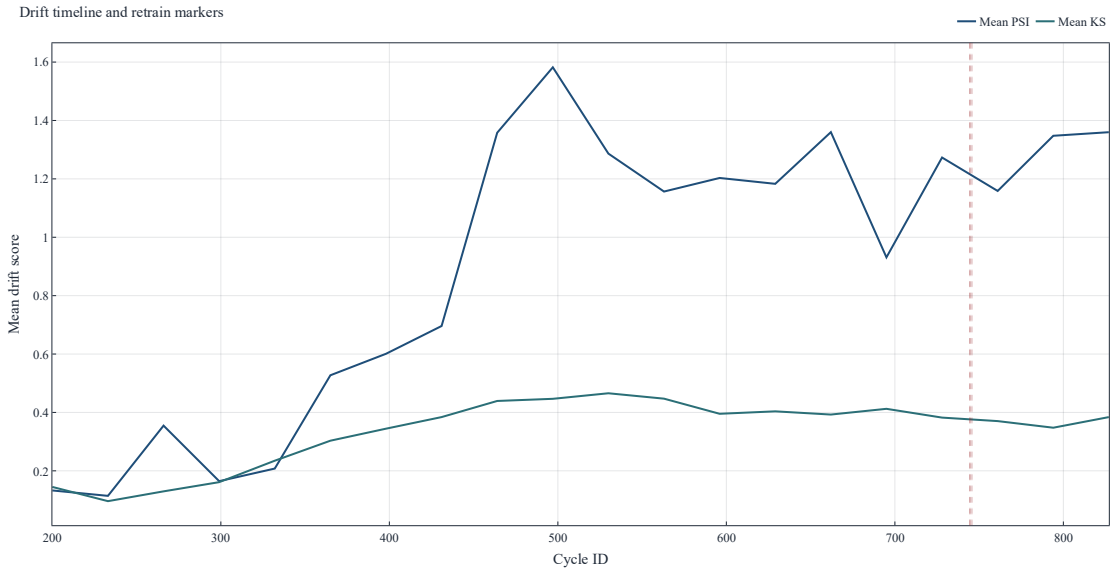


Figure 59. Drift timeline across rolling windows, showing mean PSI (blue) and mean KS (red) against CycleID, with the retraining event marked.

The drift evidence in Figure 59 shows that distributional movement varied across the evaluated sequence. During the earlier part of the displayed run, approximately Cycle 200–350, Mean PSI ranged from 0.1142 to 0.3548 and Mean KS from 0.0961 to 0.2347. These values indicate comparatively limited distributional separation during the early portion of the analysis.

During the middle portion, approximately Cycle 350–520, the distributional differences became more pronounced. Mean PSI ranged from 0.5273 to 1.5821, while Mean KS ranged from 0.3032 to 0.4466. The maximum Mean PSI observed across the complete plotted sequence was 1.5821 at cycle 497. Mean KS also increased during this period and reached its overall maximum of 0.4657 at cycle 530. The strongest excursions in the two summary statistics therefore occurred within the middle part of the evaluated sequence rather than immediately before the retraining event.

Distributional differences remained evident during the later portion of the run. Across the plotted points between approximately CycleID 600 and 850, Mean PSI ranged from 0.9311 to 1.3601 and Mean KS from 0.3477 to 0.4124. Although these later values remained elevated relative to the early part of the sequence, neither statistic exceeded its earlier maximum. The retraining event at cycle 745

consequently occurred during a period of continuing distributional change, but not at the strongest PSI or KS excursion observed in the plotted analysis.

The drift timeline should also be interpreted alongside the true-label behaviour. In both runs, the Champion remained relatively stable as a fixed reference, while the Challenger and Final output showed different trade-offs between defect capture and unnecessary rejection. The distributional evidence adds context to these results by showing that the monitored process state was not uniform throughout the run. It does not, however, establish that changes in PSI or KS directly produced the observed differences in predictive behaviour.

The non-monotonic form of the drift timeline is also important. Distributional separation increased and decreased across successive windows rather than following a continuous upward trend. This reinforces the need to interpret individual excursions within the wider sequence rather than treating a single high value as sufficient evidence of a persistent regime change. Table 49 summarises the early, middle, and late portions of the evaluated sequence together with the global Mean PSI and Mean KS maxima.

Table 49 summarises the main PSI and KS drift behaviour across the live run, separating the early, middle, and late run segments and identifying the relationship between the strongest PSI excursion and the governed retraining event.

Table 49. Summary of PSI and KS drift behaviour across the live run, showing early-, middle-, and late-stage drift characteristics and the relationship between the main PSI excursion and the retraining event.

Run segment	Cycle span	Observed PSI range	Observed KS range	Interpretation
Early phase	200–350	0.1142–0.3548	0.0961–0.2347	Distributional movement remained relatively low to moderate during the early part of the evaluated sequence.
Middle phase	350–520	0.5273–1.5821	0.3032–0.4466	Distributional movement increased substantially, including the maximum Mean PSI observed in the analysis.
Late phase	600–850	0.9311–1.3601	0.3477–0.4124	Elevated distributional differences remained visible, although the largest Mean PSI excursion had occurred earlier.
Global peak relationship	Full plotted sequence	Peak 1.5821 at cycle 497	Peak 0.4657 at cycle 530	The strongest Mean PSI and Mean KS values occurred before the recorded retraining event at cycle 745.

In summary, Figure 59 and Table 49 show that the live operating condition was not distributionally uniform across the run. The strongest Mean PSI and Mean KS excursions occurred during the middle part of the evaluated sequence, while elevated distributional differences remained visible during the later portion of the run. The retraining event at cycle 745 therefore occurred within an operating environment in which measurable distributional change was present, but it did not coincide with the maximum PSI or KS values observed in the plotted analysis. The PSI and KS results should consequently be interpreted as contextual evidence of changing feature distributions rather than as direct evidence that statistical drift triggered the retraining event. This distinction provides the basis for the next section, which examines the retraining event itself and its governed meaning within the live evidence chain.

8.7 Recorded retraining outcome and its governed meaning

The previous section showed that the live run did not proceed under one uniform distribution condition. The PSI and KS analysis identified measurable changes in the monitored feature distributions across the sequence, with the largest Mean PSI and Mean KS excursions occurring before the recorded retraining event. This section now turns to the retraining event itself and explains what was actually recorded, what changed operationally, and why this event matters in the broader thesis argument. The purpose is not merely to state that retraining occurred, but to interpret the recorded outcome of that governed update within the live evidence chain while keeping the statistical drift evidence distinct from the authority to initiate model adaptation.

The retraining-enabled run recorded one successful Challenger retraining event, and that event occurred at cycle 745. The same run included 40 human corrections, with no blocked retrain attempt recorded. The absence of a blocked retraining event in this successful run does not weaken the governance interpretation; it indicates that the recorded update occurred only after the required conditions had been met, rather than by bypassing the governed path. These values matter because they define the update as a governed and traceable live event rather than as an unobserved offline model replacement. The run remained complete at 850 accepted cycles with 850 truth-linked rows, so the retraining event was embedded within a full evaluable live record rather than

inserted through a separate external experiment. Table 50 summarises the recorded retraining outcome, including the correction count, successful retraining event, retraining cycle, training rows used, class representation in the selected training data, recorded drift metric, observed drift value, and final outcome.

Table 50. Recorded retraining outcome, showing correction count, retraining count, retraining cycle, and run integrity context.

Stage mode	Accepted cycles	Truth-linked	Successful retrain	Retrain cycle	Human corrections	Training rows used	Training positives	Training negatives	Recorded drift metric	Observed drift value	Outcome
human	850	850	1	745	40	745	390	355	*zscore_max*	7.4676	SUCCESS

The first point to establish is that the retraining event was operationally successful in the narrow governance sense. In this study, "successful retraining" does not mean that every whole-run aggregate metric must necessarily exceed the fixed-condition baseline. Instead, it means that the system admitted the update under the governed conditions, completed the update inside the live operating path, and preserved cycle continuity, truth linkage, and decision generation after the event. In other words, the successful event demonstrates not only that a Challenger update occurred, but that the update crossed the platform's admission and execution boundary correctly and remained inside the same live evidence chain. On that basis, the recorded outcome was successful. The update was accepted, executed, and retained inside the live evidence chain without breaking the run.

This distinction is important because it prevents the event from being interpreted too simplistically. A governed retraining outcome should not be judged only by whether the later run "beats" the earlier run in all summary statistics. The retraining-enabled run and the baseline run serve different evaluative purposes. The baseline run establishes the fixed-condition reference state. The retraining-enabled run demonstrates that the same governed environment can admit a controlled Challenger update while remaining operational. The success of the retraining event lies first in the correctness of the transition itself, not merely in a single aggregate score comparison.

The second point is that the retraining event did not destabilise the live decision architecture. After retraining was admitted, the three-path structure remained intact. The Champion remained stable as the fixed full-curve reference. The Challenger continued to provide the strongest raw operating balance among the three outputs in the retrain-stage run, with accuracy = 0.8694, precision = 0.7928, recall = 0.9827, and F1 = 0.8776. The Final output remained conservative, achieving recall = 1.0000 while carrying the highest false-positive burden. This continuity of role is a key governance result. The retraining event did not collapse the distinction between benchmark, operational path, and policy-released decision. The same architecture remained visible after the update.

The third point concerns how the retraining event should be interpreted alongside the drift evidence. Figure 60 positions the recorded retraining event within the same distributional timeline examined in Section 8.6. The strongest Mean PSI and Mean KS excursions occurred earlier in the run, at cycles 497 and 530 respectively, while elevated distributional differences remained visible during the later portion of the sequence. The retraining event at cycle 745 therefore occurred within a changing distributional environment, but not at the maximum of either plotted drift statistic. This distinction is important because temporal coexistence between distributional change and model adaptation should not be interpreted as evidence that the PSI or KS values directly triggered the update. Instead, Figure 60 provides contextual evidence showing where the governed retraining event occurred relative to the evolving process distributions. The statistical drift analysis and the retraining event are therefore complementary but distinct parts of the operational evidence.

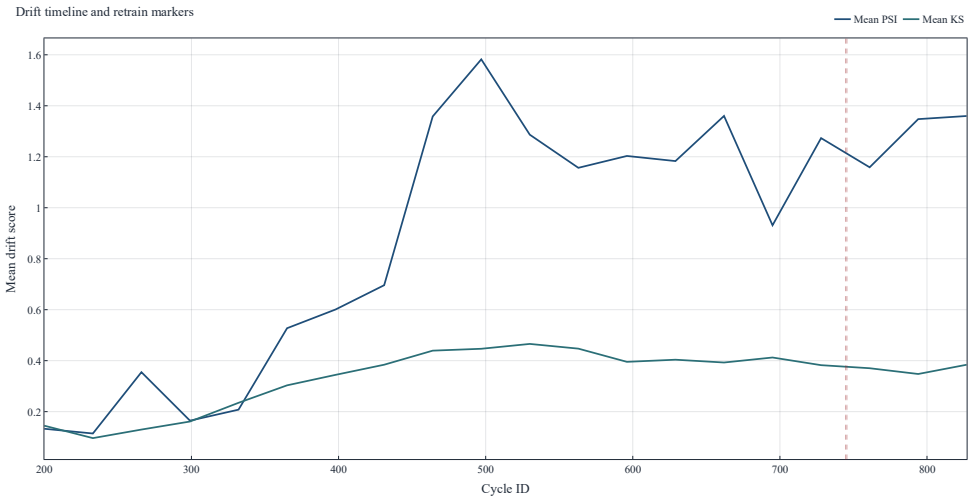


Figure 60. Retraining event positioned within the drift timeline, showing the temporal relationship between monitored distributional change and the recorded retraining cycle.

The fourth point is that the retraining event preserved practical live feasibility. The retrain-stage run remained fully truth-linked, and its timing results remained within a cycle-level response window, with mean processing time = 41.5582 ms and p95 processing time = 60.1806 ms, as reported earlier in Section 8.3 and summarised in Table 43. This matters because a retraining event is only meaningful for manufacturing deployment if it does not push the system out of practical live use. In the present evidence, the update was admitted without producing a timing profile that would make the run operationally unusable. The platform remained responsive after the event.

The fifth point concerns what the retraining event adds to the thesis argument. The baseline run already established that the event-anchored route could strongly outperform the fixed full-curve reference under stable live conditions. If the chapter ended there, the thesis would show a strong live predictive system, but not a governed adaptive one. The retraining event adds a more demanding claim: the platform can remain complete, controlled, and traceable while allowing the operational Challenger path to be updated in live use. This is why the recorded retraining outcome matters even if the whole-run aggregate Challenger metrics in the retrain stage are lower than in the baseline run. The point of the second run is not only to maximise the accuracy number. It is to show that the governed system can change state without losing control.

A further implication follows from the presence of 40 human corrections. The retraining event was not a blind automatic response to a monitoring statistic alone. It occurred within a framework where the reviewed correction input formed part of the live evidence. This strengthens the governance meaning of the event because it shows that the update was connected to observed and reviewed operational evidence, not merely to an opaque automatic trigger. In practical terms, the retraining outcome therefore represents a supervised and accountable adaptive transition rather than an uncontrolled self-modification of the live system. Table 51 summarises the governed meaning of the recorded retraining outcome, linking the successful update to operational execution, architectural continuity, drift context, and live feasibility.

Table 51. Governed interpretation of the recorded retraining outcome, distinguishing operational success, architectural continuity, drift context, and live feasibility.

Aspect	Recorded evidence	Interpretation
Operational success	One successfully governed Challenger retrain was recorded at cycle 745 with outcome SUCCESS.	The retraining pathway was executed successfully rather than remaining only a design claim.
Architectural continuity	The run completed 850 accepted cycles with 850 truth-linked cycles after the retrain event.	Live operation continued after retraining without breaking cycle accounting or truth linkage.
Drift context	The retraining event was recorded with an accompanying *zscore_max* value of 7.4676. The post-run PSI/KS analysis showed maximum Mean PSI at cycle 497 and maximum Mean KS at cycle 530, while elevated distributional differences remained visible during the later part of the run.	The recorded retraining event occurred within a drift-aware operating context, but the PSI/KS analysis and the observed *zscore_max* value should not be interpreted as independent authority for the update.
Live feasibility	Mean processing time was 41.5582 ms, p95 processing time was 60.1806 ms, and Challenger mean latency was 3.7732 ms.	The run remained feasible as a live cycle-level system while the adaptive branch was active.

Taken together, the recorded retraining outcome demonstrates that the governed real-time platform can admit and complete one controlled Challenger retraining transaction without sacrificing the core properties already established in the baseline run. The runtime record remained complete, and the three-path architecture remained interpretable. The retraining transaction occurred within the recorded monitoring context, as illustrated in Figure 60 and summarised in Tables 49 and 50, while timing remained practical. The retraining outcome matters because it demonstrates that the governed retraining pathway can produce a new versioned Challenger artefact within the same operational evidence structure. However, the preserved implementation evidence does not conclusively establish that the running inference process loaded the replacement estimator for subsequent cycles. The result is therefore interpreted as evidence of successful governed retraining execution and traceability rather than as prospective performance evidence for the newly trained model.

8.8 Timing and real-time feasibility

This section reports the timing evidence recorded during the two live runs and explains what it implies about real-time feasibility at the cycle level. The central question is whether the governed platform produced decisions quickly and consistently enough to be used in live manufacturing conditions, not merely

whether inference was fast in a laboratory sense. For this reason, the discussion focuses on end-to-end processing time and its tail behaviour, because tail delays are often more operationally relevant than best-case latency.

In the baseline run, the mean processing time was 30.7006 ms, and the p95 processing time was 34.5781 ms. These values show that most cycle decisions were produced in a very small fraction of a second, while the slower tail remained well below the seconds-scale duration of the manufacturing cycle. This is a strong practical result because the processing window is small relative to typical injection-moulding cycle times measured in seconds. The baseline timing evidence supports the claim that the system can operate as a cycle-level decision layer without requiring delayed evaluation after production has moved on.

The model-level timing also remained small. Mean per-cycle model latency was 0.5986 ms for the Champion and 2.3602 ms for the Challenger. The Challenger introduced an additional model latency of approximately 1.7616 ms relative to the Champion. This difference is expected because the two routes differ in representation and evaluation burden. However, even the Challenger remained comfortably within a low-millisecond range. The practical implication is that the dominant portion of the end-to-end processing time lies not in model execution alone but in the full governed processing chain, including representation, validation, logging, and decision release. The baseline run demonstrates not only that inference is feasible, but that the full governed decision workflow remained feasible. The baseline timing profile, including processing time and model-level latency for the Champion and Challenger, is summarised in Table 52 and illustrated in Figure 61.

Table 52. Baseline timing summary, showing mean processing time, p95 processing time, and mean model latency for Champion and Challenger.

phase	metric	count	mean	median	p95	min	max
Cycle 1-200	Champion latency	200	0.618236	0.510349986	1.130455031	0.453599961	2.704399987
Cycle 1-200	Challenger latency	200	2.3720505	2.097600023	3.489475008	1.93160004	5.53580001
Cycle 1-200	Total inference latency	200	2.9902865	2.690999972	4.194065032	2.407300053	6.866100011
Cycle 1-200	Total processing time	200	30.6403505	30.18574999	34.47121497	28.017	45.8959

phase	metric	count	mean	median	p95	min	max
Cycle 201-400	Champion latency	200	0.581562	0.494649983	1.059755031	0.454600027	2.118400007
Cycle 201-400	Challenger latency	200	2.458120002	2.049350005	3.279825	1.897600014	34.43970002
Cycle 201-400	Total inference latency	200	3.039682002	2.590150019	4.053930016	2.406899992	35.35150003
Cycle 201-400	Total processing time	200	30.520439	29.87299999	34.01741002	26.70499997	65.4656
Cycle 401-600	Champion latency	200	0.595072002	0.500199967	1.150165024	0.455099973	1.412000041
Cycle 401-600	Challenger latency	200	2.2704715	2.043449989	3.176139962	1.90849998	5.671499996
Cycle 401-600	Total inference latency	200	2.865543503	2.630450006	3.827259984	2.395100018	6.199900003
Cycle 401-600	Total processing time	200	30.6859925	30.15749998	34.47475499	27.5038	41.23700003
Cycle 601-800	Champion latency	200	0.6085685	0.509100035	1.158375019	0.456399983	2.234900021
Cycle 601-800	Challenger latency	200	2.345325998	2.043949993	3.696370029	1.891000022	7.151600032
Cycle 601-800	Total inference latency	200	2.953894498	2.608949988	4.215445009	2.396799973	8.377400052
Cycle 601-800	Total processing time	200	30.9125095	30.51780001	35.08222001	26.89039998	41.08639999
Cycle 801-850	Champion latency	50	0.563033997	0.490849983	0.963494991	0.462400028	1.197999984
Cycle 801-850	Challenger latency	50	2.339951992	2.037099999	3.481039978	1.907899976	5.858100019
Cycle 801-850	Total inference latency	50	2.902985989	2.553899976	4.17087996	2.407599997	6.406700006

phase	metric	count	mean	median	p95	min	max
Cycle 801-850	Total processing time	50	30.872368	30.40599998	34.29864	28.22440001	40.21069995

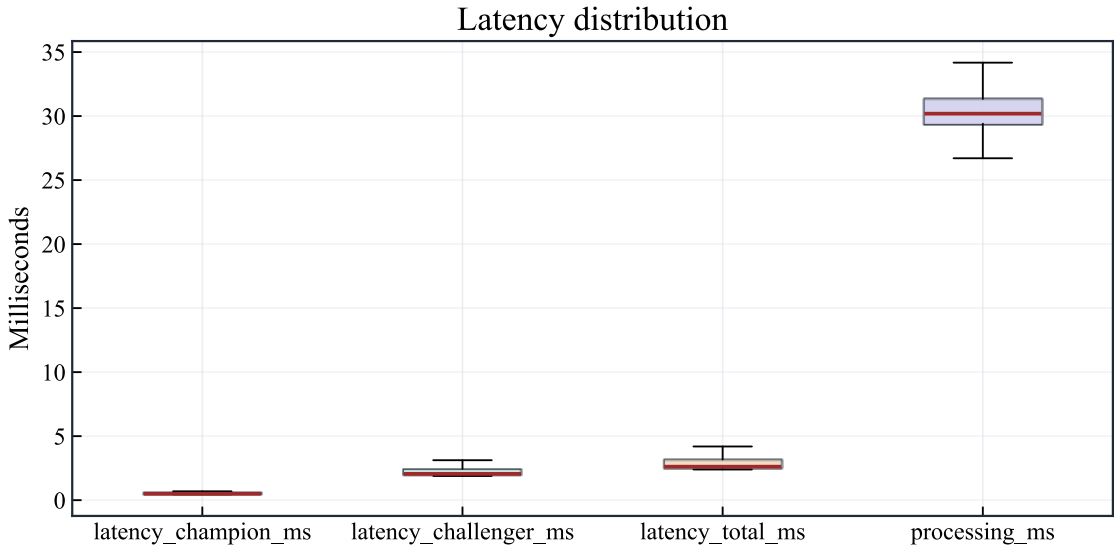


Figure 61. Baseline processing-time distribution and tail behaviour.

In the retrain-stage run, the mean processing time was 41.5582 ms, and the p95 processing time was 60.1806 ms. Compared with the baseline, this run showed a higher mean processing time and a higher p95 processing time. The mean increased by approximately 10.8576 ms, and the p95 increased by approximately 25.6025 ms. The important interpretation is not that retraining improved timing. Rather, it is that enabling the governed retraining branch did not impose a timing burden large enough to undermine live cycle-level feasibility. The retrain-stage run remained fully responsive and operationally feasible for live use. The retrain-stage timing profile is summarised in Table 53 and illustrated in Figure 62.

At the model level, mean per-cycle latency was 0.7931 ms for the Champion and 3.7732 ms for the Challenger, as summarised in Table 53. The Challenger introduced an additional model latency of approximately 2.9801 ms relative to the Champion in this run. As shown in Figure 62, these values remain small in absolute terms and support the feasibility of running both predictive routes live without compromising responsiveness.

Table 53. Retrain-stage timing summary, showing mean processing time, p95 processing time, and mean model latency for Champion and Challenger.

phase	metric	count	mean	median	p95	min	max
Cycle 1-200	Champion latency	200	0.669816999	0.537449989	1.045719982	0.455700036	16.96419995
Cycle 1-200	Challenger latency	200	2.664263501	2.205449971	4.800504982	2.006099967	11.0384
Cycle 1-200	Total inference latency	200	3.3340805	2.791700012	5.358644965	2.533200022	28.00259995
Cycle 1-200	Total processing time	200	34.2225435	29.78725001	56.75970002	27.22640004	93.09519996
Cycle 201-400	Champion latency	200	0.832781998	0.772699976	1.241285016	0.449899992	1.678199973
Cycle 201-400	Challenger latency	200	4.211844	3.821349994	6.688464974	2.213199972	8.935699996
Cycle 201-400	Total inference latency	200	5.044625998	4.625000001	7.900244958	2.699300006	9.593900002
Cycle 201-400	Total processing time	200	43.28668	42.89864999	59.38992504	22.51320001	77.30979996
Cycle 401-600	Champion latency	200	0.853329501	0.804049982	1.306724994	0.452100008	2.187499951
Cycle 401-600	Challenger latency	200	4.173871001	3.71530003	7.378909961	2.162999997	9.078499977
Cycle 401-600	Total inference latency	200	5.027200502	4.600649991	8.448849997	2.635399986	10.28499997
Cycle 401-600	Total processing time	200	41.15893	40.61629999	54.64194503	24.35690002	67.54050002
Cycle 601-800	Champion latency	200	0.805702502	0.710099965	1.213050034	0.454500027	4.816800007

phase	metric	count	mean	median	p95	min	max
Cycle 601-800	Challenger latency	200	3.953427003	3.529850015	6.895119976	2.14430003	9.070600034
Cycle 601-800	Total inference latency	200	4.759129506	4.32019998	7.673925025	2.60040001	11.24949998
Cycle 601-800	Total processing time	200	44.464072	43.69245001	59.41489502	23.85270002	73.8517
Cycle 801-850	Champion latency	50	0.836589998	0.863799971	1.187869991	0.471800042	1.228399982
Cycle 801-850	Challenger latency	50	4.130593996	3.985200019	6.058640016	2.332600008	7.644700003
Cycle 801-850	Total inference latency	50	4.967183993	4.704499966	7.191215022	2.820499998	8.873099985
Cycle 801-850	Total processing time	50	53.960074	52.81494997	70.28900501	37.2511	74.84300004

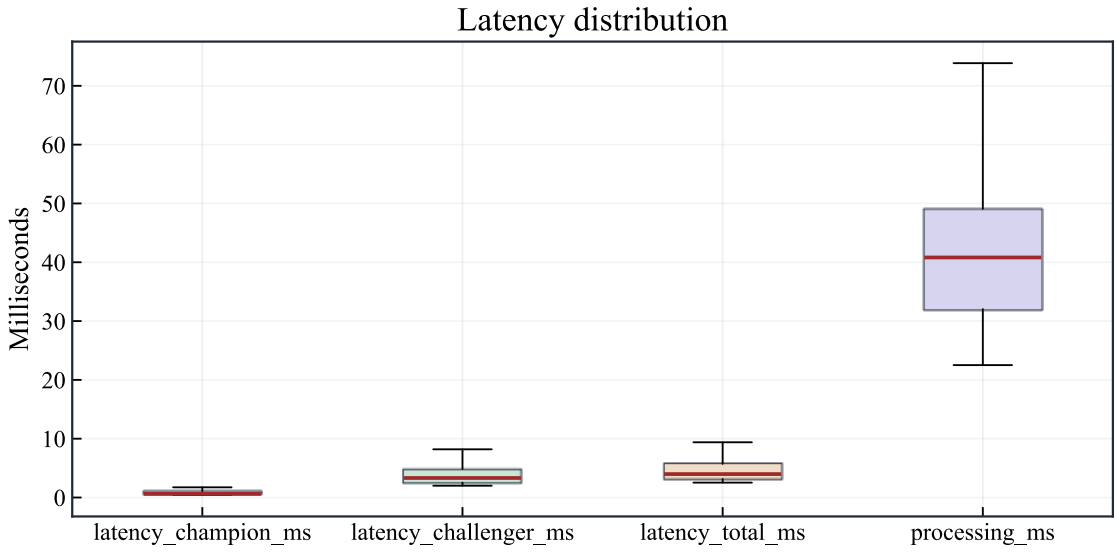


Figure 62. Retrain-stage processing-time distribution and tail behaviour.

When the two runs are read together, the timing evidence supports three conclusions. First, both runs fall comfortably within a cycle-level real-time envelope. Mean processing times remain in the tens of milliseconds and do not approach the seconds-scale cycle duration of the manufacturing process. Second,

the Challenger remains computationally heavier than the Champion, but the additional model latency remains small in both runs and does not dominate the end-to-end processing chain. Third, the retrain-stage run confirms that admitting a governed update does not force the system into delayed operation; real-time feasibility is retained even when the adaptive branch is active, and one retraining event is admitted.

A further point follows from the use of p95 rather than minimum or mean alone. Real-time readiness in manufacturing often depends on worst-case or near-worst-case behaviour. The baseline run p95 of 34.5781 ms shows that the system remained responsive even in the slower tail of the distribution. The retrain-stage p95 of 60.1806 ms likewise remains within a feasible live processing window in the second run. These values strengthen the claim that the platform can operate continuously without accumulating processing backlog at the cycle level.

Taken together, the timing evidence confirms that the platform's measured predictive and policy behaviour was produced within a feasible live processing window. The system operated as a cycle-level decision environment rather than as an offline analysis tool. This provides a necessary practical foundation for interpreting the remaining results, because it shows that the observed performance and policy trade-offs were achieved under real-time constraints rather than under relaxed offline conditions.

8.9 Cost-sensitive decision trade-offs

This section evaluates the operational consequences of classification errors using two distinct measures. The first is the normalised relative decision loss introduced in Section 7.5, defined by $L_r = (FP + rFN)/N$. This measure evaluates the combined effect of false-positive and false-negative errors under an assumed relative penalty ratio r, with the result expressed on a per-cycle basis through division by the number of truth-labelled cycles, N. The corresponding sensitivity results are presented in Tables 54 and 55 and Figures 63 and 64.

The evaluated penalty ratios were selected as illustrative cost-sensitive scenarios, spanning equal weighting through progressively stronger emphasis on false-negative errors. They were used to examine the sensitivity of the decision-path comparison to different relative error consequences rather than being derived from plant-specific financial measurements.

The second measure is the processing-weighted false-positive burden used for cross-run comparison. It is calculated as the number of false-positive decisions multiplied by the measured mean end-to-end processing time for the corresponding run. The false-positive count is obtained from the truth-labelled classification results, while the mean processing time is obtained from the runtime timing measurements reported in Section 8.8. The resulting quantity is therefore a processing-weighted indicator of unnecessary rejection burden rather than a financial cost. Within each run, the same run-level mean processing time is applied to the Champion, Challenger, and Final outputs, so differences between the three decision paths primarily reflect their different false-positive counts. Across the two runs, the measure also reflects the difference in mean processing time between the baseline and retrain-stage conditions. This quantity is separate from L_r it is not normalised by the number of cycles and does not include the false-negative penalty ratio r. The corresponding cross-run results are presented in Table 56 and Figure 65.

In industrial terms, the normalised relative decision loss represents the consequences of classification errors under an assumed penalty structure rather than measured financial expenditure. A false positive corresponds to a good part being classified as defective and may lead to unnecessary rejection, inspection, or intervention, while a false negative corresponds to a defective part being classified as good and may allow it to progress further through production. Actual financial consequences would depend on the specific factory, product, inspection procedure, downstream process, and consequence of defect escape, and were not measured in this study.

In the baseline run, the normalised relative decision loss changes according to the assumed false-negative to false-positive penalty ratio, r. When $r = 1$, false positives and false negatives are weighted equally, and the resulting loss is equivalent to the overall misclassification rate. Under this condition, the Champion, Challenger, and Final outputs produced L_1values of 0.1894, 0.0247, and 0.1835, respectively. As r increases, false negatives contribute progressively more strongly to the loss. The Challenger retains the lowest loss across the tested range from $r = 1$ to $r = 30$, reflecting its combination of relatively few false positives and false negatives. At the highest tested ratio, $r = 50$, the Final output produces a slightly lower loss than the Challenger, 0.2412 compared with 0.2553, because the Final policy reduces the number of false negatives from 4 to 1. This result illustrates the intended purpose of the sensitivity analysis: the preferred

decision path depends on the assumed relative consequence of the two error types rather than on a single fixed cost assumption. The complete baseline sensitivity results are reported in Table 54 and illustrated in Figure 63.

Table 54. Baseline normalised relative decision-loss sensitivity for Champion, Challenger, and Final output across assumed FN:FP penalty ratios.

Model	$\frac{c_{FN}}{c_{FP}}$	N_{truth}	FP	FN	$\bar{L}_{cycle}$
champion	1	850	152	9	0.189411765
champion	2	850	152	9	0.2
champion	3	850	152	9	0.210588235
champion	5	850	152	9	0.231764706
champion	8	850	152	9	0.263529412
champion	10	850	152	9	0.284705882
champion	15	850	152	9	0.337647059
champion	20	850	152	9	0.390588235
champion	30	850	152	9	0.496470588
champion	50	850	152	9	0.708235294
challenger	1	850	17	4	0.024705882
challenger	2	850	17	4	0.029411765
challenger	3	850	17	4	0.034117647
challenger	5	850	17	4	0.043529412
challenger	8	850	17	4	0.057647059
challenger	10	850	17	4	0.067058824
challenger	15	850	17	4	0.090588235
challenger	20	850	17	4	0.114117647
challenger	30	850	17	4	0.161176471
challenger	50	850	17	4	0.255294118
final	1	850	155	1	0.183529412
final	2	850	155	1	0.184705882
final	3	850	155	1	0.185882353
final	5	850	155	1	0.188235294
final	8	850	155	1	0.191764706
final	10	850	155	1	0.194117647

Model	$\frac{c_{FN}}{c_{FP}}$	N_{truth}	FP	FN	$\bar{L}_{cycle}$
final	15	850	155	1	0.2
final	20	850	155	1	0.205882353
final	30	850	155	1	0.217647059
final	50	850	155	1	0.241176471

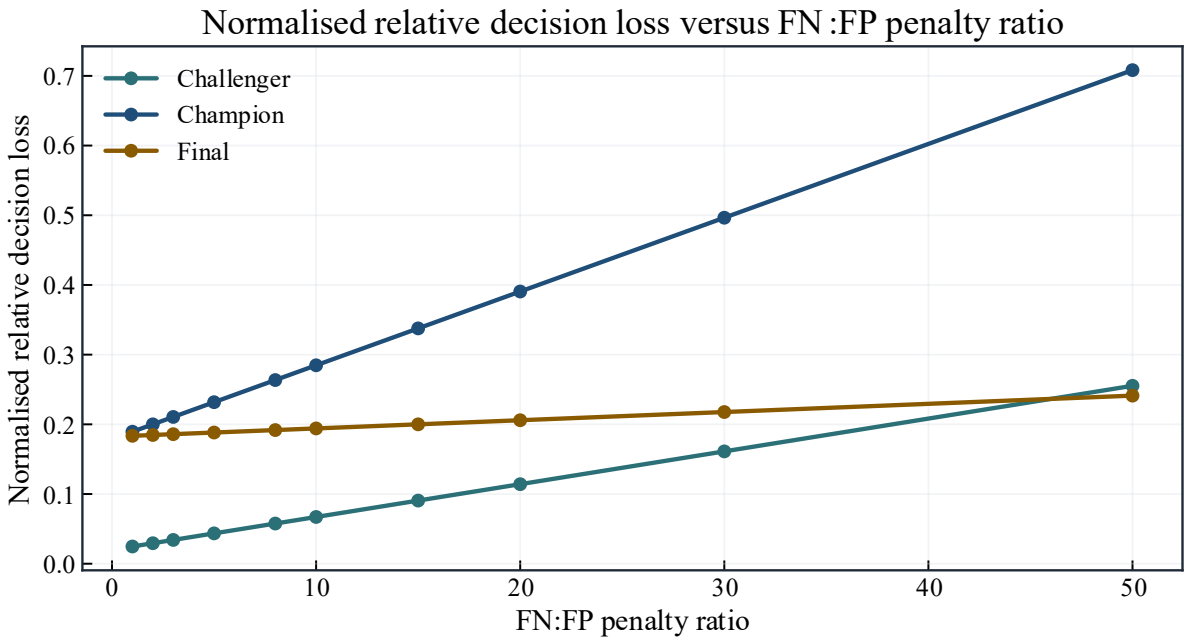


Figure 63. Baseline normalised relative decision-loss sensitivity across assumed FN:FP penalty ratios.

In the retrain-stage run, the normalised relative decision loss again changes according to the assumed false-negative to false-positive penalty ratio, r. At $r = 1$, the Challenger produces the lowest loss, with L_1=0.1306, compared with 0.1871 for the Champion and 0.2012 for the Final output. The Challenger retains the lowest loss across the tested ratios from $r = 1$ to $r = 8$. However, from $r = 10$ onwards, the Final output produces the lowest normalised loss because it records no false negatives in this run. Its loss therefore remains constant at 0.2012 as r increases, whereas the losses of the Champion and Challenger increase because both retain false-negative errors. This result demonstrates the purpose of the sensitivity analysis particularly clearly: as the assumed consequence of a false negative increases, the conservative Final policy becomes increasingly favourable

under the defined loss function despite producing more false positives. The complete retrain-stage sensitivity results are reported in Table 55 and illustrated in Figure 64.

Table 55. Retrain-stage normalised relative decision-loss sensitivity for Champion, Challenger, and Final output across assumed FN: FP penalty ratios.

Model	$\frac{c_{FN}}{c_{FP}}$	N_{truth}	FP	FN	$\bar{L}_{cycle}$
champion	1	850	149	10	0.187058824
champion	2	850	149	10	0.198823529
champion	3	850	149	10	0.210588235
champion	5	850	149	10	0.234117647
champion	8	850	149	10	0.269411765
champion	10	850	149	10	0.292941176
champion	15	850	149	10	0.351764706
champion	20	850	149	10	0.410588235
champion	30	850	149	10	0.528235294
champion	50	850	149	10	0.763529412
challenger	1	850	104	7	0.130588235
challenger	2	850	104	7	0.138823529
challenger	3	850	104	7	0.147058824
challenger	5	850	104	7	0.163529412
challenger	8	850	104	7	0.188235294
challenger	10	850	104	7	0.204705882
challenger	15	850	104	7	0.245882353
challenger	20	850	104	7	0.287058824
challenger	30	850	104	7	0.369411765
challenger	50	850	104	7	0.534117647
final	1	850	171	0	0.201176471
final	2	850	171	0	0.201176471
final	3	850	171	0	0.201176471
final	5	850	171	0	0.201176471
final	8	850	171	0	0.201176471
final	10	850	171	0	0.201176471
final	15	850	171	0	0.201176471
final	20	850	171	0	0.201176471

Model	$\frac{c_{FN}}{c_{FP}}$	N_{truth}	FP	FN	$\bar{L}_{cycle}$
final	30	850	171	0	0.201176471
final	50	850	171	0	0.201176471

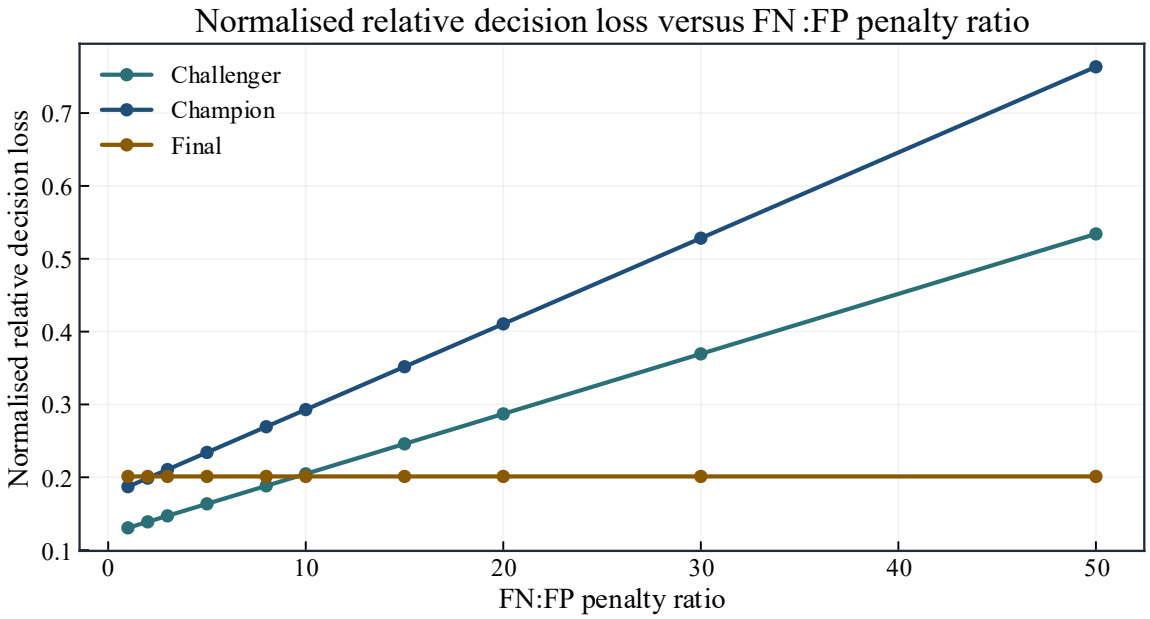


Figure 64.Retrain-stage normalised relative decision-loss sensitivity across assumed FN: FP penalty ratios.

The processing-weighted false-positive burden provides a separate cross-run indicator and should not be confused with the normalised relative decision loss reported in Tables 54 and 55. It is calculated as the number of false positives multiplied by the run-specific mean processing time. Unlike L_r, this measure is not normalised by the number of truth-labelled cycles and does not include the relative false-negative penalty r. From the baseline run to the retrain-stage run, the processing-weighted false-positive burden increased from 4666.4852 to 6192.1681 for the Champion, from 521.9095 to 4322.0502 for the Challenger, and from 4758.5869 to 7106.4479 for the Final output. These increases reflect both the false-positive counts and the higher mean processing time recorded in the retrain-stage run. The measure is therefore used only as an operational indicator of processing-weighted false-positive burden and should not be interpreted as monetary cost or as an alternative form of the normalised relative decision loss. The cross-run results are reported in Table 56 and illustrated in the corresponding cross-run figure 65.

Table 56. Cross-run change in processing-weighted false-positive burden for Champion, Challenger, and Final output.

Model	Baseline FP burden	Retrain-stage FP burden	Change (Retrain-stage – Baseline)	Direction
Champion	4666.4852	6192.1681	+1525.6828	Increase
Challenger	521.9095	4322.0502	+3800.1407	Increase
Final	4758.5869	7106.4479	+2347.8610	Increase

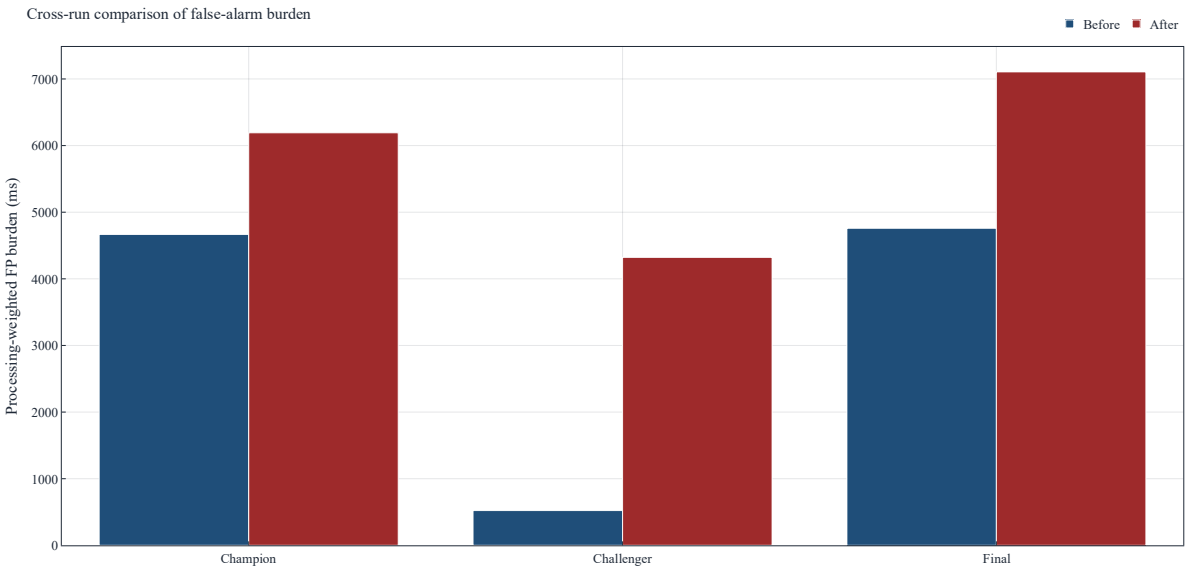


Figure 65. Cross-run comparison of processing-weighted false-positive burden (FP × mean processing time, ms) for the Champion, Challenger, and Final output.

The combined evidence clarifies the different operational roles of the three decision paths. The Challenger produces the lowest processing-weighted false-positive burden in both runs, reflecting its lower number of unnecessary rejections while maintaining strong defect detection. The Champion remains useful as a stable reference path but carries a higher false-positive burden than the Challenger. The Final output reflects a different operating objective. It is not designed to minimise false-positive burden, but to preserve a conservative release posture when the Champion and Challenger disagree. This reduces or eliminates missed defective cycles at the expense of additional false positives.

These burden results should be interpreted alongside the normalised relative decision-loss sensitivity in Tables 54 and 55, where the relative ordering of the decision paths changes as the assumed false-negative penalty r increases. The processing-weighted false-positive burden is summarised separately in Table 56 and Figure 65.

The difference between the Final output and the Challenger is particularly visible in the processing-weighted false-positive burden. In the baseline run, the Final output produced a burden of 4758.5869 compared with 521.9095 for the Challenger, making the Final value approximately 9.12 times higher. In the retrain-stage run, the corresponding values were 7106.4479 and 4322.0502, respectively, giving a smaller ratio of approximately 1.64. These differences quantify the additional false-positive-related operational burden associated with the conservative Final policy under the measured run conditions. They should not be interpreted as monetary or economic costs. Instead, they show that the reduction in missed defective cycles achieved by the Final policy is accompanied by increased unnecessary rejection and a higher processing-weighted false-positive burden.

The conservative posture reflects the risk priority adopted for the evaluated decision-support system. When the Champion and Challenger disagree, neither prediction is treated as sufficient evidence for releasing the cycle as acceptable; the disagreement is instead treated as unresolved model uncertainty. The Final policy therefore assigns the defective class in this situation, prioritising protection against defect escape while accepting the associated increase in false-positive decisions and unnecessary rejection. This represents an explicit operational trade-off rather than a claim that conservative rejection is universally preferable in manufacturing. Its suitability in practice would depend on the relative consequences of missed defects and unnecessary rejection for the particular product and production environment. The normalised relative decision-loss sensitivity in Tables 54 and 55 makes this dependence visible, while the processing-weighted false-positive burden in Table 56 and Figure 65 quantifies the additional rejection-related burden observed in the evaluated runs.

The combined sensitivity and burden evidence supports a clear operational conclusion. Across both live runs, the Challenger produced the lowest processing-weighted false-positive burden, while the Final output maintained the most conservative release posture. However, the normalised relative decision-loss results show that the preferred path under the defined loss function depends on

the assumed false-negative penalty r. As r increases, the reduced number of false negatives produced by the Final policy becomes increasingly influential and can outweigh its higher false-positive burden. The results therefore quantify an explicit trade-off between unnecessary rejection and protection against defect escape rather than identifying one universally lowest-cost decision path.

8.10 Overall interpretation of the live evidence

The results in this chapter show a consistent and interpretable pattern across the two governed live runs. The key finding is not confined to a single metric or a single figure. Rather, the evidence shows how the full decision architecture behaved under live conditions, and what trade-offs were created by retaining a fixed full-curve reference, operating an event-anchored path, and releasing decisions through a locked conservative policy.

The first point established by the results is the credibility of the live operating record itself. Both runs produced 850 accepted cycles with 850 truth-linked rows, and both had the same truth distribution (405 positive, 445 negative). This gives the chapter a strong evidential foundation. The reported behaviour is not derived from incomplete labelling, unequal run length, or a reduced subset. It is derived from two complete truth-linked live operating records executed through the same governed platform.

The second point concerns the event-anchored representation and the Challenger's practical strength. The alignment example in Figure 44 shows that the Challenger is not operating on an abstract or arbitrary feature shortcut. It evaluates a cycle snapshot tied to a meaningful physical reference event (the MP peak), with the synchronised state at that event recorded explicitly. Against that representation, the baseline results in Table 44 show that the Challenger outperformed the full-curve Champion strongly. It reduced false positives from 152 to 17 and reduced false negatives from 9 to 4, producing 0.9753 accuracy and 0.9745 F1. Even under retraining-enabled operation, the Challenger remained the best-balanced raw path, with 0.8694 accuracy and 0.8776 F1, and with lower false positives than the Champion (104 vs 149) while maintaining high defect capture (0.9827 recall), as reported in Table 45. Across both runs, the event-anchored route therefore remained analytically credible and operationally advantageous under true labels.

The third point concerns the Champion's role and stability. Across the two runs, the Champion's behaviour changed only slightly, remaining close to 0.81 accuracy with false positives around 150 and false negatives around 10. This stability is valuable because it confirms the Champion's function as a fixed reference path rather than as an efficiency-optimised deployment route. The Champion remained highly defect-aware and provided a stable benchmark against which the event-anchored route and the policy-released decision could be interpreted.

The fourth point is the effect and consistency of the Final policy. In both runs, the Final output moved the released decision towards stronger defect protection and higher rejection. In the baseline run, it reduced false negatives from 4 to 1 relative to the Challenger but increased false positives from 17 to 155. In the retrain-stage run, it reduced false negatives from 7 to 0 but increased false positives from 104 to 171. This shows that the deployed decision did not simply follow the strongest raw classifier. The policy remained active and imposed a conservative release posture under both fixed and adaptive operating conditions.

This means that the Final policy should not be judged as if its purpose were to maximise F1-score or minimise processing-weighted false-positive burden. Its purpose is to enforce the selected deployment risk posture. In this study, that posture is deliberately conservative: it gives priority to reducing defect escape, even when this increases unnecessary rejection. The value of the Final layer is therefore that it makes the operating trade-off explicit, stable, and auditable, rather than simply copying the strongest raw model output.

The fifth point is that the consequences of that conservatism are not merely qualitative; they are measurable through operational burden. The processing-weighted false-positive burden results demonstrate that the Challenger maintained the lowest burden in both runs, recording 521.9095 in the baseline run and 4322.0502 in the retrain-stage run. By comparison, the Champion and Final outputs recorded higher burdens of 4666.4852 and 4758.5869, respectively, in the baseline run, and 6192.1681 and 7106.4479, respectively, in the retrain-stage run. These findings reinforce the operational advantage of the Challenger in reducing unnecessary rejection while demonstrating the deliberate risk-management trade-off imposed by the conservative Final policy. The processing-weighted burden comparisons are presented in Table 56 and Figure 65, while the separate normalised relative decision-loss sensitivity analyses are reported in Tables 54 and 55 and Figures 63 and 64.

The sixth point concerns real-time feasibility. Both runs remained comfortably within a cycle-level timing envelope. The baseline run produced 30.7006 ms mean processing time with 34.5781 ms p95, while the retrain-stage run produced 41.5582 ms mean processing time with 60.1806 ms p95. Mean model latencies remained in the low millisecond range for both predictive paths in both runs. This confirms that the predictive strength, policy effects, and cost trade-offs reported in the chapter were achieved under live processing constraints rather than through delayed offline evaluation. The timing evidence supporting this conclusion is reported in Tables 52 and 53 and visualised in Figures 61 and 62.

Finally, the retrain-stage evidence adds a governance result that goes beyond accuracy. The run recorded one successful Challenger retraining event at cycle 745, supported by 40 human corrections, and the run remained complete and truth linked. The drift timeline in Figure 60 shows that the update occurred within an operating sequence exhibiting measurable distributional change, although the strongest Mean PSI and Mean KS excursions occurred earlier in the run. The drift evidence therefore provides operational context rather than a direct statistical trigger for the update. The value of this evidence is not that the second run must outperform the baseline in every whole-run metric. The value is that the governed system admitted a controlled in-run update while preserving cycle integrity, role separation, and real-time feasibility. The platform demonstrated that it can change state without losing control. This governed interpretation is supported by the drift evidence in Table 49 and the recorded retraining evidence summarised in Table 50.

The two 850-cycle runs establish how the architecture behaved under the operating conditions examined, but they do not demonstrate long-term stability across extended production periods, different machines, materials, or manufacturing processes. Similarly, the single governed Challenger retraining event demonstrates that an update could be admitted and executed within the controlled pathway, but it does not establish the effectiveness or stability of repeated adaptation. The 40 reviewed corrections show that human evidence was incorporated into the retraining process, although operator workload, consistency, decision quality, and usability were not evaluated independently. The Champion and Challenger assignments should also be understood as governance roles selected for this implementation rather than as a general requirement that full-curve models must act as Champions or event-anchored models as Challengers. Finally, the observed drift indicators provide evidence of

distributional change, but they do not by themselves establish concept drift, identify the physical cause of that change, or demonstrate that the change caused a particular variation in predictive performance. These boundaries limit the strength of the conclusions that can be drawn from the live results while preserving the evidence that the governed architecture itself operated coherently under the conditions examined.

Overall, the live evidence supports a coherent manufacturing interpretation of the three-path design. The Champion provides a stable full-curve reference that remains defect-aware but operationally costly in false alarms. The Challenger provides a compact and physically anchored operational route that is both analytically strong and far more efficient in unnecessary rejection burden. The Final output enforces a conservative release posture that reduces or eliminates missed defects but carries a substantial operational cost. The two-run evidence shows that this structure remains visible not only under fixed live operation, but also when a governed Challenger update is admitted. This is the central results-level conclusion of the chapter.

8.11 Summary

This chapter evaluated the measured behaviour of the governed live system across two full operational runs: a baseline run before retraining and a retrain-stage run in which drift awareness and retraining governance became active. The purpose was not simply to report model scores, but to determine whether the complete decision structure behaved credibly under real cycle-level manufacturing conditions.

The measured evidence showed a consistent and important result. In both runs, the Challenger remained the strongest predictive path. In the baseline run, it achieved 0.9753 accuracy and 0.9745 F1, compared with 0.8106 accuracy and 0.8311 F1 for the Champion. In the retrain-stage run, it again remained stronger, with 0.8694 accuracy and 0.8776 F1, compared with 0.8129 accuracy and 0.8325 F1 for the Champion. This confirms that the lighter event-anchored path was not only viable in live operation, but repeatedly superior to the fixed full-curve reference under the measured run conditions.

The chapter also showed that the final deployed decision must be interpreted separately from the raw model outputs. The Final path remained much closer to

the conservative operating posture than to the strongest raw model path because the locked policy resolved disagreement cautiously. In the baseline run, the Final output reduced false negatives from 4 to 1 relative to the Challenger but increased false positives from 17 to 155. In the retrain-stage run, it reduced false negatives from 7 to 0 but increased false positives from 104 to 171. This explains why the deployed path preserved a more cautious operational posture even when the Challenger was clearly stronger in raw predictive terms.

A further major result was the behaviour of the governance layer itself. The baseline run remained a non-retraining reference state, with 0 retraining events despite repeated monitored runtime disturbance. By contrast, the retrain-stage run entered formal adaptive review and recorded one successful Challenger retraining event at cycle 745, supported by 40 human corrections. This outcome is important because it shows that the governed framework could admit a controlled update during live operation without breaking cycle continuity, truth linkage, or the visible structure of the decision architecture. The retraining event was therefore not an external model replacement, but part of the live evidence chain itself.

The timing evidence further supported the practical credibility of the system. Mean end-to-end processing time was 30.7006 ms in the baseline run and 41.5582 ms in the retrain-stage run, while p95 timing was 34.5781 ms and 60.1806 ms, respectively. These values show that the full governed path remained comfortably within a practical cycle-level timing range in both runs, even though the retrain-stage run was slower than the baseline run. At the same time, the processing-weighted false-positive burden analysis showed that the Challenger recorded the lowest burden in both conditions, with values of 521.9095 in the baseline run and 4322.0502 in the retrain-stage run, while the Champion and Final outputs recorded substantially higher burdens. This made the operational consequences of the conservative deployment rule clearly measurable.

Taken together, the Chapter 8 findings provide the runtime evidence required to connect the research gaps, research questions, objectives, and contribution of the thesis. The cycle-resolved and truth-linked live records extend the cycle-level foundation established for Research Question 1 and Objective 1 into the operational environment by preserving the relationship between process evidence, model outputs, released decisions, and later quality confirmation. The live comparison between the event-anchored Challenger and full-curve Champion provides further operational evidence for Research Question 2 and

Objective 2, showing that the compact event-anchored representation remained a credible and, under the evaluated conditions, stronger predictive route during runtime use. The implemented separation of Champion, Challenger, Final decision policy, drift monitoring, human-reviewed evidence, retraining control, and persistent records provides the operational validation of Research Question 3 and Objective 3 by demonstrating how these functions can coexist without collapsing prediction, decision authority, and model adaptation into the same process. Most directly, the two live runs address Research Question 4 and Objective 4 by evaluating predictive behaviour, decision-policy effects, timing, distributional change, and governed model adaptation under runtime conditions. In relation to the research gaps identified in the literature, the resulting knowledge contribution is therefore not a single predictive model or isolated improvement in classification performance, but an experimentally evaluated approach for maintaining cycle continuity, complementary predictive representations, explicit decision governance, drift awareness, and controlled adaptation within one traceable real-time manufacturing architecture. The injection-moulding results establish this behaviour within the evaluated case-study conditions and provide the empirical basis for the broader architectural contribution developed in the concluding chapter.

Chapter 9
Conclusions and Future Work

9.1 Concluding the research aim and scope

This research was motivated by a gap between demonstrating predictive performance from manufacturing data and using that predictive capability reliably during cyclic production. In a runtime environment, an accurate model is not sufficient on its own. Process evidence must remain associated with the correct production cycle, predictive outputs must be converted into decisions through defined rules, changing operating behaviour must remain observable, and any modification of the predictive system must occur under controlled and traceable conditions. The research therefore addressed quality monitoring as both a prediction problem and an operational governance problem.

The aim was to propose, implement, and evaluate a governed real-time architecture for cycle-level quality monitoring and decision support in cyclic manufacturing. Injection moulding provided the experimental environment through which this aim was investigated. Synchronised process measurements were organised around individual production cycles and linked to corresponding quality outcomes. This cycle-level foundation supported statistical investigation, the development and comparison of event-anchored and full-curve predictive representations, and their subsequent integration within the governed runtime architecture. The work therefore progressed from physical process evidence to prediction and, finally, to the controlled use and management of predictive information during runtime operation.

The evidence generated through this progression supports the achievement of the research aim. The implemented architecture showed that cycle-level prediction can operate within a structure in which predictive output, released decision, monitoring evidence, and model-update authority remain connected without becoming the same function. Predictive performance does not automatically determine the released quality decision, and evidence of changing data behaviour does not automatically authorise model adaptation. Instead, these functions are governed through explicit operational roles and retained within a traceable cycle-level evidence structure. This distinction is central to the research because it extends quality prediction from an offline modelling task into a controlled manufacturing decision-support process.

These conclusions remain bounded by the scope of the investigation. Injection moulding provides the empirical evidence through which the architecture was developed and evaluated, and the work concerns quality monitoring and decision

support rather than autonomous adjustment of machine actuators. The results establish the feasibility and behaviour of the proposed architecture under the conditions examined in this study. They do not establish equivalent predictive performance or adaptation behaviour across other manufacturing processes. Wider relevance is therefore considered at the architectural level, where identifiable production cycles, measurable process evidence, and corresponding quality outcomes provide the conditions under which the approach may be transferred and reconfigured. The following sections examine the findings against the research objectives, define the contribution to knowledge, and then consider this wider applicability alongside the limitations and directions for future research.

9.2 Key findings and addressing the research questions and objectives

The four research objectives form a connected progression from experimental evidence to runtime evaluation. The findings are therefore considered here as parts of the same research pathway rather than as isolated outcomes. This allows the results to be interpreted in terms of what was learned about cycle-level quality monitoring, representation, prediction, and governed operation, while also providing a direct assessment of whether the objectives were achieved.

The experimental work first established the cycle as a reliable analytical unit for connecting process behaviour with part quality. Across the three injection moulding trials, 903 complete labelled cycles were retained with synchronised machine-side and in-mould measurements linked to the corresponding manufactured parts. This structure allowed process behaviour to be examined at the same level at which quality was ultimately assessed. The statistical investigation confirmed that the measured process variables contained meaningful quality-related information, while also revealing coupled behaviour, non-linearity, and operating variation that limited the explanatory value of simple linear relationships alone.

The representation and modelling results developed this evidence further. Two different descriptions of each production cycle were investigated: an event-anchored representation centred on the maximum cavity-pressure event and a full-curve representation retaining the wider temporal behaviour of the monitored signals. Both supported strong cycle-level quality prediction. For the

direct route comparison, XGBoost achieved ROC-AUC and PR-AUC values of 0.991 using the event-anchored representation and 0.994 using the full-curve representation. The difference of 0.003 in both measures indicates that the additional temporal information retained by the full-curve route produced only a small improvement in discrimination for the dataset examined. The event-anchored route also achieved an F1 score of 0.963 with Gradient Boosting at its strongest threshold-based operating point. These findings demonstrate that a physically meaningful in-cycle event can preserve a large proportion of the quality-related information required for classification while providing a more compact representation. They also establish that representation choice should be based on the relationship between predictive value, information requirement, and intended operational role rather than on input richness alone.

Once the predictive capability of both representations had been established, the research moved from offline model comparison to their integration within the governed real-time architecture. The full-curve and event-anchored routes were assigned complementary Champion and Challenger roles, but neither predictive route was given unrestricted authority over the released quality decision. A separate Final policy resolved the operational output, while monitoring evidence and model-management functions were maintained independently from that decision. This architecture created explicit boundaries between prediction, decision authority, and adaptation authority. A model could provide strong predictive evidence without automatically determining the released decision, and evidence of changing operating conditions could be recorded without automatically initiating model replacement.

The runtime evaluations provided the evidence required to assess how this structure behaved in operation. Two evaluations were conducted using the same governed architecture, each containing 850 accepted cycles with corresponding quality evidence. Across both runs, the event-anchored Challenger produced the stronger raw predictive balance than the fixed full-curve Champion. The Final output behaved differently because the locked decision policy was deliberately conservative when the two predictive routes disagreed. This reduced the number of missed defects relative to the Challenger, including eliminating false negatives in the retraining-enabled run, but increased false-positive rejection. The result demonstrates that the performance of a predictive model and the behaviour of the decision released by the system are not equivalent. The selected decision

policy introduces its own risk preference, and its operational consequences must therefore be evaluated separately from model accuracy.

The runtime evidence also clarified the relationship between monitoring and adaptation. Distributional indicators identified changing behaviour during the evaluated sequence, including increased change in the later stage of the retraining-enabled run. These observations provide evidence of distributional change, but they are not treated as proof of concept drift. More importantly, the monitored change did not itself authorise retraining. One governed Challenger update was executed at cycle 745 after the required admission conditions had been satisfied, while the Champion remained fixed and the Final decision policy remained unchanged. The significance of this event is therefore not that retraining necessarily improved subsequent predictive performance, but that a model transition could be executed within the governed pathway without removing the separation between monitoring evidence, update authority, model roles, and decision policy.

Operational feasibility was also supported by the timing results. Mean end-to-end processing time was approximately 30.7 ms in the baseline run and 41.6 ms in the retraining-enabled run, with p95 values of approximately 34.6 ms and 60.2 ms, respectively. The complete processing pathway therefore remained within the available cycle-level operating window under both evaluated conditions. Cost-sensitive analysis provided a complementary operational perspective by showing that the conservative Final policy carried a higher false-positive burden than the Challenger. This reinforces a broader finding of the research: practical evaluation of predictive manufacturing systems must consider not only classification performance, but also decision policy, error consequences, changing operating conditions, timing, traceability, and the governance of model change.

The principal findings are summarised in Table 57, together with the evidence on which they are based and their main interpretation within the governed real-time architecture.

Table 57. Summary of the principal findings of the research and their supporting evidence.

Key finding	Supporting evidence	Main interpretation
Event-anchored representation provided an effective operational prediction route.	The Challenger achieved the strongest raw predictive performance in the baseline run and remained the better-balanced raw route under retraining-enabled operation.	A compact representation centred on the MP1 event can retain useful quality-related information without requiring the complete full-curve representation.
Decision authority was separated from raw model performance.	Champion and Challenger outputs were resolved through a separate locked Final policy. The conservative rule reduced missed defects but increased false positives.	The released manufacturing decision can be governed explicitly rather than being determined automatically by the strongest predictive model.
Changing process behaviour was observable during runtime evaluation.	Distributional monitoring showed non-uniform behaviour across the run, including increased late-stage drift evidence before the recorded retraining event.	Drift awareness provides operational evidence for model management, but distributional change alone should not be treated as proof of concept drift or as an automatic retraining trigger.
Model adaptation was introduced through a governed pathway.	One governed Challenger retraining event was recorded at cycle 745 after the required evidence and admission conditions had been satisfied.	Adaptation can be separated from normal inference and treated as a controlled, traceable model-management action.
The complete architecture was feasible at cycle level.	Cycle processing, representation, inference, policy resolution, persistence, and monitoring operated within the available cycle-level operating window.	The complete decision-support workflow was executable under the evaluated cycle-level runtime conditions rather than being assessed only through offline analytical performance.
Conservative deployment created a measurable operational trade-off.	The Challenger produced the lowest false-alarm burden, while the Final policy reduced missed defects by accepting additional false positives.	Manufacturing deployment should be evaluated in terms of operational consequences as well as predictive performance.

The findings also provide direct responses to the four research questions established from the literature review. RQ1 is answered by showing that synchronised multi-sensor process evidence can be organised around identifiable production cycles and linked to corresponding part-quality outcomes, providing a consistent basis for analysis and quality monitoring. RQ2 is answered by the comparison of the event-anchored and full-curve representations, which showed that the event-anchored route retained predictive discrimination close to the full-curve route while requiring substantially less raw cycle information. RQ3 is addressed through the implemented governed architecture, in which complementary predictive roles, explicit decision authority, drift-aware

monitoring, controlled model adaptation, and traceable evidence are integrated while remaining functionally distinct. RQ4 is answered through the runtime evaluation, which established how the architecture behaved in terms of predictive performance, decision-policy effects, processing time, distributional change, and governed model updating under the conditions examined.

These findings also provide the evidence required to assess the four research objectives defined in Chapter 1. The first objective was achieved by establishing a cycle-level experimental and analytical basis in which synchronised process measurements were linked to corresponding part-quality outcomes. Building on this foundation, the second objective was addressed through the development and comparison of event-anchored and full-curve representations and the evaluation of their predictive capability. The third objective extended this predictive work into operation through the design and implementation of the governed real-time architecture, in which Champion–Challenger roles, Final decision authority, drift-aware monitoring, controlled model updating, and traceable evidence were integrated within the same runtime structure. The fourth objective was addressed through the runtime evaluation of this architecture, including its predictive behaviour, decision-policy effects, timing, robustness, distributional change, and governed adaptation. The four objectives were therefore achieved within the experimental and runtime scope of the injection moulding study, providing the evidence from which the contribution to knowledge is defined in the following section.

9.3 Contributions to knowledge and manufacturing practice

The principal contribution of this research is the development and evaluation of a governed real-time architecture for cycle-level quality monitoring and decision support in cyclic manufacturing. Its originality does not arise from any individual sensor, predictive model, drift indicator, or retraining mechanism. Rather, it lies in integrating these elements within a common operational structure while maintaining clear boundaries between prediction, decision release, monitoring, and model adaptation. The production cycle provides the common reference through which process information, analytical outputs, decisions, and subsequent model-management actions remain connected.

This overarching contribution is supported by four specific contributions to knowledge.

1. **Cycle-level governance and continuity of manufacturing evidence**

The study establishes the manufacturing cycle as the organising unit that connects process measurements, analytical representations, predictive outputs, released decisions, quality outcomes, and model-management records. The contribution is not the identification or segmentation of cycles in itself. It is the preservation of cycle identity across the analytical and runtime pathway so that information generated at different stages remains associated with the same production event.

This continuity creates a traceable relationship between what was measured, how the cycle was represented, what each predictive route produced, which decision was released, and what subsequent monitoring or model-management activity occurred. The resulting evidence chain provides the foundation for the governance mechanisms implemented throughout the system.

2. **Complementary cycle representations with distinct operational roles**

The research demonstrates how different representations of the same manufacturing cycle can be used for complementary purposes rather than being treated only as competing alternatives during offline model selection. The full-curve representation retains the wider temporal behaviour of the monitored process and is used as the fixed Champion reference. The event-anchored representation provides a substantially more compact Challenger pathway centred on a physically meaningful point within the cycle.

Their contribution therefore extends beyond a comparison of predictive performance. The two representations are assigned different responsibilities within the runtime system, allowing a stable reference pathway to coexist with a separately manageable and updateable Challenger. This role-based arrangement preserves direct comparison between the two analytical views while supporting controlled adaptation without modifying the fixed reference route.

3. Separation of predictive, decision, and adaptation authority

A central contribution of the proposed architecture is the explicit separation of three functions that may otherwise become combined in predictive manufacturing systems: generating a prediction, releasing a quality decision, and authorising a model update.

The Champion and Challenger produce predictive evidence, but neither model alone determines the released output. The Final decision is resolved through a separate policy layer. Model updating is controlled independently through defined admission conditions. As a result, observed distributional change does not automatically trigger retraining, and an update to the Challenger does not alter either the fixed Champion or the Final decision policy.

This separation establishes an important governance principle. Predictive performance does not automatically confer decision authority, and evidence of changing process behaviour does not automatically provide authority to modify a deployed model. Maintaining these distinctions makes operational decisions and model-management actions easier to trace, review, and justify.

4. System-level runtime evaluation beyond classification performance

The research also contributes an evaluation approach that examines predictive manufacturing intelligence as a complete operational system rather than judging it only through classifier performance. The runtime assessment considered the behaviour of both predictive routes, their disagreement, the effect of the Final policy, distributional change, governed adaptation, processing time, traceability, and the consequences of false-positive and false-negative decisions.

This broader perspective reveals effects that are not visible through offline model comparison alone. A predictive route may achieve strong classification performance while the decision policy produces a different operational balance. Likewise, monitored change may provide important evidence without necessarily justifying model replacement. Evaluating these elements together provides a more complete basis for assessing how predictive quality support behaves under runtime manufacturing conditions.

For manufacturing practice, the proposed approach provides a structured way to introduce predictive quality support while retaining explicit control over how analytical outputs influence operational decisions and model updates. A fixed

reference model can be maintained alongside an adaptable Challenger, the released quality decision can remain governed by a defined policy, and the associated process, prediction, monitoring, and model-management records can be preserved for later review. The practical value therefore lies not only in producing cycle-level predictions, but also in controlling how those predictions are used, interpreted, and managed during operation.

Injection moulding provides the experimental environment in which these contributions were developed and evaluated. The wider significance of the work lies in the architectural and governance principles established through that implementation. Their transferability to other cyclic manufacturing processes, together with the limitations of the present evidence and the priorities for further research, is considered in the following section.

9.4 Generalisability, limitations and future research

The proposed architecture was developed and evaluated using injection moulding, but its underlying logic was formulated for cyclic manufacturing more broadly. Transfer to another process depends on three basic conditions: production cycles must be identifiable, process measurements must be associated with individual cycles, and a relevant quality outcome must be linked to the corresponding production event. Where these conditions are met, the same general logic can be reconfigured to support cycle-level representation, prediction, decision governance, monitoring, evidence preservation, and controlled model management.

The most transferable element is therefore the architectural and governance logic rather than the specific injection moulding implementation. Sensor types and locations, cycle-boundary signals, event anchors, predictive variables, model parameters, decision thresholds, drift references, retraining conditions, and operational cost assumptions would need to be defined for the process concerned. In particular, the maximum cavity-pressure event used in this study should not be regarded as a universal event anchor. It demonstrates how a repeatable and physically meaningful event can support compact cycle representation, but another manufacturing process would require an equivalent event appropriate to its own behaviour and quality mechanism. The evidence therefore supports architectural generalisability, but not performance generalisability. Predictive accuracy, decision-policy behaviour, drift

characteristics, and adaptation outcomes would require independent evaluation in other cyclic manufacturing environments.

The present evidence also has clear limitations. The empirical implementation was confined to injection moulding and therefore does not represent the full range of cyclic manufacturing processes. In addition, the runtime assessment was based on two evaluations of 850 accepted cycles each. These records served a different purpose from the 903-cycle dataset used for offline model development and representation comparison. They were used to examine runtime behaviour, timing, decision-policy effects, distributional change, and governed adaptation under the evaluated operating conditions. The two runs also covered bounded operating periods and therefore do not establish how the architecture would behave under longer-term process and sensing variation.

A further limitation concerns the Final decision policy. The implemented rule was deliberately conservative and was not designed to maximise predictive balance or minimise false-positive burden. Its purpose was to maintain a defect-protective decision posture when the Champion and Challenger disagreed. The resulting operational burden therefore reflects the selected governance rule rather than an optimised industrial decision strategy. The normalised relative decision-loss sensitivity analysis and processing-weighted false-positive burden comparison provide complementary measures of error-related consequences within the evaluated runs. These measures are intended to support transparent comparison of decision policies rather than represent a complete factory-level economic model incorporating inspection, scrap, rework, throughput loss, labour, downtime, or downstream disruption.

The evidence for adaptation is also limited. One governed Challenger update was recorded at cycle 745 after the defined admission conditions had been satisfied. This establishes that a model update could be admitted and executed within the controlled pathway while the Champion remained fixed and the Final policy remained unchanged. It does not demonstrate the long-term effectiveness of repeated adaptation, nor does it establish sustained predictive improvement after the update. The available post-update evaluation period was limited, and the study did not examine multiple accepted and rejected update opportunities under different forms of process change. The 40 human-reviewed corrections demonstrate that human-provided evidence was incorporated into the governed retraining pathway, while the independent evaluation of operator workload, review consistency and usability remains an opportunity for future research.

These limitations define several priorities for further research. The first is broader evaluation across additional cyclic manufacturing processes with different cycle structures, sensing arrangements, process timescales, and quality mechanisms. Such studies would provide evidence on which architectural elements transfer directly and which require process-specific redesign. Longer runtime evaluations are also needed to assess predictive stability, disagreement behaviour, monitoring performance, and governance over more extended production periods.

Sensor drift should receive particular attention in this longer-term evaluation. A change in the measured data distribution may result from a genuine change in the manufacturing process, degradation or drift in the sensing system, or a combination of both. Future implementations should therefore distinguish sensor-related change from process-related change before monitored distributional movement is used as evidence for model-management decisions. This distinction would strengthen the interpretation of drift indicators and reduce the risk of responding to measurement change as though it necessarily represented a change in the underlying manufacturing process.

The adaptation pathway should also be examined across repeated model-management events. Future studies should include accepted and rejected update attempts, successive adaptations, different forms of distributional change, and longer post-update verification periods. Such evaluation would allow the effectiveness and stability of governed adaptation to be assessed over time rather than through a single recorded transition.

Further development of the Final decision policy is another important direction. The fixed conservative rule used in this study provided a clear and traceable operating policy, but the runtime results also showed the burden associated with that conservatism. Future work should investigate risk- and cost-sensitive alternatives that retain explicit decision governance while allowing the balance between missed defects and unnecessary rejection to be adjusted according to manufacturing requirements. This should be accompanied by a more complete industrial cost model. The disagreement between Champion and Challenger also warrants further investigation as a possible source of evidence for process review, model assessment, or retraining admission rather than being treated only as a reported runtime outcome.

A further direction for future research is the extension of the implemented governed real-time architecture towards a fully governed Digital Twin with two complementary integration pathways. The existing architecture provides a technical foundation through its cycle-resolved acquisition, predictive modelling, operational monitoring, decision governance, and traceable data records. The first pathway concerns bidirectional integration with the manufacturing machine. Future work could incorporate quantitative part-quality measurements, including length, width, and thickness, to establish validated relationships between product dimensions, process behaviour, and controllable machine parameters. These relationships could support recommendations for process adjustment and, following appropriate validation and safety controls, governed feedback to machine settings. The second pathway concerns integration with manufacturing management systems, where cycle-level evidence could provide continuous information about production status, process behaviour, the numbers of acceptable and defective parts, quality trends, and emerging operational risks. Further predictive development could also support short-term forecasts of process and quality behaviour over subsequent production cycles, allowing management to anticipate potential deterioration and plan corrective interventions. Together, these extensions would provide a pathway from the present governed quality-monitoring and decision-support architecture towards a more comprehensive Digital Twin connecting machine-level operation with manufacturing-level decision-making.

Further research should therefore focus on testing the robustness and transferability of the principles established in this study rather than simply increasing predictive complexity. One specific extension is the timing of event-anchored inference. In the present implementation, the \(MP1\)-anchored Challenger is evaluated after cycle completion because the governed Final decision also incorporates the full-curve Champion, which requires the completed cycle record. The event-anchored representation itself does not inherently require this delay. A future implementation could introduce online \(MP1\)-peak confirmation and evaluate the Challenger shortly after the peak has been confirmed from the live signal. This would permit an earlier provisional quality indication when the Challenger is authorised to operate independently, while the full-curve Champion could remain as a later reference or verification route. Such an approach would require a robust peak-confirmation rule to distinguish the final \(MP1\) maximum from earlier local peaks and would

require explicit governance of the relationship between any provisional prediction and the later Final decision. Cross-process evaluation, longer-duration operation, explicit sensor-drift assessment, repeated governed adaptation, improved decision-policy design, earlier event-anchored inference, and stronger industrial cost modelling would provide the evidence needed to determine how far the evaluated injection moulding implementation can be extended towards a more broadly applicable architecture for cyclic manufacturing.

9.5 Final concluding statement

This research develops and evaluates, through an injection-moulding case study, a governed approach to cycle-level quality monitoring and decision support with potential applicability to other cyclic manufacturing processes. The work shows how process evidence, predictive interpretation, decision logic, monitoring, model management, and traceable records can be coordinated within one operational architecture while preserving clear boundaries between prediction, decision release, and model adaptation. The principal outcome is therefore not a single predictive model, but a structured way of managing how predictive information is used during manufacturing operation.

Within the injection moulding study, the architecture was implemented and evaluated under runtime conditions, providing evidence that this governed structure can operate at cycle level while supporting complementary predictive routes, explicit decision control, monitoring of changing behaviour, and controlled model updating. These conclusions remain bounded by the process and operating conditions examined in the thesis. The wider significance of the research lies in providing an architectural foundation that can be adapted and further evaluated in other cyclic manufacturing environments where individual production cycles can be identified, monitored, and linked to corresponding quality outcomes.

References

Abd Aziz, A.A.B. *et al.* (2015) 'Investigation of plastic injection moulding process optimisation in complex shape product', *Applied Mechanics and Materials*, 695, pp. 260–264.

Ageyeva, T., Horváth, S. and Kovács, J.G. (2019) 'In-mold sensors for injection molding: On the way to Industry 4.0', *Sensors*, 19(16), article 3551. Available at: https://doi.org/10.3390/s19163551.

Akın, Ö. *et al.* (2021) 'Enabling Big Data Analytics at Manufacturing Fields of Farplas Automotive', in Kahraman, C. et al. (eds) *Intelligent and Fuzzy Techniques: Smart and Innovative Solutions*. Cham: Springer International Publishing, pp. 817–824. Available at: https://doi.org/10.1007/978-3-030-51156-2_94.

Alkhatib, F. *et al.* (2025) 'Implementing and evaluating the quality 4.0 PMQ framework for process monitoring in automotive manufacturing', *Scientific Reports*, 15(1), article 24742. Available at: https://doi.org/10.1038/s41598-025-10226-4.

Aminabadi, S.S. *et al.* (2022) 'Industry 4.0 in-line AI quality control of plastic injection molded parts', *Polymers*, 14(17), article 3551. Available at: https://doi.org/10.3390/polym14173551.

Arslan, E., Konuşkan, Y. and Lazoglu, I. (2025) 'AI-driven cognition for advanced injection molding and industrial implementation', *The International Journal of Advanced Manufacturing Technology*, 138(5-6), pp. 2043–2064. Available at: https://doi.org/10.1007/s00170-025-15611-x.

Babalola, A.S. (2021) 'Design and construction of a sensor analytic system for the monitoring of the parameters of a plastic injection mould', *Frontiers in Engineering and Built Environment*, 1(1), pp. 32–40. Available at: https://doi.org/10.1108/FEBE-02-2021-0003.

Bauer, J.C. *et al.* (2025) 'A continual active learning approach to adapt neural networks to distribution shifts in quality monitoring applications', *The International Journal of Advanced Manufacturing Technology*, 138(9-10), pp. 4137–4153. Available at: https://doi.org/10.1007/s00170-025-15786-3.

Bauer, J.C. *et al.* (2026) 'Handling data drift in deep learning-based quality monitoring: Evaluating calibration methods using the example of friction stir welding', *Journal of Intelligent Manufacturing*, 37(2), pp. 759–774. Available at: https://doi.org/10.1007/s10845-025-02569-6.

Behnke, D. *et al.* (2018) 'Intelligent Network Services enabling Industrial IoT Systems for Flexible Smart Manufacturing', in *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. Limassol: IEEE, pp. 1–4. Available at: https://doi.org/10.1109/WiMOB.2018.8589088.

Bilen, A. *et al.* (2025) 'Coping with concept drift in a virtual metrology application to predict part quality in micro gear manufacturing', *Measurement: Sensors*, 38, article 101797. Available at: https://doi.org/10.1016/j.measen.2024.101797.

Bogedale, L. *et al.* (2023) 'Online Prediction of Molded Part Quality in the Injection Molding Process Using High-Resolution Time Series', *Polymers*, 15(4), p. 978. Available at: https://doi.org/10.3390/polym15040978.

Cao, P. *et al.* (2025) 'Emerging approaches to process control in injection molding: A comprehensive review', *The International Journal of Advanced Manufacturing Technology*, 140(9), pp. 4483–4502. Available at: https://doi.org/10.1007/s00170-025-16577-6.

Catti, P., Nikolakis, N. and Alexopoulos, K. (2025) 'A human-centric drift controller framework for adaptive and explainable quality control in manufacturing', in *2025 IEEE 30th International Conference on Emerging Technologies and Factory Automation (ETFA)*. Porto, Portugal: IEEE, pp. 1–8. Available at: https://doi.org/10.1109/ETFA65518.2025.11205586.

Cerquitelli, T. *et al.* (2021) 'Enhancing manufacturing intelligence through an unsupervised data-driven methodology for cyclic industrial processes', *Expert Systems with Applications*, 182, article 115269. Available at: https://doi.org/10.1016/j.eswa.2021.115269.

Chaciński, P. and Sutowski, P. (2021) 'Common defects in injection molding of plastic products and their influence on product quality', *Journal of Mechanical and Energy Engineering*, 5(1), pp. 7–14. Available at: https://doi.org/10.30464/jmee.2021.5.1.7.

Chang, C. *et al.* (2025) 'Scientific Molding and Adaptive Process Quality Control with External Sensors for Injection Molding Process', *Technologies*, 13(3), p. 97. Available at: https://doi.org/10.3390/technologies13030097.

Charest, M., Finn, R. and Dubay, R. (2018) 'Integration of artificial intelligence in an injection molding process for on-line process parameter adjustment', in *2018 Annual IEEE International Systems Conference (SysCon)*. Vancouver, Canada: IEEE, pp. 1–6. Available at: https://doi.org/10.1109/SYSCON.2018.8369500.

Chen, Z. and Turng, L. (2005) 'A review of current developments in process and quality control for injection molding', *Advances in Polymer Technology*, 24(3), pp. 165–182. Available at: https://doi.org/10.1002/adv.20046.

Chen, W.-C., Nguyen, M.-H. and Tai, P.-H. (2018) 'An intelligent manufacturing system for injection molding', *Proceedings of Engineering and Technology Innovation*, 8, pp. 9–14.

Chen, J.-Y., Yang, K.-J. and Huang, M.-S. (2018) 'Online quality monitoring of molten resin in injection molding', *International Journal of Heat and Mass Transfer*, 122, pp. 681–693. Available at: https://doi.org/10.1016/j.ijheatmasstransfer.2018.02.019.

Chen, J.C., Guo, G. and Chang, Y.H. (2023) 'Intelligent dimensional prediction systems with real-time monitoring sensors for injection molding via statistical regression and artificial neural networks', *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 17(3), pp. 1265–1276.

Chen, J.Y., Chen, X.P. and Huang, M.S. (2025) 'In-situ measurement of transient polymer melt flow at injection mold gate using pressure sensing and mass conservation principle', Measurement, 256, article 118246. Available at: https://doi.org/10.1016/j.measurement.2025.118246.

Chen, P. et al. (2026) 'Application of ensemble learning for label noise mitigation in intelligent manufacturing monitoring platforms', The International Journal of Advanced Manufacturing Technology, 142(7–8), pp. 3461–3478. Available at: https://doi.org/10.1007/s00170-025-17242-8.

Cheng, H. *et al.* (2017) 'Ontology-based web service integration for flexible manufacturing systems', in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*. Emden, Germany: IEEE, pp. 351–356. Available at: https://doi.org/10.1109/INDIN.2017.8104797.

Cheng, J. et al. (2018) 'Industrial IoT in 5G environment towards smart manufacturing', Journal of Industrial Information Integration, 10, pp. 10–19. Available at: https://doi.org/10.1016/j.jii.2018.04.001.

Cheng, Y. *et al.* (2025) 'Data-driven online prediction and control method for injection molding product quality', *Journal of Manufacturing Processes*, 145, pp. 252–273. Available at: https://doi.org/10.1016/j.jmapro.2025.04.054.

Chew, J.W. *et al.* (2026) 'AI in chemical engineering: From promise to practice', *AIChE Journal*, 72(7), article e70358. Available at: https://doi.org/10.1002/aic.70358.

Cohen, J. and Ni, J. (2021) 'A semi-supervised multiclass anomaly detection approach for partially labelled in-process measurement data', in *Proceedings of the ASME 2021 16th International Manufacturing Science and Engineering Conference*. ASME, article V002T09A004. Available at: https://doi.org/10.1115/MSEC2021-63465.

Demchenko, Y. *et al.* (2013) 'Addressing big data issues in scientific data infrastructure', in *2013 International Conference on Collaboration Technologies and Systems (CTS)*. San Diego, CA, USA: IEEE, pp. 48–55. Available at: https://doi.org/10.1109/CTS.2013.6567203.

Ding, W. *et al.* (2025) 'Novel deep learning based soft sensor feature extraction for part weight prediction in injection molding processes', *Journal of Manufacturing Systems*, 78, pp. 58–68. Available at: https://doi.org/10.1016/j.jmsy.2024.11.011.

Elkan, C. (2001) 'The foundations of cost-sensitive learning', in *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pp. 973–978.

Esmaili, P. *et al.* (2025) 'Cycle-Informed Triaxial Sensor for Smart and Sustainable Manufacturing', *Sensors*, 25(14), article 4431. Available at: https://doi.org/10.3390/s25144431.

EUROMAP (2000) *EUROMAP 63: Data Exchange Interface*. Available at: https://euromap.org/media/recommendations/63/2000/eu63.pdf.

EUROMAP (2020) *EUROMAP 77: OPC UA Interfaces for Plastics and Rubber Machinery – Data Exchange between Injection Moulding Machines and MES*. Available at: https://www.euromap.org/euromap77.

Fathy, Y., Jaber, M. and Brintrup, A. (2021) 'Learning with imbalanced data in smart manufacturing: A comparative analysis', IEEE Access, 9, pp. 2734–2757. Available at: https://doi.org/10.1109/ACCESS.2020.3047838.

Fernandes, C. *et al.* (2018) 'Modeling and Optimization of the Injection-Molding Process: A Review', *Advances in Polymer Technology*, 37(2), pp. 429–449. Available at: https://doi.org/10.1002/adv.21683.

Ferraro, A. *et al.* (2026) 'ExpLusion: Explanation-driven Late Fusion for enhanced production process monitoring', *Computers & Industrial Engineering*, 217, article 112101. Available at: https://doi.org/10.1016/j.cie.2026.112101.

Garcia, A. *et al.* (2024) 'Time Series Manufacturing Data Edge Monitoring and Visualization to Support Industrial Maintenance Teams', *SN Computer Science*, 5(1), p. 131. Available at: https://doi.org/10.1007/s42979-023-02442-4.Gaspar-Cunha, A. *et al.* (2025) 'A Review on Injection Molding: Conformal Cooling Channels, Modelling, Surrogate Models and Multi-Objective Optimization', *Polymers*, 17(7), p. 919. Available at: https://doi.org/10.3390/polym17070919.

Gerling, A. *et al.* (2022) 'Comparison of algorithms for error prediction in manufacturing with AutoML and a cost-based metric', *Journal of Intelligent Manufacturing*, 33(2), pp. 555–573. Available at: https://doi.org/10.1007/s10845-021-01890-0.

Gim, J. and Rhee, B. (2021) 'Novel Analysis Methodology of Cavity Pressure Profiles in Injection-Molding Processes Using Interpretation of Machine Learning Model', *Polymers*, 13(19), article 3297. Available at: https://doi.org/10.3390/polym13193297.

Gim, J. and Turng, L.S. (2023) 'Interpretation of the effect of transient process data on part quality of injection molding based on explainable artificial intelligence', *International Journal of Production Research*, 61(23), pp. 8192–8212. Available at: https://doi.org/10.1080/00207543.2023.2216310.

Gordon, G. *et al.* (2015) 'Quality control using a multivariate injection molding sensor', *The International Journal of Advanced Manufacturing Technology*, 78(9–12), pp. 1381–1391. Available at: https://doi.org/10.1007/s00170-014-6706-6.

Gülçür, M. and Whiteside, B. (2021) 'A study of micromanufacturing process fingerprints in micro-injection moulding for machine learning and Industry 4.0 applications', *The International Journal of Advanced Manufacturing Technology*, 115(5-6), pp. 1943–1954. Available at: https://doi.org/10.1007/s00170-021-07252-7.

Haberchad, O. and Salih-Alj, Y. (2025) 'Cavity sensing for defect prevention in injection molding', Journal of Engineering Research and Sciences, 4(5), pp. 10–19. Available at: https://doi.org/10.55708/js0405002.

He, M. et al. (2023) 'Learning with supervised data for anomaly detection in smart manufacturing', International Journal of Computer Integrated Manufacturing, 36(9), pp. 1331–1344. Available at: https://doi.org/10.1080/0951192X.2023.2177747.

Heymann, H. et al. (2023) 'Assessment framework for deployability of machine learning models in production', Procedia CIRP, 118, pp. 32–37. Available at: https://doi.org/10.1016/j.procir.2023.06.007.

Hinder, F., Vaquet, V. and Hammer, B. (2024) 'One or two things we know about concept drift: A survey on monitoring in evolving environments. Part A: Detecting concept drift', *Frontiers in Artificial Intelligence*, 7, article 1330257. Available at: https://doi.org/10.3389/frai.2024.1330257.

Huang, M.S., Nian, S.C. and Lin, G.T. (2021) 'Influence of V/P switchover point, injection speed, and holding pressure on quality consistency of injection-molded parts', *Journal of Applied Polymer Science*, 138(41), article 51223. Available at: https://doi.org/10.1002/app.51223.

Jourdan, N. et al. (2023) 'Handling concept drift in deep learning applications for process monitoring', Procedia CIRP, 120, pp. 33–38. Available at: https://doi.org/10.1016/j.procir.2023.08.007.

Jourdan, N. (2024) *Addressing Concept Drift in Machine Learning-Based Monitoring of Manufacturing Processes*. PhD thesis. Technische Universität Darmstadt. Available at: https://doi.org/10.26083/tuprints-00028043.

Junker, H. and Domann, C. (2017) 'Towards Industry 4.0 in corporate energy management', WIT Transactions on Ecology and the Environment, 214, pp. 49–56. Available at: https://doi.org/10.2495/ECO170051.

Kafunah, J., Ali, M.I. and Breslin, J.G. (2021) 'Handling imbalanced datasets for robust deep neural network-based fault detection in manufacturing systems', Applied Sciences, 11(21), article 9783. Available at: https://doi.org/10.3390/app11219783.

Kazmer, D.O. et al. (2011) 'Feasibility analysis of an in-mold multivariate sensor', International Polymer Processing, 26(1), pp. 63–72. Available at: https://doi.org/10.3139/217.2397.

Ke, K.-C. and Huang, M.-S. (2020) 'Quality Prediction for Injection Molding by Using a Multilayer Perceptron Neural Network', *Polymers*, 12(8), article 1812. Available at: https://doi.org/10.3390/polym12081812.

Kim, Y. and Lee, D.Y. (2025) 'The Development of REST API Gateway Application Using Unified CNC Interface Platform', *International Journal of Precision Engineering and Manufacturing-Smart Technology*, 3(1), pp. 97–105. Available at: https://doi.org/10.57062/ijpem-st.2024.00171.

Köken, R., Firuzan, A.R. and Yavuz, İ. (2026) 'Application of Artificial Intelligence-Integrated Six Sigma Methodology for Multi-Objective Optimization in Injection Molding Processes', *Applied Sciences*, 16(2), article 1025. Available at: https://doi.org/10.3390/app16021025.

Kotsiopoulos, T. *et al.* (2024) 'Revolutionizing defect recognition in hard metal industry through AI explainability, human-in-the-loop approaches and cognitive mechanisms', *Expert Systems with Applications*, 255, article 124839. Available at: https://doi.org/10.1016/j.eswa.2024.124839.

Kozjek, D. et al. (2017) 'Interpretative identification of the faulty conditions in a cyclic manufacturing process', Journal of Manufacturing Systems, 43, pp. 214–224. Available at: https://doi.org/10.1016/j.jmsy.2017.03.001.

Kuhn, M. and Franke, J. (2021) 'Data continuity and traceability in complex manufacturing systems: a graph-based modeling approach', International Journal of Computer Integrated Manufacturing, 34(5), pp. 549–566. Available at: https://doi.org/10.1080/0951192X.2021.1901320.

Kumar, R. *et al.* (2026) 'Cyber physical production system for smart manufacturing analytics and management: a systematic literature review, framework and roadmap', *International Journal of Computer Integrated Manufacturing*, 39(10), pp. 1579–1609. Available at: https://doi.org/10.1080/0951192X.2025.2563261.

Kurt, M. et al. (2009) 'Experimental investigation of plastic injection molding: Assessment of the effects of cavity pressure and mold temperature on the quality of the final products', Materials & Design, 30(8), pp. 3217–3224. Available at: https://doi.org/10.1016/j.matdes.2009.01.004.

Kusić, D. and Slapšak, M. (2020) 'Development of IoT device for temperature and cavity pressure measurements', in Hopmann, C. and Dahlmann, R. (eds) *Advances in Polymer Processing 2020: Proceedings of the International Symposium on Plastics Technology*. Berlin: Springer Vieweg, pp. 108–115. Available at: https://doi.org/10.1007/978-3-662-60809-8_9.

Kvaktun, D., Liu, D. and Schiffers, R. (2023) 'Detection of concept drift for quality prediction and process control in injection molding', AIP Conference Proceedings, 2884(1), article 110008. Available at: https://doi.org/10.1063/5.0168491.

Landset, S. et al. (2015) 'A survey of open source tools for machine learning with big data in the Hadoop ecosystem', Journal of Big Data, 2(1), article 24. Available at: https://doi.org/10.1186/s40537-015-0032-1.

Latsou, C. *et al.* (2024) 'A unified framework for digital twin development in manufacturing', *Advanced Engineering Informatics*, 62, article 102567. Available at: https://doi.org/10.1016/j.aei.2024.102567.

Le Goff, R. *et al.* (2025) 'Virtual sensor for injection molding monitoring', Journal of Manufacturing and Materials Processing, 9(9), article 311. Available at: https://doi.org/10.3390/jmmp9090311.

Lee, H., Ryu, K. and Cho, Y. (2017) 'A framework of a smart injection molding system based on real-time data', *Procedia Manufacturing*, 11, pp. 1004–1011. Available at: https://doi.org/10.1016/j.promfg.2017.07.206.

Lee, J., Lee, Y.C. and Kim, J.T. (2020) 'Fault detection based on one-class deep learning for manufacturing applications limited to an imbalanced database', Journal of Manufacturing Systems, 57, pp. 357–366. Available at: https://doi.org/10.1016/j.jmsy.2020.10.013.

Liew, K.F. et al. (2022) 'Injection barrel/nozzle/mold-cavity scientific real-time sensing and molding quality monitoring for different polymer-material processes', Sensors, 22(13), article 4792. Available at: https://doi.org/10.3390/s22134792.

Lin, J. and Chen, K. (2024) 'A novel decision support system based on computational intelligence and machine learning: Towards zero-defect manufacturing in injection molding', *Journal of Industrial Information Integration*, 40, article 100621. Available at: https://doi.org/10.1016/j.jii.2024.100621.

Lin, S.C. and Chen, C.Y. (2026) 'An integrated framework featuring policy-governed agentic AI for closed-loop manufacturing control with multi-source sensor–MES–ERP', The International Journal of Advanced Manufacturing Technology, 143, pp. 6405–6428. Available at: https://doi.org/10.1007/s00170-026-17806-2.

Liu, X. *et al.* (2025) 'A Novel Industrial Defect Recognition Method Under Label Noise Based on Parallel Hybrid Penalty Network', *IEEE Transactions on Automation Science and Engineering*, 22, pp. 9584–9597. Available at: https://doi.org/10.1109/TASE.2024.3508774.

Lobo, A., Sampaio, P. and Novais, P. (2024) 'Enhancing Quality 4.0 and reducing costs in lot-release process with machine learning-based complaint prediction', *The TQM Journal*, 36(9), pp. 175–192.

Loftis, J., Farahani, S. and Pilla, S. (2020) 'Online quality monitoring of plastic parts using real-time data from an injection molding machine', in *Proceedings of the ASME 2020 15th International Manufacturing Science and Engineering Conference*. ASME, article V001T02A001. Available at: https://doi.org/10.1115/MSEC2020-8469.

López-Adrio, D. and Álvarez, D. (2024) 'Polymers injection molding, process & defects', in Carou, D. and Davim, J.P. (eds) *Notes for Manufacturing Instructors: From Class to Workshop*. Cham: Springer Nature Switzerland, pp. 81–104. Available at: https://doi.org/10.1007/978-3-031-48468-1_5.

Loufakis, M. *et al.* (2024) 'Intelligent model management: Using reinforcement learning to detect data drift and retrain industrial machine learning systems', in *2024 26th International Conference on Digital Signal Processing and its Applications (DSPA)*. Moscow: IEEE, pp. 1–6. Available at: https://doi.org/10.1109/DSPA60853.2024.10510082.

Masato, D., Kazmer, D.O. and Panchal, R.R. (2023) 'Analysis of in-mold shrinkage measurement for amorphous and semicrystalline polymers using a multivariate sensor', The International Journal of Advanced Manufacturing Technology, 125(1–4), pp. 587–602. Available at: https://doi.org/10.1007/s00170-022-10755-6.

Massano, A.P. *et al.* (2024) 'Towards dynamic multiscale feedback during the injection moulding cycle of plastics', *Frontiers in Materials*, 11, article 1466753. Available at: https://doi.org/10.3389/fmats.2024.1466753.

Mateo-Casalí, M.Á., Boza, A. and Fraile, F. (2026a) 'Digital assets in zero-defect manufacturing: literature review and proposed framework', *International Journal of Production Research*, 64(17), pp. 7480–7507. Available at: https://doi.org/10.1080/00207543.2025.2563746.

Mateo-Casalí, M.Á., Boza, A. and Fraile, F. (2026b) 'Towards a Reference Architecture for Machine Learning Operations', *Computers*, 15(4), p. 218. Available at: https://doi.org/10.3390/computers15040218.

Mgbemena, C.E. and Okeagu, F.N. (2023) 'Development of an IoT-based real-time remote monitoring device for the maintenance of injection moulding machines in plastic industries', *UNIZIK Journal of Engineering and Applied Sciences*, 2(1), pp. 260–278.

Mourya, A. et al. (2023) 'An explanatory study on defects in plastic molding parts caused by machine parameters in injection molding process', Materials Today: Proceedings, 78, pp. 656–661. Available at: https://doi.org/10.1016/j.matpr.2022.12.070.

Muaz, M. *et al.* (2025) 'Explainable AI for correct root cause analysis of product quality in injection moulding', *Journal of Manufacturing Processes*, 145, pp. 371–380. Available at: https://doi.org/10.1016/j.jmapro.2025.03.114.

Narowski, P. and Wilczyński, K. (2024) 'A global approach to modeling injection molding', Polymers, 16(1), article 147. Available at: https://doi.org/10.3390/polym16010147.

O'brien, R.M. (2007) 'A Caution Regarding Rules of Thumb for Variance Inflation Factors', *Quality & Quantity*, 41(5), pp. 673–690. Available at: https://doi.org/10.1007/s11135-006-9018-6.

Ördek, B., Borgianni, Y. and Coatanea, E. (2024) 'Machine learning-supported manufacturing: a review and directions for future research', *Production & Manufacturing Research*, 12(1), article 2326526. Available at: https://doi.org/10.1080/21693277.2024.2326526.

Ozdemir, A., Uluer, O. and Guldas, A. (2004) 'Flow front advancement of molten thermoplastic materials during filling stage of a mold cavity', Polymer Testing, 23(8), pp. 957–966. Available at: https://doi.org/10.1016/j.polymertesting.2004.04.011.

Permin, E., Wohlgemuth, C. and Keller, T. (2025) 'Use-case-driven architectures for data platforms in manufacturing', Platforms, 3(3), article 15. Available at: https://doi.org/10.3390/platforms3030015.

Pramujati, B., Dubay, R. and Samaan, C. (2006) 'Cavity pressure control during cooling in plastic injection molding', Advances in Polymer Technology, 25(3), pp. 170–181. Available at: https://doi.org/10.1002/adv.20068.

Prioli, J.P.J. et al. (2025) 'Self-adaptive production performance monitoring framework under different operating regimes', Journal of Manufacturing Systems, 80, pp. 380–394. Available at: https://doi.org/10.1016/j.jmsy.2025.02.011.

Punyangarm, V. and Chotayakul, S. (2025) 'Hybrid sequence learning with interpretability for multi-class quality prediction in injection molding', *Results in Engineering*, 27, article 106408. Available at: https://doi.org/10.1016/j.rineng.2025.106408.

Puthanveettil Madathil, A. *et al.* (2025) 'A review of explainable artificial intelligence in smart manufacturing', *International Journal of Production Research*, 63(23), pp. 8654–8697. Available at: https://doi.org/10.1080/00207543.2025.2513574.

Radlbauer, E., Moser, T. and Wagner, M. (2025) 'Designing a system architecture for dynamic data collection as a foundation for knowledge modeling in industry', Applied Sciences, 15(9), article 5081. Available at: https://doi.org/10.3390/app15095081.

Rajashekarappa, M. *et al.* (2026) 'Industrial MLOps: a systematic review of architectures and implementation challenges', *Production & Manufacturing Research*, 14(1), article 2658878. Available at: https://doi.org/10.1080/21693277.2026.2658878.

Reinhardt, H. *et al.* (2024) 'Efficient data acquisition for traceability and analytics', *Procedia CIRP*, 122, pp. 73–78. Available at: https://doi.org/10.1016/j.procir.2024.01.011.

Rungtalay, N. and Kaitwanidvilai, S. (2025) 'An Adaptive Machine Learning Framework Integrating AutoML and MLOps for Two-Stage Classification in Hard Disk Drive Manufacturing', *IET Collaborative Intelligent Manufacturing*, 7(1), article e70047. Available at: https://doi.org/10.1049/cim2.70047.

Safdar, M. *et al.* (2024) 'Fundamental requirements of a machine learning operations platform for industrial metal additive manufacturing', *Computers in Industry*, 154, article 104037. Available at: https://doi.org/10.1016/j.compind.2023.104037.

Saha, N. *et al.* (2024) 'Performance evaluation framework of MQTT client libraries for IoT applications in manufacturing', *Manufacturing Letters*, 41, pp. 1237–1245. Available at: https://doi.org/10.1016/j.mfglet.2024.09.150.

Sankhye, S. and Hu, G. (2020) 'Machine learning methods for quality prediction in production', *Logistics*, 4(4), p. 35.

Schmetz, A. and Kampker, A. (2025) 'Towards Pattern Modeling for Asynchronicity in Production Data', *Procedia CIRP*, 134, pp. 37–42. Available at: https://doi.org/10.1016/j.procir.2025.03.023.

Shen, G. *et al.* (2025) 'Dynamic Monitoring Method of Polymer Injection Molding Product Quality Based on Operating Condition Drift Detection and Incremental Learning', *Polymers*, 17(22), article 3025. Available at: https://doi.org/10.3390/polym17223025.

Shin, S. *et al.* (2025) 'Recent developments of in-situ process and in-line quality monitoring in injection molding using intelligent sensors', *Sensors and Actuators A: Physical*, 383, article 116248. Available at: https://doi.org/10.1016/j.sna.2025.116248.

Tanane, B. *et al.* (2025) 'Bridging the gap between Industry 4.0 and manufacturing SMEs: A framework for an end-to-end Total Manufacturing Quality 4.0's implementation and adoption', *Journal of Industrial Information Integration*, 45, article 100833. Available at: https://doi.org/10.1016/j.jii.2025.100833.

Tayalati, F. *et al.* (2025) 'Interpreting injection molding quality defect using explainable artificial intelligence and analysis of variance', *Engineering Applications of Artificial Intelligence*, 147, article 110362. Available at: https://doi.org/10.1016/j.engappai.2025.110362.

Tsai, K.M. and Lan, J.K. (2015) 'Correlation between runner pressure and cavity pressure within injection mold', The International Journal of Advanced Manufacturing Technology, 79(1–4), pp. 273–284. Available at: https://doi.org/10.1007/s00170-014-6776-5.

Turkmanović, H. et al. (2023) 'High performance software architectures for remote high-speed data acquisition', Electronics, 12(20), article 4206. Available at: https://doi.org/10.3390/electronics12204206.

Virta, J. *et al.* (2010) 'SOA-Based integration for batch process management with OPC UA and ISA-88/95', in *2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010)*. Bilbao: IEEE, pp. 1–8. Available at: https://doi.org/10.1109/ETFA.2010.5641286.

Vukovic, M. (2024) *Cross-Phase Injection Molding Control for Thermoplastics Processing*. PhD thesis. RWTH Aachen University. Available at: https://doi.org/10.18154/RWTH-2025-05212.

Werner, M. *et al.* (2020) 'Development of a digital assistance system for continuous quality assurance in the plastic injection moulding process with a focus on self-learning algorithms', *SPE: Inspiring Plastics Professionals*.

Windmann, S. (2022) 'Data-driven fault detection in industrial batch processes based on a stochastic hybrid process model', IEEE Transactions on Automation Science and Engineering, 19(4), pp. 3888–3902. Available at: https://doi.org/10.1109/TASE.2021.3138925.

Yin, M. (2025) 'Drift-Aware Streaming Predictive Maintenance for Semiconductor Equipment', *Applied and Computational Engineering*, 197(1), pp. 205–213. Available at: https://doi.org/10.54254/2755-2721/2025.28630.

Zhang, C. *et al.* (2024) 'Spatio-temporal information analytics based performance-driven industrial process monitoring framework with cloud–edge-device collaboration', *Journal of Manufacturing Processes*, 110, pp. 224–237. Available at: https://doi.org/10.1016/j.jmapro.2023.12.036.

Zhang, X. and Zhan, H. (2026) 'A quality prediction method for injection molding products based on multi-stage feature decoupling and fusion', Expert Systems with Applications, 310, article 131372. Available at: https://doi.org/10.1016/j.eswa.2026.131372.

Zhao, P. et al. (2020) 'Intelligent injection molding on sensing, optimization, and control', Advances in Polymer Technology, 2020(1), article 7023616. Available at: https://doi.org/10.1155/2020/7023616.

Zhao, N. *et al.* (2024) 'Measurement techniques in injection molding: A comprehensive review of machine status detection, molten resin flow state characterization, and component quality adjustment', *Measurement*, 226, article 114163. Available at: https://doi.org/10.1016/j.measurement.2024.114163.

Zheng, W. (2025) *Development and Application of Integrated Capacitance-Pressure-Temperature Sensing Probe for Injection Molding*. PhD thesis. The Hong Kong University of Science and Technology. Available at: https://doi.org/10.14711/thesis-hdl167706.

Zhu, F. *et al.* (2024) 'Data-driven long-term condition-based maintenance using anomaly detection under concept drift: A case study of an ultrasonic sieve machine', *Procedia CIRP*, 126, pp. 128–133. Available at: https://doi.org/10.1016/j.procir.2024.08.312.

Appendixes

Appendix A. Experimental Hardware, Sensor Installation, and Data-Acquisition Details

Appendix A provides supporting hardware, sensor installation, and data-acquisition evidence for the experimental setup summarised in Chapter 3. The appendix preserves the detailed installation photographs and connection evidence while keeping the main methodology chapter focused and concise.

A.1 Hydraulic pressure sensor installation

The Kistler 4262A hydraulic pressure sensor was installed on the machine-side hydraulic line to monitor pressure delivery during injection and packing. This signal provided the machine-side actuation reference used to interpret pressure delivery and cycle-to-cycle variation.

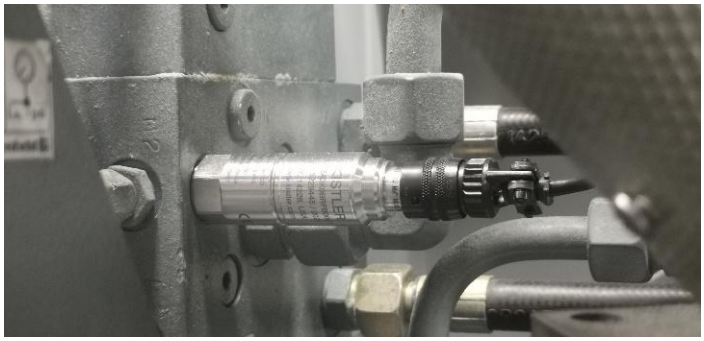


Figure A. 1. Kistler 4262A hydraulic pressure sensor used for machine-side pressure measurement.

A.2 Screw position sensor installation

The Kistler P50 linear displacement sensor was installed to monitor screw movement during each injection cycle. This signal provided information on shot delivery behaviour, cycle progression, and motion-related variation.

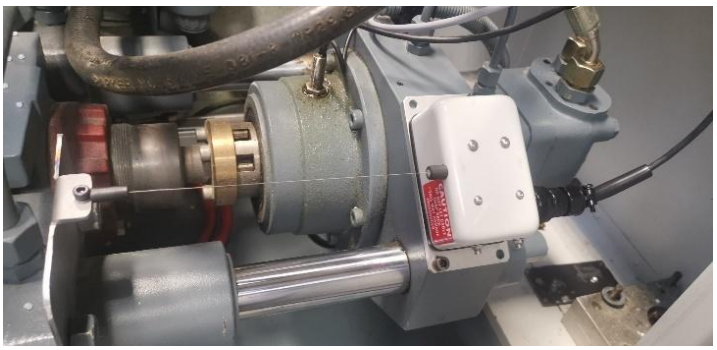


Figure A. 2. Screw position sensor installation on the injection moulding machine.

A.3 Nozzle temperature sensor installation

The nozzle temperature sensing arrangement monitored the melt thermal condition before mould entry. This measurement supported the interpretation of thermal variation, viscosity-related effects, and their relationship with pressure response and part quality.

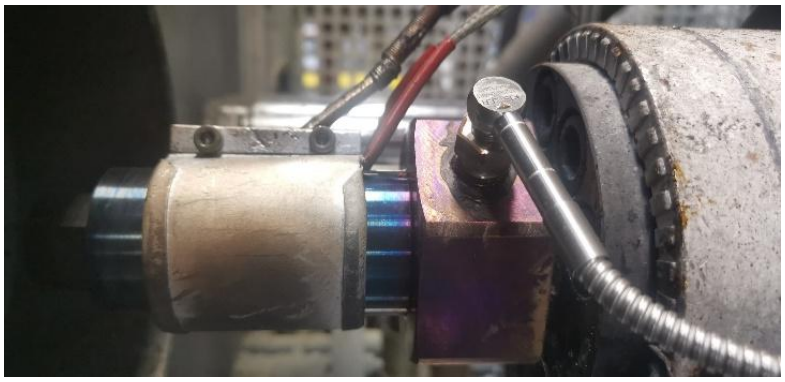


Figure A. 3. Nozzle temperature sensing arrangement used to monitor melt temperature before mould entry.

A.4 Cavity pressure and temperature sensor installation

The Kistler 6188A pressure–temperature sensor was installed inside the mould cavity to monitor the in-mould forming environment. Cavity pressure and temperature were central to the thesis because they describe the conditions experienced by the polymer during filling, packing, and cooling. The cavity pressure peak was also used as the physical anchor for event-based feature extraction.

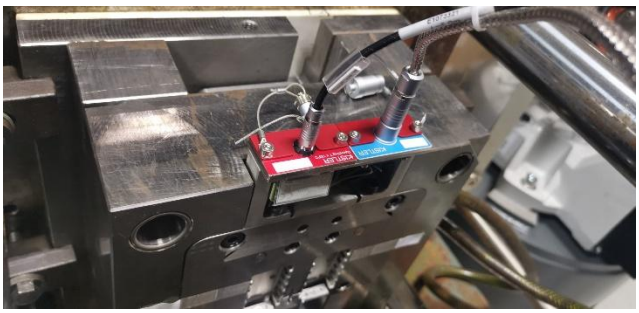


Figure A. 4. Kistler 6188A cavity pressure–temperature sensor used for in-mould process monitoring.

A.5 Heating-water temperature and flow sensor installation

The Smartflow Tracer VM sensor was installed on the mould heating-water inlet line to monitor thermal boundary conditions. This signal supported the interpretation of cooling stability, thermal drift, and their influence on cavity temperature and part quality.



Figure A. 5. Heating-water inlet temperature and flow sensor installation.

A.6 ComoNeo acquisition platform and sensor connection overview

The ComoNeo platform was used as the central data acquisition unit for the experimental trials. It collected the machine-side and mould-side sensor signals and provided both cycle-based recording and later network access for external analytics.

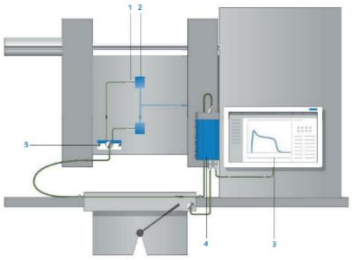


Figure A. 6. ComoNeo system and sensor connection overview.

A.7 Mould sensor connection and multi-channel acquisition

The pressure–temperature sensor installation and its connection to the ComoNeo monitoring system are shown to document the physical routing between the mould, sensor channels, and acquisition unit.

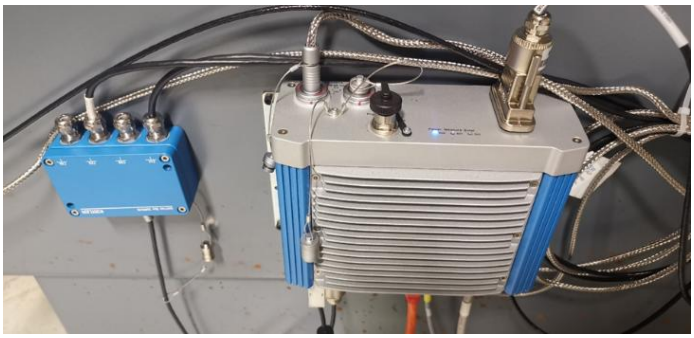


Figure A. 7. Pressure–temperature sensor installation on the mould and connection to the ComoNeo monitoring system.

A.8 Conditioning hardware and connector layout

Signal conditioning and connector hardware were used to organise sensor channels and support stable acquisition during repeated cycles. This arrangement reduced wiring complexity and supported repeatable channel mapping.

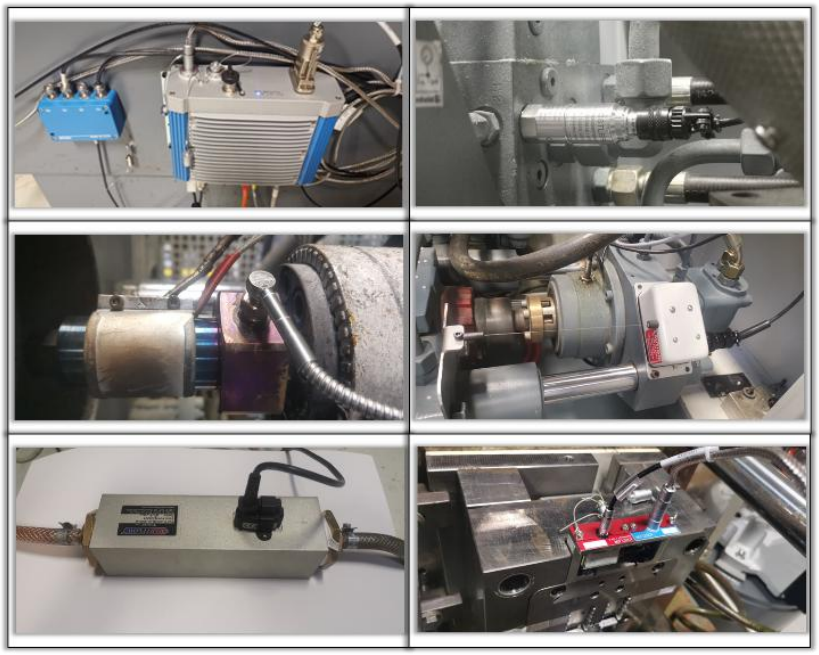


Figure A. 8. Position of conditioning and connector hardware in the mould.

A.9 Sensor channel mapping

The sensor channel mapping into the ComoNeo platform was documented to preserve traceability between physical sensors, recorded channels, and later feature extraction.

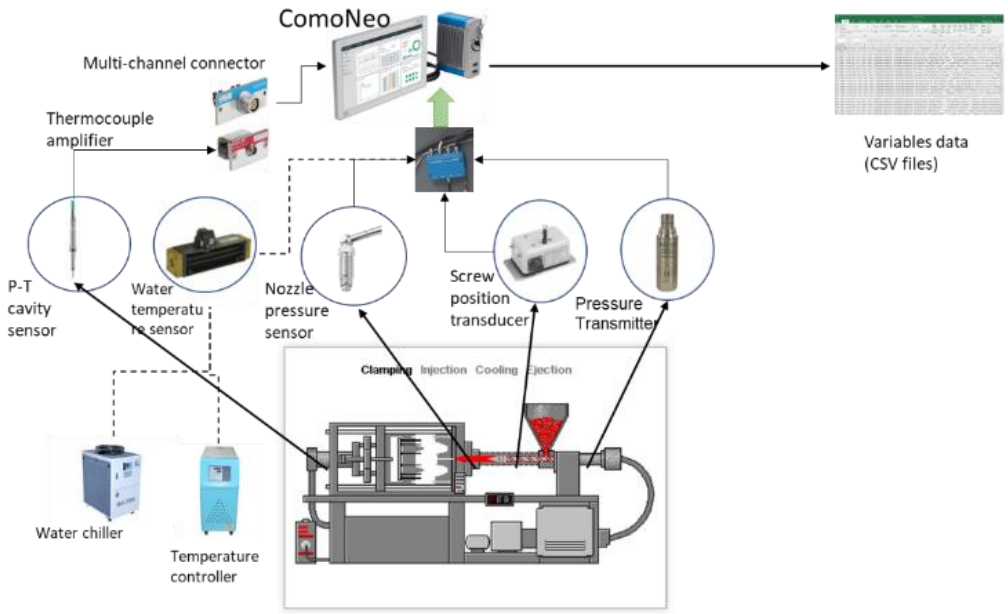


Figure A. 9. Pressure–temperature sensor installation on the mould and connection to the ComoNeo monitoring system. Labels indicate: 1) sensors, 2) cavities, 3) Kistler display screen, 4) ComoNeo unit, and 5) multi-channel connection.

Appendix B: Governed Real-Time System Interface and Runtime Evidence Views

Appendix B presents selected interface, and evidence views from the implemented governed real-time platform. The screenshots were captured from the real dashboard using the preserved validated run databases reported in Chapter 8. The purpose of this appendix is to provide visual implementation evidence for the life cycle-level workflow, event-anchored feature extraction, Champion–Challenger prediction, drift monitoring, governed retraining, final policy evaluation, timing behaviour, and run-level traceability.

The figures in this appendix are not additional experimental results. They are supporting system-interface views corresponding to the validated live evaluation evidence discussed in Chapter 8. The baseline and retrain-stage evidence databases contain 850 accepted and truth-linked runtime cycles, with the retrain-stage run including the recorded governed retraining event at CycleID 745.

B.1 Live cycle view and event-anchored alignment

The first set of views shows how the implemented governed real-time platform connects the physical cycle signal to the online event-anchored representation used by the operational prediction route. The MP peak is detected within the cycle curve and used as the alignment point for extracting the synchronised process state. This provides the compact live feature vector used by the Challenger model, while the dashboard also records the Champion and Challenger scores for the same cycle.

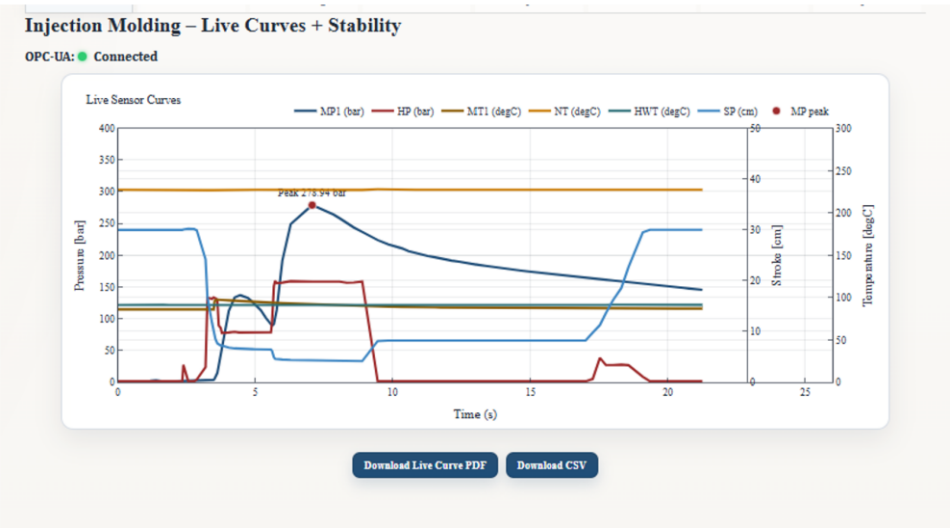


Figure B. 1. Cycle-to-digital-representation alignment view showing the detected MP peak, the aligned process state, and the Champion and Challenger scores for the final validated runtime evaluation.

This view demonstrates how the dashboard connects the physical cycle curve to the event-anchored representation used during live prediction. In this example, the MP peak is detected at approximately 7.157 s. At this aligned point, the dashboard reports HP = 157.34 bar, NT = 226.94 °C, HWT = 97.00 °C, Champion score = 0.872, and Challenger score = 0.884. The Champion and Challenger values are probability scores for the defective class. Since both values are close to 1, both model routes interpret this cycle as high-risk or defective. This supports the thesis claim that each live cycle is linked to a physically meaningful event before prediction, rather than being classified only from disconnected scalar summaries.

B.2 Cycle-level monitoring and prediction table

The cycle-level monitoring table provides the traceable record used by the dashboard during live evaluation. Each accepted cycle is stored with the MP-peak-aligned process features, the Champion and Challenger scores and labels, the truth label when available, the final governed label, and the agreement state between the model routes.

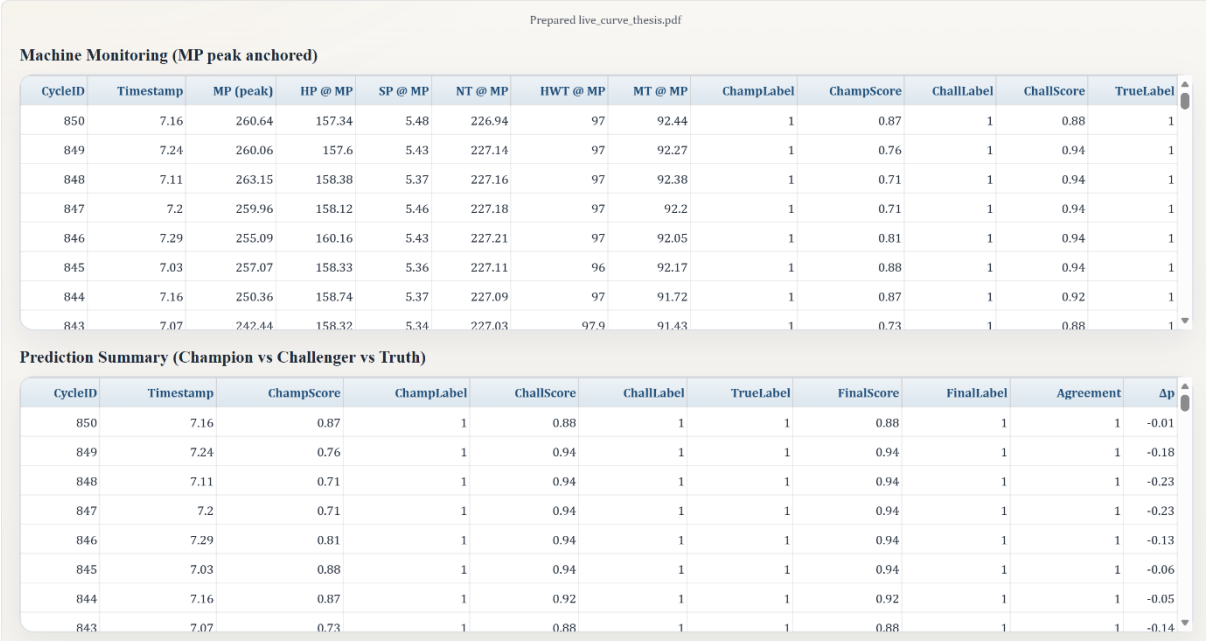


Figure B. 2. Cycle-level monitoring table showing MP-peak-aligned features, Champion and Challenger predictions, truth labels, final governed labels, agreement state, and probability difference for the final validated run.

This table shows the traceable cycle-level evidence stored by the dashboard. The final visible cycle, CycleID 850, records MP peak = 260.64 bar, HP = 157.34 bar, NT = 226.94 °C, HWT = 97.00 °C, Champion score = 0.87, Challenger score = 0.88, TrueLabel = 1, and FinalLabel = 1. In the thesis notation, TrueLabel = 1 means that the cycle is defective. The agreement between the model scores and the final label shows that this cycle was interpreted consistently as defective. This figure therefore supports the traceability argument in Chapters 7 and 8, because the system preserves the physical features, model scores, truth label, final label, and agreement state for individual cycles.

B.3 Run stability across the live evaluation window

The process stability view summarises the MP peak behaviour across the full 850-cycle runtime evaluation window. This view is useful because it shows the system-level variation across the run and provides a visual link between the live cycle records and the stability-monitoring layer.

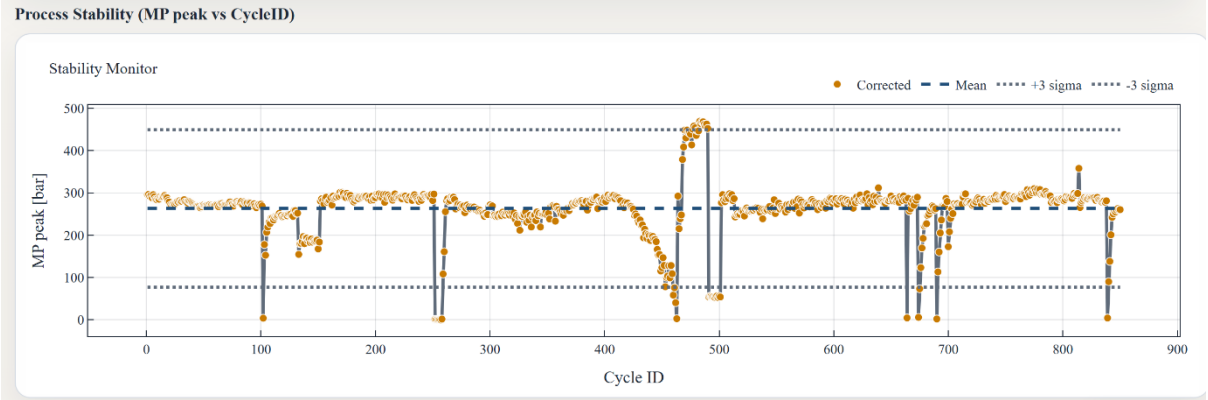


Figure B. 3. Process stability view showing MP peak variation across the complete 850-cycle live evaluation window.

This stability view shows MP peak behaviour across the complete 850-cycle live evaluation window. The plot makes clear that the MP peak does not remain constant across the run. Instead, the process moves through different regions, including lower-pressure sections and later higher-pressure sections. This is important because the thesis argues that quality monitoring in cyclic manufacturing must remain cycle aware. The figure supports that argument visually: the governed real-time platform does not only produce one final model score but also monitors how a key physical event changes across all accepted runtime cycles.

B.4 Drift monitoring and distributional change

The drift-monitoring view shows how the implemented system monitors feature-level distributional change during live operation. The dashboard reports drift indicators, including PSI, KS, and JS divergence for the main process variables. These indicators provide operational evidence for identifying changing behaviour during the live run.



Figure B. 4. Drift monitoring view showing feature-level distributional change using PSI, KS, and JS-divergence indicators during the governed runtime evaluation.

This drift-monitoring view translates distributional change into operational indicators. The dashboard reports PSI, KS, and JS divergence for the monitored process variables, including MP1, MT1, HP, SP, NT, and HWT. These values indicate how far the live feature distributions have moved from their reference behaviour. Larger values mean stronger evidence that the process has shifted. This connects directly to Chapter 8, where drift is treated as an observable runtime condition rather than a hidden modelling issue. The figure supports the drift-aware part of the governed real-time architecture.

B.5 Governed retraining and evidence gates

The governed retraining view records the relationship between drift monitoring, correction evidence, gate conditions, and the controlled Challenger update. The view shows the latest successful retraining event in the run and separates the pre-retraining evidence window from the post-retraining section.

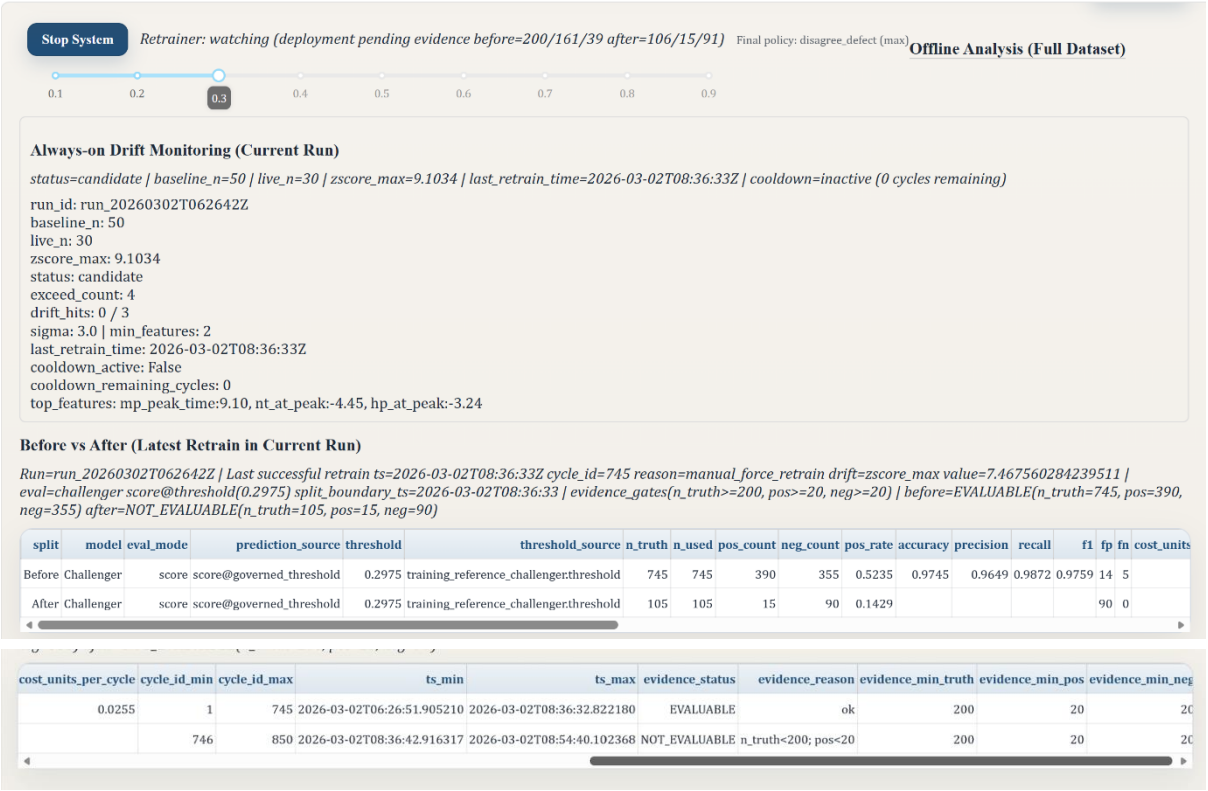
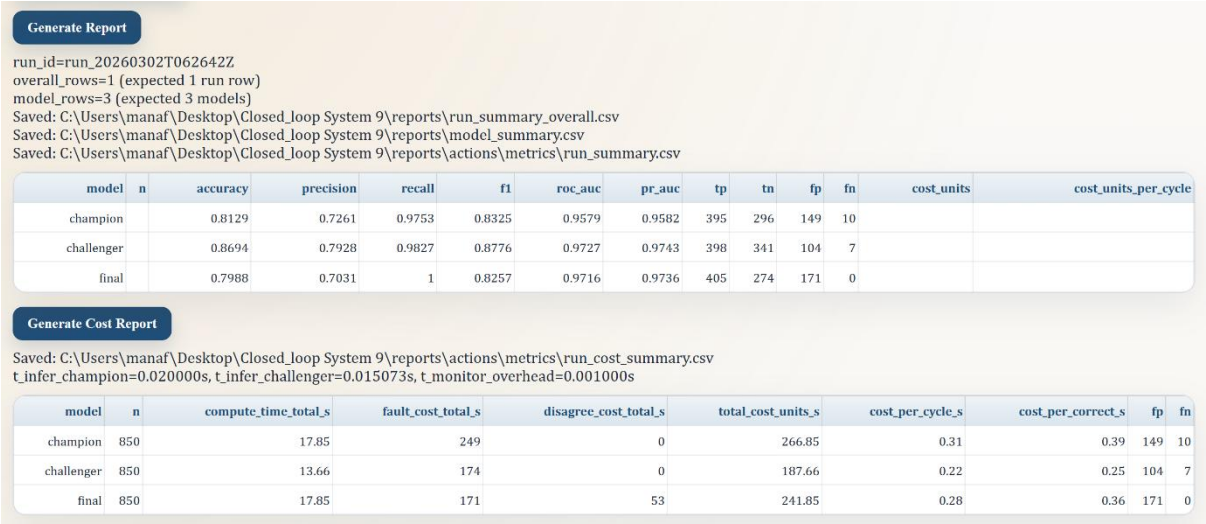


Figure B. 5. Governed retraining evidence view showing the drift timeline, retraining marker, evidence-gate status, and recorded retraining context at CycleID 745.

This governance view shows how drift evidence and retraining evidence are linked in the implemented system. The retraining marker appears at CycleID 745, which matches the recorded retrain-stage result in Chapter 8. The thesis reports this as one successful governed Challenger retraining event, supported by 40 human corrections. The drift context is also meaningful: the main PSI peak occurs earlier, around CycleID 725, and the retraining event follows at CycleID 745. This figure shows that retraining was not an uncontrolled model change. It was recorded as a traceable event within the drift-aware governance structure.

B.6 Final operational metrics and cost report

The run-level report view summarises the final classification and cost-related evidence generated by the implemented system. The report includes the Champion, Challenger, and Final outputs, allowing the deployed final policy to be compared with the two model routes under the same truth-linked runtime cycles.



model	n	accuracy	precision	recall	f1	roc_auc	pr_auc	tp	tn	fp	fn	cost_units	cost_units_per_cycle
champion		0.8129	0.7261	0.9753	0.8325	0.9579	0.9582	395	296	149	10		
challenger		0.8694	0.7928	0.9827	0.8776	0.9727	0.9743	398	341	104	7		
final		0.7988	0.7031	1	0.8257	0.9716	0.9736	405	274	171	0		

model	n	compute_time_total_s	fault_cost_total_s	disagree_cost_total_s	total_cost_units_s	cost_per_cycle_s	cost_per_correct_s	fp	fn
champion	850	17.85	249	0	266.85	0.31	0.39	149	10
challenger	850	13.66	174	0	187.66	0.22	0.25	104	7
final	850	17.85	171	53	241.85	0.28	0.36	171	0

Figure B. 6. Run-level metrics and cost report for the retrain-stage run, showing Champion, Challenger, and Final performance together with cost-related evidence.

This report view summarises the main retrain-stage classification and cost-sensitive evidence generated by the dashboard. The dashboard reports Champion accuracy = 0.8129 and F1 = 0.8325, Challenger accuracy = 0.8694 and F1 = 0.8776, and Final accuracy = 0.7988 and F1 = 0.8257. The same view shows the operational trade-off between the raw Challenger route and the governed Final output. The Challenger produces FP = 104 and FN = 7, while the Final output produces FP = 171 and FN = 0. This means the Final policy removes all missed defective cycles but does so by increasing the number of false alarms.

The dashboard presents an illustrative error-penalty calculation rather than a measurement of financial expenditure. In this view, false-positive errors receive a reference penalty of 1, while false-negative errors receive an assumed penalty of 10. The resulting fault-cost term is $FP + 10FN$ which corresponds to the unnormalised numerator of the relative decision-loss measure defined in Equation 26 when $r = 10$ For the 850 truth-labelled cycles in the retrain-stage run, this produces values of 249 for the Champion, 174 for the Challenger, and 171 for the Final output. Dividing these values by 850 gives the corresponding normalised relative decision loss reported in Table 55.

The dashboard also reports a separate composite cost indicator that incorporates additional processing- and disagreement-related terms. This is an implementation-specific reporting quantity and should not be treated as equivalent to either the normalised relative decision loss in Equation 26 or the processing-weighted false-positive burden reported in Table 56. The additional

dashboard terms were not established from measured industrial financial penalties, and the composite indicator is therefore retained as an illustrative interface output rather than used to establish economic superiority between the decision paths. The principal Chapter 8 comparisons remain based on the explicitly defined relative decision-loss sensitivity and the separately reported processing-weighted false-positive burden.

B.7 Final operational quality-evaluation view

The quality-evaluation view presents the final governed output using all truth-linked runtime cycles. It includes the confusion matrix and precision-recall behaviour for the selected metric source. This provides a direct interface-level view of the final deployed decision output.



Figure B. 7. Final operational quality-evaluation view showing the confusion matrix and precision-recall behaviour for the governed Final output across all 850 truth-linked runtime cycles.

This quality-evaluation view shows the Final operational output across the 850 truth-linked runtime cycles. The confusion matrix reports TN = 274, FP = 171, FN = 0, and TP = 405. These values mean that all 405 defective cycles were captured by the Final output, while 171 good cycles were conservatively rejected. The absence of false negatives is the key operational result in this view. It visually

supports the thesis interpretation that the Final policy prioritises avoiding defect escape, even though this increases the false-positive burden.

B.8 Champion–Challenger disagreement monitoring

The model-monitoring view compares Champion and Challenger defect probabilities over the late-stage run window. It also shows the probability difference between the two routes and highlights disagreement points. This view supports the thesis argument that disagreement is a useful operational signal, not only a modelling inconvenience.

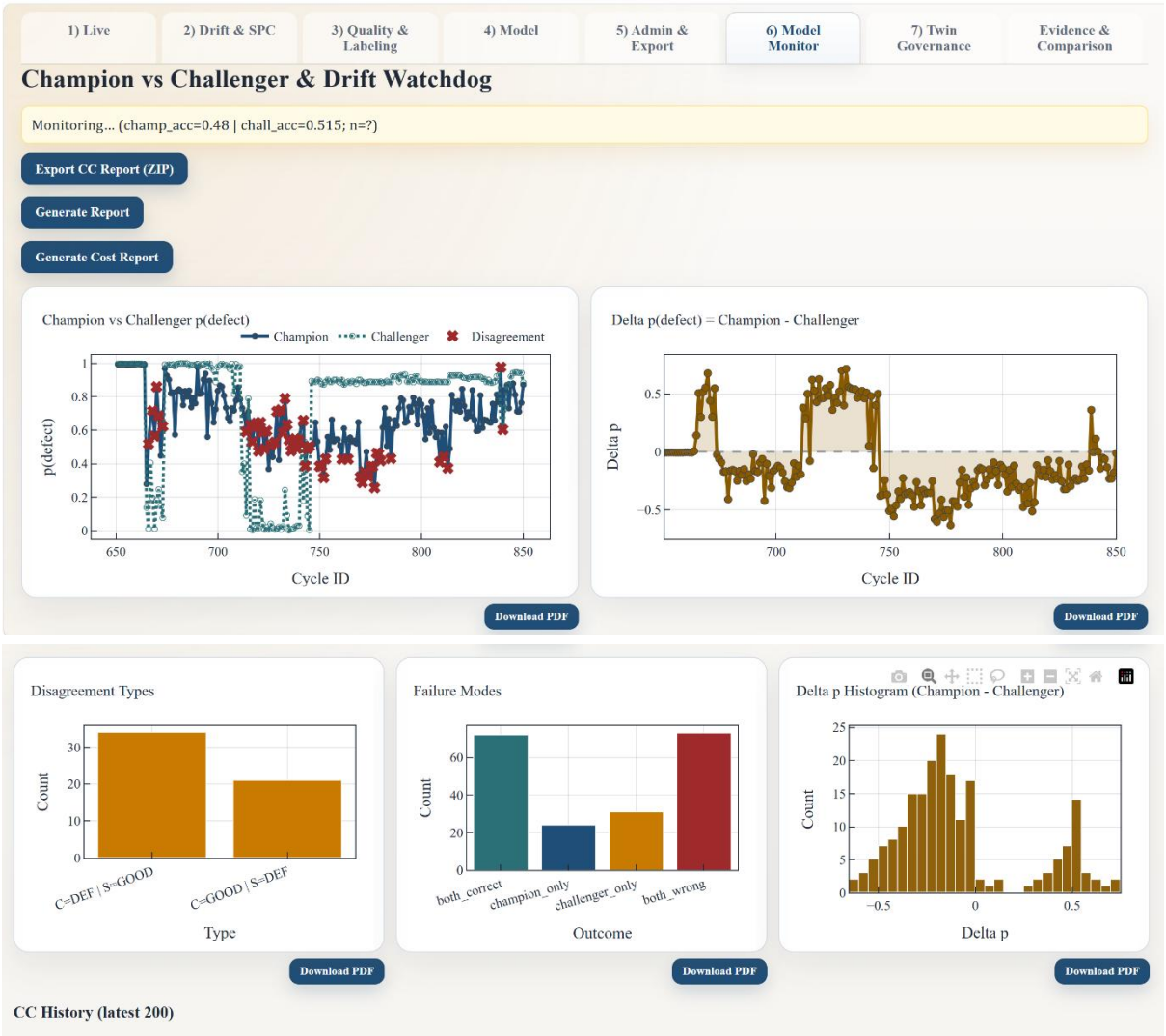


Figure B. 8. Champion–Challenger monitoring view showing predicted defect probabilities, disagreement points, and probability-difference behaviour across the late-stage run window.

This model-monitoring view shows how the Champion and Challenger probabilities behave over the late-stage section of the run. The plot highlights periods when the two routes agree and when they disagree. The probability-difference trace shows the size and direction of the gap between the two model scores. This is important because the thesis does not treat disagreement as noise to be ignored. Instead, disagreement is retained as operational evidence and is resolved through the locked Final policy. The figure therefore supports the governed decision structure described in Chapter 7 and evaluated in Chapter 8.

B.9 Rolling discrimination and model monitoring

The rolling ROC-AUC view provides time-local evidence of discrimination performance across the run. Instead of reporting only a single aggregate AUC value, the dashboard tracks how model discrimination changes across rolling windows of truth-linked runtime cycles.

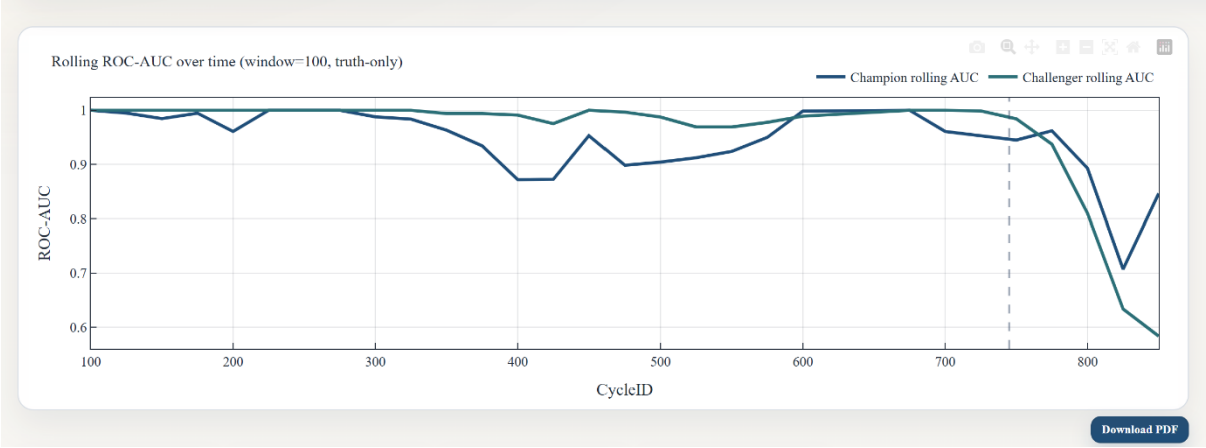


Figure B. 9. Rolling ROC-AUC monitoring view showing Champion and Challenger discrimination behaviour over the live evaluation window.

This rolling ROC-AUC view shows that model discrimination is monitored over time, rather than as a single, whole-run value. Both Champion and Challenger performance are tracked across rolling truth-linked windows. The value of this figure is that it translates model reliability into a time-local monitoring view. A model may have strong aggregate performance throughout the run, while still showing weaker behaviour in specific sections. This supports the thesis argument

that runtime evaluation should include temporal monitoring of model behaviour, not only final summary metrics.

B.10 Runtime latency and throughput

The latency and throughput view summarises the timing behaviour of the implemented governed real-time platform. It separates Champion inference latency, Challenger inference latency, total model latency, and overall processing time. This supports the real-time feasibility evidence discussed in Chapter 8.

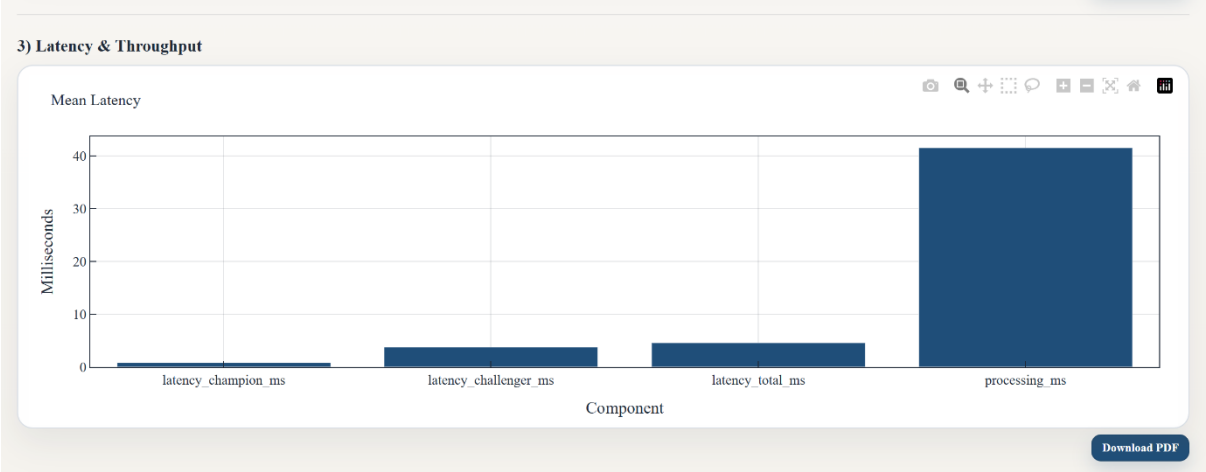


Figure B. 10. Runtime latency and throughput view showing mean latency components for Champion inference, Challenger inference, total model latency, and end-to-end processing time.

This timing view shows the runtime feasibility of the implemented governed real-time platform. In the retrain-stage run, Chapter 8 reports mean processing time = 41.5582 ms and p95 processing time = 60.1806 ms. At the model level, mean Champion latency = 0.7931 ms, mean Challenger latency = 3.7732 ms, and mean total model latency = 4.5663 ms. These values show that model inference is only a small part of the full processing path. The wider processing time includes feature preparation, validation, decision logic, storage, and dashboard update. This supports the thesis claim that the system operated as a live cycle-level decision environment rather than as delayed offline analysis

Appendix C: Key Software Implementation Excerpts

Appendix C presents selected excerpts from the Python implementation developed for this research. The purpose is not to reproduce the complete software codebase, but to provide implementation evidence for the principal methodological functions described in Chapters 4–8. The selected excerpts cover event-anchored cycle representation, model training and inference, full-curve processing, the Final decision policy, drift monitoring, retraining governance, and persistent cycle-level traceability. User-interface, plotting, and other supporting software functions are omitted because they are not central to the research methodology.

C.1 Event-anchor detection

Source: peak_detect.py

Function: detect_mp_peak()

```
mp = np.asarray(mp_raw, dtype=float)
t = np.asarray(time_raw, dtype=float)

mp = np.nan_to_num(mp, nan=-np.inf)

idx = int(np.argmax(mp))

mp_peak = float(mp[idx])
t_peak = float(t[idx])

return idx, mp_peak, t_peak
```

This function identifies the event anchor used by the live event-anchored representation. The index corresponds to the first occurrence of the maximum recorded MP1 cavity-pressure value within the completed cycle. The same index is subsequently used to sample the remaining process channels.

C.2 Synchronised event-state extraction

Source: services/snapshot.py

Function: _process_snapshot_sync()

```
peak_time = safe_val(t, idx)
peak_mp = safe_val(mp, idx)
peak_mt = safe_val(mt, idx)
peak_hp = safe_val(hp, idx)
peak_sp = safe_val(sp, idx)
peak_nt = safe_val(nt, idx)
peak_hwt = safe_val(hwt, idx)

raw_peak_features = {
   "time": peak_time, "mp": peak_mp, "mt": peak_mt,
   "hp": peak_hp, "sp": peak_sp,
   "nt": peak_nt, "hwt": peak_hwt,
}
```

The repeated use of the same index is important because the variables are not independently selected peaks. Instead, they represent the synchronised process state at the MP1 anchor event.

C.3 Five-fold cross-validation and out-of-fold evaluation

Source: newpeak.py

Function: main()

```
X = df[args.features].to_numpy()
y = df[args.label].astype(int).to_numpy()

models = build_models()
oof = {n: {"y_true": [], "y_prob": [], "y_pred": []}
     for n in all_model_names}

skf = StratifiedKFold(
   n_splits=5,
   shuffle=True,
   random_state=RANDOM_STATE
)

for fold, (tr, te) in enumerate(skf.split(X, y), start=1):
   Xtr, Xte, ytr, yte = X[tr], X[te], y[tr], y[te]
```

```
clf.fit(Xtr, ytr)
y_score = clf.predict_proba(Xte)[:, 1]

oof[name]["y_true"] += yte.tolist()
oof[name]["y_prob"] += y_score.tolist()

y_true = np.array(oof[name]["y_true"])
y_prob = np.array(oof[name]["y_prob"])

roc = roc_auc_score(y_true, y_prob)
pra = average_precision_score(y_true, y_prob)
```

These excerpts show the stratified five-fold validation procedure and confirm that ROC-AUC and PR-AUC were calculated using aggregated out-of-fold probabilities rather than

predictions from cycles used to fit the corresponding model.

C.4 Full-curve Champion representation

Source: core/fullcurve_inference.py

Function: predict_champion_fullcurve()

```
a = np.asarray(vals if vals is not None else [], dtype=float)

# Clip exactly like training
if ch == "HWT":
    a = np.clip(a, 0, 150)
elif ch == "MP1":
    a = np.clip(a, 0, 500)
elif ch == "MT1":
    a = np.clip(a, 0, 200)
elif ch == "HP":
    a = np.clip(a, 0, 300)
elif ch == "SP":
    a = np.clip(a, 0, 100)
elif ch == "NT":
    a = np.clip(a, 0, 300)

# Signal resample
a = _resample_signal(a, length)

# Channel normalisation
a = (a - float(mu[i])) / (float(sd[i]) + 1e-8)

vecs.append(a)

X_mat = np.stack(vecs, axis=1)
X = X_mat.reshape(1, -1)
```

Each of the six complete process signals is clipped, resampled to the fixed cycle length and normalised using the stored training statistics. The six vectors are then stacked and flattened to form the full-curve input used by the fixed Champion model. With six channels and a resampling length of 970, the input contains 5,820 values per cycle.

C.5 Challenger inference and stored decision threshold

Source: core/inference.py

Function: predict_challenger()

```
feat_cols = list(identity["features"])
thr = float(identity["threshold"])

x = ensure_csv_schema(x_df.copy())
x = add_interaction_features(x)

for c in feat_cols:
    if c not in x.columns:
        x[c] = 0.0

x = x[feat_cols]

proba = float(model.predict_proba(x)[:, 1][0])
label = int(proba >= thr)
```

The event-aligned process state is converted into the stored Challenger feature contract, arranged in the required feature order and passed to the classifier. The output probability represents the estimated probability of the defective class, and the corresponding model-version threshold is then used to generate the binary Challenger decision.

C.6 Final decision policy

Source: services/snapshot.py

Function: apply_final_policy()

```
def apply_final_policy(
    champ_label: int,
    champ_proba: float,
    chall_label: int,
    chall_proba: float,
    champ_available: bool = True,
    chall_available: bool = True,
) -> tuple[int, float, str]:

    mode = _resolve_final_policy_mode()

    if not champ_available and not chall_available:
        return 0, 0.5, "policy_no_model_available"

    if champ_available and not chall_available:
        return champ_label, champ_proba, \
            "policy_champion_fallback_challenger_unavailable"

    if chall_available and not champ_available:
        return chall_label, chall_proba, \
            "policy_challenger_fallback_champion_unavailable"

    if mode == "challenger_only":
        return chall_label, chall_proba, "policy_challenger_only"

    if mode == "champion_only":
        return champ_label, champ_proba, "policy_champion_only"

    if champ_label != chall_label:
        p = max(champ_proba, chall_proba)
        return 1, float(p), "policy_disagree_defect"

    return chall_label, chall_proba, "policy_agree_use_challenger"
```

The Final layer is therefore a deterministic policy rather than a third predictive model. Under the evaluated disagreement policy, disagreement between Champion and Challenger produces a defective Final decision, while agreement retains the common classification.

C.7 Drift calculation

Source: services/retrainer.py

Function: _compute_drift_zscore()

```
baseline = d.iloc[:DRIFT_BASELINE_WINDOW]
live = d.iloc[-DRIFT_LIVE_WINDOW:]

zscores = {}

for key, col in cols.items():
   mu = float(np.nanmean(baseline[col].values))
   sd = float(np.nanstd(baseline[col].values))
   lv = float(np.nanmean(live[col].values))

   if not np.isfinite(mu) or not np.isfinite(lv) \
         or not np.isfinite(sd) or sd == 0.0:
      continue

   zscores[key] = float((lv - mu) / sd)

abs_exceed = [
   k for k, z in zscores.items()
   if abs(z) >= DRIFT_SIGMA
]

max_abs_z = max(abs(z) for z in zscores.values())

return "zscore_max", float(max_abs_z), details
```

Runtime drift awareness compares the mean of the recent window with the baseline mean after scaling by the baseline standard deviation. The maximum absolute standardised change is retained as $\text{zscore}_{\text{max}}$. This quantity provides evidence of distributional change but does not, by itself, authorise model retraining.

C.8 Retraining admission conditions

```
training_ready = (
    n_rows >= min_rows
    and n_pos >= min_pos
    and n_neg >= min_neg
    and pos_rate is not None
    and min_rate <= float(pos_rate) <= max_rate
)

if int(stats_new.get("corrected_count", 0)) < MIN_NEW_CORRECTIONS:
    reasons.append("insufficient_new_corrections")

if (
    int(stats_new.get("pos_count", 0)) < MIN_CORRECTED_POS
    or int(stats_new.get("neg_count", 0)) < MIN_CORRECTED_NEG
):
    reasons.append("insufficient_corrected_balance")

if corr.get("available"):
    c_rate = _as_float(corr.get("correction_rate"))
    if c_rate is not None and c_rate > MAX_CORRECTION_RATE:
        reasons.append("correction_rate_too_high")

payload["allow_retrain"] = bool(len(reasons) == 0)
```

```
REGIME_STABILITY_WINDOW = 200
REGIME_MAX_POSRATE_DELTA = 0.20
REGIME_MAX_IMBALANCE_RATIO = 4.0
```

These checks illustrate that retraining admission is not based on drift alone. The implementation verifies data sufficiency, representation of both classes, reviewed correction evidence and regime suitability before permitting the governed retraining pathway.

C.9 Governed Challenger retraining

Source: services/retrainer.py

Function: force_retrain()

```
def force_retrain(
    *,
    freeze_run_id: str | None = None,
    freeze_cycle_id: int | None = None,
    regime_snapshot: dict | None = None,
) -> str:
    """
    Manually trigger retrain (bypasses drift checks).

    - Checks Human Gate.
    - Checks Regime Gate.
    - Uses the frozen run and cycle context where provided.
    """
```

```
label_mode = (
    "human_override"
    if _app_mode() == "human"
    else "true_label"
)

os.environ["RETRAIN_LABEL_MODE"] = label_mode

model_path = core_retrain.train_and_save(
    min_rows=30,
    label_mode=label_mode
)
```

The manually initiated retraining pathway bypasses the requirement for an automatic drift trigger, but it does not bypass the remaining human-evidence, regime and training-safety checks. The retraining operation updates the Challenger only.

C.10 Cycle-level persistence and traceability

```
def insert_cycle_row(data: Dict[str, Any]) -> int:

  if not data.get("ts"):
    data["ts"] = datetime.now(timezone.utc).isoformat()

  if not data.get("run_id"):
    data["run_id"] = RUN_ID

  if not str(data.get("final_policy_mode", "")).strip():
    data["final_policy_mode"] = str(
      FINAL_POLICY_MODE or "disagree_defect"
    ).strip().lower()

  if not str(data.get("final_reason", "")).strip():
    data["final_reason"] = \
      f"policy:{data['final_policy_mode']}"
```

Persistent cycle records provide the connection between the physical cycle, model outputs, Final decision, model versions, timing and later truth-linked evaluation. This storage layer is therefore part of the traceability mechanism rather than only a software database function.