The Application of System Dynamics to Project Management
An Integrated Methodology (SYDPIM)

By

Alexandre G. Rodrigues

Thesis submitted to the Department of Management Science in partial fulfilment of the requirements for the degree of:

Doctor of Philosophy

at the

UNIVERSITY OF STRATHCLYDE

July, 2000

Glasgow, United Kingdom

July, 2000.

Appendices

Appendix A – Definition of key terms and expressions

The description of the four elements of the SYDPIM process model requires the use of unambiguous terms and expressions. The key ones employed throughout are defined below in sub-groups.

Definitions of project behaviour

- *project behaviour* – a set of continuous patterns or discrete series of data-points over-time. Each pattern or series of data-points describes how a variable of a project model changes over-time. The term is employed independently from the source of the information, but this information is available and stored as data.
- *project past behaviour* – sub-component of "project behaviour" considering only the period of time from the beginning of the project to the present moment.
- *past segment of project behaviour* – same as "project past behaviour" but considering explicitly that this is the past sub-component of the "project behaviour". It is assumed that the project behaviour has two segments: past (includes present) and future.
- *present data-point of past behaviour* – the last data-point of all patterns of the "project past behaviour" that corresponds to the present moment in time. In SYDPIM, the present moment corresponds to the period of time where progress is being assessed and the project plan revised. For purposes of specifying the project behaviour, this is considered as part of its past segment.
- *project future behaviour* – sub-component of "project behaviour" considering only the period of time from the present moment (excluded) to the anticipated moment when the project will be completed.
- *future segment of project behaviour* – same as "project future behaviour" but considering explicitly that this is the future sub-component of the "project behaviour". It is assumed that the project behaviour has two segments: past (includes present) and future.
- *project planned behaviour* – same as "project future behaviour" but specifying that the source of the information is a project *plan*. This plan is available and can be stored either in the PERT/CPM model or in the SD model.

- *steady behaviour* – a project behaviour where the patterns of the variables "estimated cost at completion" and "estimated schedule at completion" are constant over time within the period specified. This implies a project being implemented with no cost and schedule slippage or compression.
- *unsteady behaviour* – a project behaviour where the patterns of the variables "estimated cost at completion" and "estimated schedule at completion" vary over time within the period specified. This implies a project being implemented with some cost and schedule slippage or compression.

Specification of project behaviours from SD and PERT/CPM model

- *extract project past behaviour from PERT/CPM model* – operation that consists in specifying the patterns of the "project past behaviour" from one or more PERT/CPM plans stored in the PERT/CPM model. This is, the PERT/CPM model is the source of information used. The set of patterns that can be defined and their granularity over-time will depend on the data available in the PERT/CPM model and the number of PERT/CPM plans stored in the model.
- *extract project future behaviour from PERT/CPM model* – operation that consists in specifying the patterns of the "project future behaviour" from one or more PERT/CPM plans stored in the PERT/CPM model. This is, the PERT/CPM model is the source of information used. The set of patterns that can be defined and their granularity over-time will depend on the data available in the PERT/CPM model and the number of PERT/CPM plans stored in the model. The "current PERT/CPM plan" is the most important one and in SYDPIM it is of no use to carry out this operation is the this plan is not available in the PERT/CPM model.
- *PERT/CPM past behaviour* – the "project past behaviour" as extracted from the PERT/CPM model.
- *PERT/CPM future behaviour* – the "project future behaviour" as extracted from the PERT/CPM model.
- *PERT/CPM project behaviour* – the "project behaviour" as extracted from the PERT/CPM model, including both past and future behaviours.
- *SD project past behaviour* – set of continuous patterns of "project past behaviour" produced by the simulation of the SD model.

- *SD project future behaviour* – set of continuous patterns of "project future behaviour" produced by the simulation of the SD model.
- *SD project behaviour* – set of continuous patterns of "project behaviour" produced by the simulation of the SD model, including both past and future behaviour.
- *Reproduce project behaviour* – the patterns of the "project behaviour" produced by the simulation of the SD model match a given set of patterns of project behaviour. This matching consists in running and passing statistical tests of "goodness-of-fit". The given patterns of "project behaviour" are defined as series of discrete data-points over-time.

Metrics and project results:

- *Project result metrics* – data collected from the project referring to its outcome up to present. For example, lines of code developed, effort spent, actual completion date of a task, or defects detected. The term "metrics" implies that the data is measured within a degree of accepted accuracy and can be used in the PERT/CPM and/or SD model.
- *Project actual results* – information describing the project outcome up to present. This includes "project result metrics" but it can be more than that, also including more subjective information or the reporting of the certain complex events.
- *SYDPIM metrics database* – a database containing the project result metrics specified in the "SYDPIM metrics plan". These metrics refer to the project outcome up to present and are updated in every control cycle as result metrics are collected.
- *SYDPIM metrics plan* – a metrics plan part of the SYDPIM methodology. It specifies which metrics are to be collected, how they are measured, and how they are to be used in the SYDPIM process framework.
- *Improve past behaviour from metrics database* – operation that consists in enhancing a given "project past behaviour" (typically one "extracted from the PERT/CPM model"), with the data available in the SYDPIM metrics database. For example, the pattern "cumulative defects detected" cannot be extracted

from the PERT/CPM model but can be derived from the SYDPIM metrics database.

- *Derive past behaviour pattern from metrics database* – definition of a specific behaviour pattern of the "project past behaviour" as a series of discrete data-points over time using the data available in the SYDPIM metrics database.

Model readjustment and calibration:

- *Re-plan project future in PERT/CPM model* – operation that consists in changing the future segment of the current PERT/CPM plan (see "future segment of PERT/CPM plan) to re-plan the work remaining. This is a conventional operation of the traditional PM framework: tasks are created and/or re-scheduled, etc..
- *Re-plan project future in SD model* – operation that consists in changing the future segment of the SD plan in the SD model (see "future segment of SD plan") to re-plan the work remaining. This consists of implementing exogenous control decisions in the model (e.g. changing the future profile of allocated resources) as well as changing future control policies (e.g. changing the pattern of "willingness to hire more people in face of schedule pressure").
- *Re-calibrate SD model for past behaviour* – calibrate the SD model, changing only model variables that refer to the project past, so that the model reproduces a given "project past behaviour" (see "reproduce project behaviour"). Typically the "project past behaviour" was extracted from the PERT/CPM model and eventually improved from the metrics database.
- *Re-calibrate SD model for planned behaviour* – calibrate the SD model, changing only variables that refer to the project future, so that the model reproduces a given "project planned behaviour" (see "reproduce project behaviour"). Typically, the "project planned behaviour" will be extracted from the PERT/CPM model (see "PERT/CPM future behaviour"). In some cases it may be considered changing some variables in the SD model that refer to the project past, in which case this will be mentioned explicitly.
- *Re-calibrate SD model for future behaviour* – calibrate the SD model, changing only variables that refer to the project future, so that the model reproduces a given "project future behaviour" (see "reproduce project behaviour"). Typically,

the "project future behaviour" will be extracted from the PERT/CPM model (see "PERT/CPM future behaviour"). In some cases it may be considered changing some variables in the SD model that refer to the project past, in which case this will be mentioned explicitly.

- *Re-calibrate SD model for PERT/CPM plan* – same as "Re-calibrate SD model for planned behaviour", but making explicit that the "project planned behaviour" is the "project future behaviour" to be extracted, or already extracted, from the PERT/CPM model (see "extract project future behaviour from PERT/CPM model").
- *Readjust PERT/CPM model to SD plan* – change the "future segment of the PERT/CPM plan" so that the "PERT/CPM future behaviour" matches the "SD project future behaviour". This matching consists in running and passing statistical tests of "goodness-of-fit". This operation implies changing the PERT/CPM plan for the project future so that it becomes consistent with the one developed in the SD model.
- *Update PERT/CPM model with actual results* – update of the past segment of the current PERT/CPM plan with the result metrics collected. This generally consists of entering actual results in the task fields, like the actual completion date or effort spent.

Use of models:

- *PERT/CPM analysis of project future* – conventional critical-path based analysis of the project future using the PERT/CPM model. Typically this involves identifying the critical path, task floats, cost and resources profiles, and sometimes Monte Carlo simulation analysis.
- *PERT/CPM diagnosis of project past* – conventional critical-path based analysis of the project past using the PERT/CPM model. Typically this involves identifying changes in the critical path, tasks' floats and duration, cost and resources profiles, against a base-line previous plan.
- *SD analysis of project future* – simulation of the project future, identification of deviations from targets, assessment of performance and diagnosis of causes. What-if analysis is carried to identify best corrective actions in the current project plan for the future.

- *SD diagnosis of project past* – simulation of project past, identification of deviation from established targets, assessment of performance and diagnosis of causes. "What-would-have-happened-if" analysis is carried out to identify whether better or worse results could have been achieved and thereby support process improvement initiatives.

Models and project plan:

- *PERT/CPM model* – a logical network model that stores one or more project work plans and other data about the project (e.g. resources availability over-time). Typically, but not necessarily, the model will contain a "current PERT/CPM plan" and an "initial PERT/CPM plan".
- *PERT/CPM plan* – a project work plan in the form of a logical network specifying the task schedules, task dependencies and resource allocation to the tasks (conventional PM tool).
- *Current PERT/CPM plan* – the most updated PERT/CPM plan available. It is updated in terms of project past and project future.
- *Past PERT/CPM plan or plans* – a set of PERT/CPM plans that were elaborated in the past. They may no longer reflect what is specified in the "current PERT/CPM plan". The most relevant for SYDPIM is the "initial PERT/CPM plan". As the project unfolds and this plan is updated, the organisation may (or not) store the evolving versions over-time of this plan.
- *Initial PERT/CPM plan* – the "PERT/CPM" plan developed in the "day zero" of the project, generally prior to implementation.
- *Past segment of PERT/CPM plan* – sub-component of a PERT/CPM plan which includes only those tasks which are already complete or are in progress (i.e. they started but are not finished).
- *Future segment of PERT/CPM plan* – sub-component of a PERT/CPM plan which includes only those tasks which have not started yet or are in progress (i.e. they started but are not finished).
- *SD plan* – a project work pan as specified in a SD project model. The information contained in this plan depends on the structure and details of the specific SD model. Typically, it will include a set of major phases, their

scheduled dates, scope and resource allocation, and also project control policies and exogenous decisions.

- *Future segment of SD plan* – same as "SD plan" but considering only those variables that refer to the project future.
- *Past segment of SD plan* -- same as "SD plan" but considering only those variables that refer to the project past.
- *Project plan* – work plan specified for the project specified in one of the models, generally in both.

Appendix B – Tables

Generic SD Process	Forrester (1961)	Richardson (1981)	Richmond (1990)	Coyle (1996)	Wolstenholme (1990)
1. Problem identification and system definition	1. Goal	1. Problem identification	1. Focus the effort	1. Problem recognition	**1. Qualitative System Dynamics** 1.1 Identify cause of concern
2. Development of ID to represent the system feedback structure	2. Verbal description of the situation	2. System conceptualisation: dynamic representation of the problem identifying the reference modes of behaviour and feedback structure	---	2. System description	1.2 Development of IDs
3. Qualitative system analysis: process and policy re-design (informal inference)			---	3. Qualitative analysis	1.3 Qualitative analysis of feedback structure and modes of behaviour (informal inference)
4. Development of a simulation model capturing the feedback structure	3. Mathematical model	3. Model formulation: translation of feedback structure into a simulation model through the use of equations 4. Analysis of model behaviour 5. Model evaluation	2. Map: model structure 3. Model: quantify relationships 4. Simulate: validate model	4. Simulation modelling (if necessary). Translation of ID to a simulation model.	**2. Quantitative System Dynamics** (if necessary). 2.1 Model development and validation
5. Quantitative system analysis: process and policy re-design (formal test of parameter and structural model changes)	4. Simulation 5. Interpretation 6. System revision	6. Policy analysis	5. Communicate	5. Policy testing and design	2.2 Improvement of system behaviour
6. Re-iterate at any stage if appropriate	7. Repeated experimentation	7. Model use or implementation	6. Challenge (re-iterate)	6. Re-iterate	3. Re-iterate

Table 2.1 – A generic overview of the System Dynamics process: a comparison with some descriptions proposed by other authors

Research needs	Research methods	
	Qualitative	***Quantitative***
Validate the process underlying the methodology		
Assess whether the use of the SD model in certain ways, or other processes related with implementing the methodology being developed (e.g. metrics collection), were perceived practical and useful	• Regular group presentations to the project team about the research progress, followed by debate • Observation of management and staff behaviour changes, in supporting the research project • Individual informal interviews with managers	NA
Validate the distinctive benefits of the methodology		
Assess whether the analyses produced by the SD model were perceived valid and useful	• Observation of changes in management decision-making and behaviour, in using the SD results • Observation of continued management and staff behaviour in supporting the research project • Individual informal interviews with managers	NA
Validate the SD models		
Validate the calibration of the SD models (is it producing the right behaviours for the right reasons?)	• Individual informal interviews with different managers and staff, to discus the assumptions • Demo session with the SD models	• Statistical tests of "goodness-of-fit" • Data collection and analysis (metrics)
Validate the structure of the SD model (is the model incorporating the relevant factors?)	• Informal questionnaires and individual interviews with managers and staff, to identify relevant factors • Demo session with the SD model • Development of influence diagrams and discussion in a "Cognitive-mapping fashion"	NA
Validate the technical aspects of the methodology		
Validate the analytical links between the SD and PERT/CPM model	• Individual informal interviews with managers and staff about what was being represented in the two models	• Statistical tests of "goodness-of-fit"

Table 3.4 – Qualitative and quantitative methods used during the field work, related to the research needs

SD model	Model development process
Roberts (1978)	"Top down" approach to model conceptualisation, based on the identification of policies. The process comprises the following steps: 1. Identification of management policies 2. Identification of information used as input to policies 3. Modelling of the direct effects of policies 4. Modelling of managerial perceptions of the project status 5. Modelling of the indirect effects of policies 6. Modelling of other uncontrolled feedback effects The model simulates a R&D project.
Cooper (1980)	An iterative process with an initial prototype. Comprised the following phases: 1. Structural model design based on identification of physical processes 2. Specification of data requirements 3. Data collection and information gathering from interviews 4. Development, quantification and review of a prototype single-phase model 5. Development and refinement of full multi-phase model in three iterations 6. Model validation (mainly behaviour reproduction of real scenarios) This process required extensive data collection and close staff and management involvement. The model simulates a shipbuilding development programme.
Richardson and Pugh (1981)	An iterative process, comprising the following steps: 1. Problem identification through the dynamic definition of key reference modes of behaviour (patterns over-time): "problem free" and "disruption". Identification of key variables to represent the project state. Identification of likely relationships by comparing the two reference modes. 2. Definition of model boundaries: what to include and exclude, level of detail/aggregation. 3. Identification of feedback structure using Ids. a "top-down" approach based on the identification of the physical process. Specification of "dynamic hypothesis" used to "validate" feedback structure. 4. Conversion of the IDs into "level/rate" diagrams, through the identification of the levels, rates and auxiliaries in the IDs. 5. Model quantification. Model parameters and parameter estimating techniques are classified into categories. 6. Validation testing. As the model is progressively quantified, each iteration should finish with a testing phase based on sensitivity analysis in order to identify eventual needs for re-formulation. The model simulates a R&D project.
Abdel-Hamid and Madnick (1991)	A life-cycle approach comprising the following steps: 1. Identification of generic feedback project dynamics using IDs 2. Specification of model boundaries: project life-cycle phases, activities, managerial policies, and project size 3. Specification of model architecture: sub-systems of project activity and their interrelationships 4. Interviews with managers for information gathering about the sub-systems 5. Development of a prototype model 6. Literature review based on the prototype model to quantify relationships 7. Development of detailed model 8. Iterative revision of detailed model based on interviews with managers 9. Validation The model simulates a software development project.

Table 6.2 – Summary comparison of the model development process followed in past SD project models

SD model	Model development process
Lin and Levary (1989), Lin 1993)	"Top down" process, decomposing a software project into main functional sub-systems, based on a dual life-cycle view of engineering and management. No specific phases of model development are specified, but final validation is comprehensively addressed.
Williams et al (1995)	An iterative process using cognitive maps, focusing on a mix soft and hard modelling techniques. The process comprised the following stages: 1. Development of managers' individual cognitive maps 2. Development of a merged global cognitive "group map" 3. Conversion of cognitive "group map" into an influence diagram (ID) 4. Quantification of the influence diagram into a "level/rate" simulation model 5. Validation The relationships between the cognitive map, the ID and simulation model involves aggregation and decomposition of variables. The model simulates a development project of train wagons.
Ford (1995)	Top-down process of decomposing the project into sub-systems and further into sectors. New generic structures are developed and existing structures from other models are re-used. These structures are used as "building-blocks", which are then integrated into a final model. The following phases were followed: 1. Identification of the model sub-systems and their activities 2. The product development process is specified as a network of tasks 3. Development / re-use of generic structures for each sub-system 4. Integration of the generic structures into a generic model 5. Validation of generic model (behaviour reproduction and sensitivity testing) 6. Tailoring of the structures of the generic model to a specific project 7. Validation (behaviour reproduction and sensitivity testing) The model is generic and aimed at modelling any type of product development project.
PMMS (1993 – present)	Iterative process based on a "building-block" approach, comprising the following phases: 1. Design: model architecture, specifying the project decomposition 2. Programming: "building blocks" are assembled and basic calibration is performed 3. Refinement: detailed calibration for the specific project. Validation.

Table 6.2 (cont.) – Summary comparison of the model development process followed in past SD project models

SD model	Model structure
Roberts (1978)	No breakdown of project schedule and budget, and of work activities. The project is aggregated into a single task.
Cooper (1980)	Project breakdown into a life-cycle of work phases, according to the process of product development. Therefore, there is an explicit breakdown of schedule and budget. Each work phase is modelled by a "building block" containing a core generic structure called the "rework cycle". Building blocks include design, build, testing, and acquisition. Management control is implemented at the phase-level and at the project level (though not specified how). Management decisions include work scheduling, materials acquisition allocation of manpower among phases. The product development process is not split into separate entity-flows of work and defects; tasks cycle back if flawed. The model structure has never been published and so it is not available in the literature for analysis.
Richardson and Pugh (1981)	No breakdown of project schedule and budget, and of work activities. The project is aggregated into a single task.
Abdel-Hamid and Madnick (1991)	No breakdown of project schedule and budget. The project is aggregated into a single task. Within this task there is a breakdown of the software development process into three main entity-flows: work, errors and staff. There is also a breakdown of the development process into four main continuous development activities: development, QA, rework from QA, testing (includes rework).
Lin and Levary (1989), Lin 1993)	Project breakdown into a life-cycle of work phases, according to the classic life-cycle model of software development – unlike in Cooper (1980) phases are not implemented in parallel. There is an explicit breakdown of schedule and budget.
Williams et al (1995)	Project is decompose into two main phases of design and construction, with individual schedules and budgets. Each of these phases is modelled in great detail by specialised SD structures. The overall product development process is split into two main entity-flows of work and resources.
Ford (1995)	Project is decomposed into a network of tasks linked through dynamic dependencies. Each task has its own targets and so there is an explicit schedule and budget breakdown. Management takes place at both task and project level and include schedule, budget and quality control. Within each task the following activities are modelled: base-work, rework QA and co-ordination. Defects are modelled as flawed tasks in a co-flow which is linked to the flow of tasks. Resources are decomposed into categories, one per task and are controlled at the task level.
PMMS (1993 – present)	Like in Cooper (1980). The project is decomposed into a high level network of phases or activities with individual schedules and budgets. These phases/activities are linked through dependencies and can overlap. The rework-cycle structure at the core of each phase suggests all phases are only finished at the end of the project, hence representing more continuous activities than phases.

Table 6.3 – Summary comparison of the model structure in past SD project models

SD model	Model quantification
Roberts (1978)	Most relationships considered as generic and quantified based on empirical reasoning, and on general empirical knowledge gained from experiences with real past projects
Cooper (1980)	Quantification of "hard" relationships is based on extensive data collection. Empirical approach to quantify "soft" relationships, based on information gathering from managers and staff in a real project.
Richardson and Pugh (1981)	Suggests some validation principles. Equations should be subjected to dimensional consistency checking, and their structure should be supported by a rationale. Sources of information used were data collection, expert judgement, and literature. Based on this information, empirical reasoning is used to produce intelligent estimates. Suggests a classification framework for parameters and estimating techniques. Stresses the risks of statistical and of behaviour based parameter estimating techniques. Advocates the use of information from the real world, below the level of aggregation in the model.
Abdel-Hamid and Madnick (1991)	Quantification of all relationships supported by a rationale using information from interviews, data collection and from extensive literature review. "Soft" relationships and intangible parameters quantified based on empirical reasoning.
Lin and Levary (1989), Lin 1993)	Quantification of relationships is based on data available in the literature, structured questionnaires, and interviews with managers.
Williams et al (1995)	Quantification based on extensive metrics collection from the real project, and on information gathered from interviews and workshops with managers.
Ford (1995)	Quantification of all relationships supported by a rationale, using information from previous SD project models and form literature. Quantification of the model for the case-study project based on information from interviews and extensive data collection.
PMMS (1993 – present)	Quantification of relationships is based on numerous experiences with using the model to replicate real past projects.

Table 6.4 – Summary comparison of model quantification in past SD project models

SD model	Model validation
Roberts (1978)	Model's ability to reproduce typical general behaviour patterns of R&D projects. Model's ability to provide plausible explanations for fictitious scenarios. No requirement to reproduce a real project.
Cooper (1980)	Based on accurate reproduction of project history, assessed through statistical tests of "goodness-of-fit". The calibration for specific scenarios was validated based on the definition of valid ranges for parameter values, and on the definition of "axioms" of consistency among the values of various parameter. "Extreme condition" tests carried out with assessment of the plausibility of the behaviour produced. Structural alternatives were tested and compared.
Richardson and Pugh (1980)	Based on the use of confidence tests to assess both (i) purpose suitability and (ii) consistency with reality. In this specific model, there was no requirement to reproduce a real project.
Abdel-Hamid and Madnick (1991)	Based on the model's ability to reproduce the history behaviour of a real past project, and on its ability to reproduce known generic behaviour of software projects (e.g. Raleigh curve).
Lin and Levary (1989), Lin 1993)	Validation based on the model's ability of reproducing the behaviour of a real past project. Basic confidence tests used and other particular tests based on management expert judgement (e.g. Turing tests).
Williams et al (1995)	Validation primarily based on model's ability to reproduce the past behaviour of a real project. Empirical evidence used to support parameter calibration. Use of general confidence tests (Forrester and Senge 1980)
Ford (1995)	Based on the plausibility of the behaviour produced by the generic model and on its ability of explaining this behaviour based on its feedback structure. Sensitivity testing also carried out. The model was tailored for a specific project and had to reproduce its past behaviour. Validity ranges for the model parameters were specified based on data collection and on interviews.
PMMS (1993 – present)	Validation based on "parameter tuning", so that the model replicates the past behaviour of the real project. Managers' expert judgement used to support the specification of the high level "building-block" structure (i.e. phases and their interdependencies) .

Table 6.5 – Summary comparison of model validation in past SD project models

SD model	Model use
Roberts (1978)	"What-if" analysis for policy analysis and improvement, using fictitious scenarios.
Cooper (1980)	Post mortem diagnosis (i.e. "what-would-have-happened-if" analysis): real past behaviour vs. behaviour with no Client disruptive actions. Aimed at providing accurate estimates of costs for which Client actions were responsible.
Richardson and Pugh (1980)	"What-if" analysis for policy analysis and improvement, using fictitious scenarios.
Abdel-Hamid and Madnick (1991)	Post mortem analysis to diagnose past behaviour and uncover information about the project past performance. As a generic model of software projects, "what-if" analysis was carried out to investigate the performance and impact of generic control policies on the software development process.
Lin and Levary (1989), Lin 1993)	A generic model of software projects aimed at supporting the management of an on-going software project. Model adjustment for a specific project based on parameter calibration during the requirements phase.
Williams et al (1995)	Post mortem diagnosis (i.e. "what-would-have-happened-if" analysis): real past behaviour vs. behaviour with no Client disruptive actions. Aimed at providing accurate estimates of costs for which Client actions were responsible.
Ford (1995)	Post mortem and retrospective policy analysis and improvement.
PMMS (1993 – present)	Claims that range of uses covers past, on-going, and prospective projects. Based on calibration for a reference mode of behaviour for the project ("problem free" or "disrupted", followed by "what-if" analysis (risks, Client actions, policies, uncertainties).

Table 6.6 – Summary comparison of model use in past SD project models

Management Decisions	Systems	High	Integratio	System	Management	Management	System	Project	Project
Work	—	—	—	—	—	—	—	—	—
Scope increases	T	T	T	T	T	T			ED/EX
Scope changes	T	T	T	T	T	T			ED/EX
Scope reductions	T	T	T	T	T	T			ED/EX
Time	—	—	—	—	—	—	—	—	—
Schedule extension	ED/T	ED/T	ED/T	ED/T	ED/T	ED/T			ED
Schedule compression	ED/T	ED/T	ED/T	ED/T	ED/T	ED/T			ED
Effort	—	—	—	—	—	—	—	—	—
Prioritisation to engineering	ED/T	ED/T	ED/T	ED/T	ED	ED			
Allocation to engineering	ED	ED	ED	ED					
Budget increases	T	T	T	T	ED/T	ED/T			ED
Budget reductions	T	T	T	T	ED/T	ED/T			ED
Over-time requests	ED/T	ED/T	ED/T	ED/T	ED	ED			
Resources	—	—	—	—	—	—	—	—	—
Hire/fire/transfer								T	ED
Increase current availability	T	T	T	T	ED/T	ED/T	ED/T		ED
Reduce current availability	T	T	T	T	ED/T	ED/T	ED/T		ED
Adjustment of future planned	T	T	T	T	ED/T	ED/T	ED/T		ED/EX
Processes	—	—	—	—	—	—	—	—	—
Engineering	—	—	—	—	—	—	—	—	—
Change inter-task concurrency	T	T	T	T	ED/T	ED/T			ED/EX
Change intra-task concurrency	ED/T	ED/T	ED/T	ED/T	ED/T	ED/T			
Eliminate tasks					T	T			ED/EX
Introduce tasks					T	T			ED/EX
Eliminate dependencies	T	T	T	T	T	T			ED/EX
Change process timings	ED	ED	ED	ED	ED	ED			
Management	—	—	—	—	—	—	—	—	—
Change monitoring delays	T	T	T	T	ED/T	ED/T			ED/EX
Change control policies	ED/T	ED/T	ED/T	ED/T	ED/T	ED/T			ED/EX

Table 6.11 – Example of a matrix mapping elementary decisions against the tasks in the model architecture

Types of Feedback Loops	Engineering Tasks									
	System Requirements	HL Design	Detail Des Comp A	Detail Des Comp B	Coding Comp A	Coding Comp B	Testing Comp A	Testing Comp B	Integration	System Testing
E1 – progress ↔ work rate										
Intra-task work availability	✓	✓	✓	✓					✓	
Error re-generation		✓	✓	✓	✓	✓			✓	
Passive errors effect					✓	✓	✓	✓	✓	✓
Work complexity			✓	✓		✓	✓	✓	✓	✓
E2 – progress ↔ staff										
Productivity learning curve	✓	✓	✓	✓	✓	✓			✓	
Rework learning curve			✓	✓	✓	✓	✓	✓	✓	✓
QA learning curve		✓	✓	✓	✓	✓			✓	
QA cuts			✓	✓	✓				✓	
Non-reported over-time			✓		✓		✓		✓	✓
E3 – staff ↔ staff status										
Staff exhaustion from over-time			✓	✓	✓	✓	✓	✓	✓	✓
Training accomplishment	✓	✓	✓		✓		✓		✓	✓

Table 6.12 – Example of an engineering matrix mapping intra-task E-type of feedback loops into the engineering tasks of the model architecture

Validation stage	Direct structure tests	Structure oriented behaviour tests	Behaviour accuracy tests
1. During the implementation steps of a super-structure	*Advised:* • All	*Advised:* • Partial model testing	NA.
2. At the end of each implementation step of a super-structure	NA.	*Advised:* • Qualitative features – tolerant • Extreme condition – tolerant • Boundary adequacy – new added structures *Useful:* • Behaviour sensitivity – tolerant • Phase relationship – tolerant	NA.
3. At the end of the full development of a super-structure	NA.	*Advised:* • Qualitative features – strict • Extreme condition – strict • Behaviour sensitivity – strict • Phase relationship – strict Useful: • Boundary adequacy – new added structures	NA.
4. At the end of the tailoring and calibration of a super-structure to a project task	*Advised:* • Parameter confirmation.	*Advised:* • Partial model testing. *Useful:* • Turing test – if historical scenarios available	*Advised:* • Reproduction of historical behaviour • Reproduction of key reference models of behaviour
5. During the integration of project tasks	*Advised:* • All – focused on the equation and parameters of the links among tasks.	NA.	NA.
6. At the end of integrating project tasks into sub-project-networks	NA.	*Advised:* • Qualitative features – tolerant • Extreme condition – strict • Behaviour sensitivity – strict • Phase relationship – across tasks. Strict.	NA.
7. After the full model has been integrated	NA.	NA.	*Advised:* • Reproduction of historical behaviour • Reproduction of key reference modes of behaviour

Table 6.16 – Validation framework – proposed set of confidence tests to be implemented in each stage of the implementation process

PERT/CPM Tasks	SD-Tasks									
	Engineering			*Management*			*Human Resource Management*			
Management	Requirements Specification	...	System Testing	Project Control	...	Testing Control	Project HRM	...	Integration and Test HRM	
Develop Reqs Plan	✓	...			...			...		
Review Reqs Status	✓	...			...			...		
Develop testing plan		...			...	✓		...		
Conduct Quality Audit		...		✓	...			...		
...	...	...	...	...	...	...	...	...	...	
Conduct Interviews		...			...		✓	...		
Interview testers		...			...			...	✓	
Engineering	—	—	—	Not applicable						
Requirements stg 1	✓	...								
Requirements stg 2	✓	...								
...		...								
Interface Whitebox Tests		...	✓							
Interface Blackbox Tests		...	✓							
...	...	...	...							

Table 7.3 – Matrix Specifying Work Breakdown Structural Correspondence Links

Purpose	Data-item	PERT to SD	SD to PERT	Data Consistency
Transfer Initial Plan				
	Profiles of Project Resources Availability	DEI-1.PERT-SD	DEI-1.SD-PERT	DCI-1
	Project Start Date	DEI-2.PERT-SD	DEI-2.SD-PERT	DCI-2
	Start and Finish Dates of SD-Tasks	DEOI-1.PERT-SD	NA	DCOI-1.PERT-SD
	Budget of SD-Tasks	DEOI-2.PERT-SD	NA	DCOI-2.PERT-SD
	Scope of SD-Tasks	DEOI-3.PERT-SD	NA	DCOI-3.PERT-SD
	Profiles of Resource Allocation to SD-Task	DEOI-4.PERT-SD	NA	DCOI-4.PERT-SD
	Budget Breakdown of SD-Tasks	*DEOI-5.PERT-SD*	*NA*	*DCOI-5.PERT-SD*
Transfer Project Status (Past Segment)				
	Present Level of Project Resources Availability	DEI-3.2.PERT-SD	NA	DCOI-6.2.SD-PERT
	Profiles of Project Resources Availability	DEI-3.3.PERT-SD	DEOI-10.2.SD-PERT	DCOI-6.2.SD-PERT
	Present Planned Start / Finish Dates of SD-Tasks	DEOI-6.PERT-SD	NA	DCO-1.2
	Present Planned CAC of SD-Tasks	DEOI-7.PERT-SD	NA	DCO-2.2
	Present Planned SCAC of SD-Tasks	DEOI-8.PERT-SD	NA	DCO-4.2
	Present Level of Resource Allocation to SD-Task	DEOI-9.2.PERT-SD	NA	DCO-3.2
	Profiles of Resource Allocation to SD-Task	DEOI-9.3.PERT-SD	NA	DCO-3.2
Transfer New Current Plan (Future Segment)				
	Present Level of Project Resources Availability	DEI-3.2.PERT-SD	NA	DCOI-6.1.SD-PERT
	Profiles of Project Resources Availability	DEI-3.1.PERT-SD	DEOI-10.1.SD-PERT	DCOI-6.1.SD-PERT
	Present Planned Start / Finish Dates of SD-Tasks	DEOI-6.PERT-SD	NA	DCO-1.1
	Present Planned CAC of SD-Tasks	DEOI-7.PERT-SD	NA	DCO-2.1
	Present Planned SCAC of SD-Tasks	DEOI-8.PERT-SD	NA	DCO-4.1
	Present Level of Resource Allocation to SD-Tasks	DEOI-9.2.PERT-SD	NA	DCO-3.1
	Profiles of Resource Allocation to SD-Task	DEOI-9.1.PERT-SD	NA	DCO-3.1

Table 7.5 – Summary overview of the data-links of SYDPIM basic-mode

Analytical Link	Description
Structural links	
Correspondence	
SC-WBS	Mapping of PERT/CPM tasks to SD-Tasks
SC-OBS	Mapping of PERT/CPM resources to SD-Resources
SC-WD	Mapping of PERT/CPM dependencies to SD-Dependencies
Consistency	
SCN-RA	Check consistency of SC-WBS with SC-OBS regarding resource allocation
SCN-WD	Check consistency of SC-WBS with SC-WD
Data Links	
Exchange	
DEI-1.SD-PERT	Transfer profiles of project resources availability in initial plan
DEI-2.SD-PERT	Transfer project start date in initial plan
DEOI-10.1.SD-PERT	Transfer profiles of project resources availability in future segment of current plan
DEOI-10.2.SD-PERT	Transfer profiles of project resources availability in past segment of current plan
DEI-1.PERT-SD	Transfer profiles of project resources availability in initial plan
DEI-2.PERT-SD	Transfer project start date in initial plan
DEI-3.1.PERT-SD	Transfer profiles of project resources availability in future segment of current plan
DEI-3.2.PERT-SD	Transfer present level of project resources availability in future segment of current plan
DEI-3.3.PERT-SD	Transfer profiles of project resources availability in past segment of current plan
DEOI-1.PERT-SD	Transfer start and finish dates of SD-Tasks in initial plan
DEOI-2.PERT-SD	Transfer budget of SD-Tasks in initial plan
DEOI-3.PERT-SD	Transfer scope of SD-Tasks in initial plan
DEOI-4.PERT-SD	Transfer profiles of resource allocation to SD-Tasks in initial plan
DEOI-5.PERT-SD	Transfer budget breakdown of SD-Tasks in initial plan
DEOI-6.PERT-SD	Transfer present planned start / finish dates of SD-Tasks
DEOI-7.PERT-SD	Transfer present CAC of SD-Tasks
DEOI-8.PERT-SD	Transfer present SCAC of SD-Tasks
DEOI-9.1.PERT-SD	Transfer profiles of resource allocation to SD-Tasks in future segment of current plan
DEOI-9.2.PERT-SD	Transfer present level of resource allocation to SD-Tasks
DEOI-9.3.PERT-SD	Transfer profiles of resource allocation to SD-Tasks in past segment of current plan
Consistency	
DCI-1	Check profiles of project resources availability in initial plan
DCI-2	Check project start date in initial plan
DCOI-1.PERT-SD	Check start and finish dates of SD-Tasks in initial plan
DCOI-2.PERT-SD	Check budget of SD-Tasks in initial plan
DCOI-3.PERT-SD	Check scope of SD-Tasks in initial plan
DCOI-4.PERT-SD	Check profiles of resource allocation to SD-Tasks in initial plan
DCOI-5.PERT-SD	Check budget breakdown of SD-Tasks in initial plan
DCOI-6.1.SD-PERT	Check profiles of project resources availability in future segment of current plan
DCOI-6.2.SD-PERT	Check profiles of project resources availability in past segment of current plan
DCO-1.1	Check planned start / finish dates of SD-Tasks in future segment of current plan
DCO-1.2	Check planned start / finish dates of SD-Tasks in past segment of current plan
DCO-2.1	Check CAC of SD-Tasks in future segment of current plan
DCO-2.2	Check CAC of SD-Tasks in past segment of current plan
DCO-3.1	Check profiles of resource allocation to SD-Tasks in future segment of current plan
DCO-3.2	Check profiles of resource allocation to SD-Tasks in past segment of current plan
DCO-4.1	Check SCAC of SD-Tasks in future segment of current plan
DCO-4.2	Check SCAC of SD-Tasks in past segment of current plan
Data-Structural Links	
Readjustment	
DSR-1	Generate shape of intra-task dependencies of SD-Tasks in future segment of current plan
DSR-2	Generate shape of inter-task dependencies of SD-Tasks in future segment of current plan
Consistency	
DSC-1	Check shape of intra-task dependencies of SD-Tasks in future segment of current plan
DSC-2	Check shape of inter-task dependencies of SD-Tasks in future segment of current plan

Table 7.6 – Summary of the analytical links of SYDPIM basic mode

Algorithm	Description
Structural Links	
SC-WD-1	Map PERT/CPM dependencies to SD intra-task dependencies
SC-WD-2	Map PERT/CPM dependencies to SD inter-task dependencies
SCN-RA-1	Consistency-check work and organisation breakdown links for resource allocation
SCN-WD-1	Consistency-check intra-task work dependency links with work breakdown links
SCN-WD-2	Consistency-check inter-task work dependency links with work breakdown links
Data-structural links	
DSR-1-1	Derivation of progress curve for SD intra-task dependency
DSR-2-1	Derivation of progress curve for SD inter-task dependency

Table 7.7 – Summary of automated algorithms of SYDPIM basic mode

Pattern	Calculation	Requirements
Schedule		
SAC[t]	For the whole project and per SD-Task. Constant value – steady behaviour. **SD-Task**: = MAX planned finishing date of all tasks in the PERT/CPM current plan mapped to the SD-Task. **Project**: = Max SAC[t] of all SD-Tasks.	Current PERT/CPM plan SC-WBS link
Start Date[t]	For each SD-Task. Constant value – steady behaviour. **SD-Task:** = MIN planned start date of all tasks in the PERT/CPM current plan mapped to the SD-Task.	Current PERT/CPM plan SC-WBS link

Table 7.8 – Calculation and requirements of the schedule patterns of future behaviour extracted from the PERT/CPM model.

Pattern	Calculation	Requirements
Effort		
ACWP[t]	For the whole project and per SD-Task. Varies over-time until currently planned completion date is reached. Can be split into engineering and management type of effort. **SD-Task**: = ACWP[present] + SUM of *cumulative over-time effort allocated to all tasks* in the future segment of the PERT/CPM current plan mapped to the SD-Task. **Project**: = SUM ACWP[t] of all SD-Tasks.	Current PERT/CPM plan SC-WBS link
BCWP[t]	For the whole project and per SD-Task. Varies over-time until currently planned completion date is reached. **SD-Task**: = BCWP[present] + SUM of initially planned budget times the planned % scope accomplished over-time, for all tasks in the initial PERT/CPM plan which are present in the future segment of the current PERT/CPM plan and are mapped to the SD-Task. **Project**: = SUM BCWP[t] of all SD-Tasks.	Current PERT/CPM plan Initial PERT/CPM plan SC-WBS link SC-WBS link (initial)
BCWS[t]	For the whole project and per SD-Task. Varies over-time until initially planned completion date is reached. **SD-Task**: = BCWS[present] + SUM of planned budget allocation over-time for all tasks in the initial PERT/CPM plan which are mapped to the SD-Task and fall in the future segment of the project. **Project**: = SUM BCWS[t] of all SD-Tasks.	Initial PERT/CPM plan SC-WBS link (initial)
CTC[t]	For the whole project and per SD-Task. Varies over-time until currently planned completion date is reached. **SD-Task**: = CAC[t] - ACWP[t] **Project**: = SUM CTC[t] of all SD-Tasks.	Current PERT/CPM plan SC-WBS link
CAC[t]	For the whole project and per SD-Task. Constant value – steady behaviour. **SD-Task**: = ACWP[SAC[t]] **Project**: = SUM CAC[t] of all SD-Tasks.	Current PERT/CPM plan SC-WBS link

Table 7.9 – Calculation and requirements of the effort patterns of future behaviour extracted from the PERT/CPM model.

Pattern	Calculation	Requirements
Scope		
SCAC[t]	For the whole project and per SD-Task. Constant value – steady behaviour. **SD-Task**: = ASCWP[SAC[t]] **Project**: = SUM SCAC[t] of all SD-Tasks.	Current PERT/CPM plan SC-WBS link
CSCC[t]	For the whole project and per SD-Task. Constant value – steady behaviour. **SD-Task**: = 0 **Project**: = SUM CSCC[t] of all SD-Tasks.	–
ASCWP[t]	For the whole project and per SD-Task. Varies over-time until currently planned completion date is reached. **SD-Task**: = ASCWP[present] + SUM of cumulative over-time scope planned to be spent for all tasks in the future segment of the PERT/CPM current plan mapped to the SD-Task **Project**: = SUM ASCWP[t] of all SD-Tasks.	Current PERT/CPM plan SC-WBS link
SCTC[t]	For the whole project and per SD-Task. Varies over-time until currently planned completion date is reached. **SD-Task**: = SCAC[t] – ASCWP[t] **Project**: = SUM SCTC[t] of all SD-Tasks.	Current PERT/CPM plan SC-WBS link

Table 7.10 – Calculation and requirements of the scope patterns of future behaviour extracted from the PERT/CPM model.

Pattern	Calculation	Requirements
Resources		
ASP[t]	Per SD-Resource, for the whole project and per SD-Task. Varies over-time until currently planned completion date is reached. **SD-Task**: = SUMT of planned allocation profiles of all resources mapped to the SD-Resource, allocated to all tasks in the future segment of the current PERT/CPM plan mapped to the SD-Task. **Project**: same as actual availability profiles of all resources in the current PERT/CPM plan	Current PERT/CPM plan SC-WBS link SC-OBS link
PSP[t]	Per SD-Resource, for the whole project and per SD-Task. Varies over-time until initially planned completion date is reached. **SD-Task**: = SUMT of planned allocation profiles of all resources mapped to the SD-Resource allocated to all tasks in the initial PERT/CPM plan which are mapped to the SD-Task and fall in the future segment of the project **Project**: same as planned availability profiles of all resources in the initial PERT/CPM plan	Initial PERT/CPM plan SC-WBS link (initial) SC-OBS link (initial)
CASP[t]	Per SD-Resource, for the whole project and per SD-Task. Varies over-time until currently planned completion date is reached. **SD-Task**: = CASP[present] + CUMULATIVE(ASP[t]) **Project**: CASP[present] + CUMULATIVE(ASP[t])	Current PERT/CPM plan SC-WBS link SC-OBS link
CPSP[t]	Per SD-Resource, for the whole project and per SD-Task. Varies over-time until initially planned completion date is finished. **SD-Task**: = CPSP[present] + CUMULATIVE(PSP[t]) **Project**: = CPSP[present] + CUMULATIVE(PSP[t])	Initial PERT/CPM plan SC-WBS link (initial) SC-OBS link (initial)
CSPAC[t]	Per SD-Resource, for the whole project and per SD-Task. Constant value – steady behaviour. **SD-Task**: = CASP[SAC[t]] **Project**: = CASP[SAC[t]]	Current PERT/CPM plan SC-WBS link SC-OBS link

Table 7.11 – Calculation and requirements of the resource patterns of future behaviour extracted from the PERT/CPM model.

Metric	Object variable
Update of SYMDB ***PERT/CPM derived metrics***	
(1) Project	
(1.1) Resources	
ASP	SYMDB.PERT/CPM derived metrics.Resources.Project Resources.ASP[present]
...	...
CSPAC	SYMDB.PERT/CPM derived metrics.Resources.Project Resources.CSPAC[present]
(2) SD-Tasks	
(2.1) Schedule	
Start date	SYMDB.PERT/CPM derived metrics.Schedule.Task Schedule.Start date[i][present]
Finish date	SYMDB.PERT/CPM derived metrics.Schedule.Task Schedule.Finish date[i][present]
(2.2) Effort	
ACWP	SYMDB.PERT/CPM derived metrics.Effort.Task Effort.ACWP[i][present]
...	...
CAC	SYMDB.PERT/CPM derived metrics.Effort.Task Effort.CAC[i][present]
(2.3) Scope	
SCAC	SYMDB.PERT/CPM derived metrics.Scope.Task Scope.SCAC[i][present]
...	...
SCTC	SYMDB.PERT/CPM derived metrics.Scope.Task Scope.SCTC[i][present]
(2.4) Resources	
ASP	SYMDB.PERT/CPM derived metrics.Resources.Task Resources.ASP[i][present]
...	...
CSPAC	SYMDB.PERT/CPM derived metrics.Resources.Task Resources.CSPAC[i][present]

Table 7.17 – Object variables of SYMDB to be updated in SYDPIM activity (M2a) with PERT/CPM derived metrics

Metric	Calculation	Requirements
Update of SYMDB ***PERT/CPM derived metrics***		
(1) Project		
(1.1) Resources (availability)	*These SYMDB data-points are calculated for each SD-Resources* *(*) this present data needs to be updated again at the end of re-planning*	
ASP[present]	Actual project availability profiles of SD-Resources, in the current PERT/CPM plan	Past segment of current PERT/CPM plan SC-OBS link
PSP[present]	Planned project availability profiles of all resources, in the initial PERT/CPM plan	Initial PERT/CPM plan SC-OBS link (initial)
CASP[present]	Total cumulative actual project availability of all resources, in the past segment of the current PERT/CPM plan	Past segment of current PERT/CPM plan SC-OBS link
CPSP[present]	Total cumulative planned project availability up to present, of all resources in the initial PERT/CPM plan	Initial PERT/CPM plan SC-OBS link (initial)
CSPAC[present] (*)	CASP[present] + total cumulative planned project availability of all resources in the future segment of the current PERT/CPM plan	Past segment of current PERT/CPM plan Future segment of current PERT/CPM plan SC-OBS link

Table 7.18 – Calculations, PERT/CPM plans and analytical links required to update the object variables of SYMDB in activity (M2a), with PERT/CPM derived metrics (project resources availability)

Metric	Calculation	Requirements
Update of SYMDB ***PERT/CPM derived metrics***		
(2) SD-Tasks		
(2.1) Schedule	*These SYMDB data-points are calculated for each SD-Task* *(*) if in the future segment, this present data needs to be updated again at the end of re-planning*	
Start date[present] (*)	MIN planned start date of all tasks in the PERT/CPM current plan mapped to SD-Task	Past segment of current PERT/CPM plan If SD-Task not started, future segment of current PERT/CPM plan SC-WBS link
Finish date[present] (*)	MAX planned finishing date of all tasks in the PERT/CPM current plan mapped to the SD-Task	Past segment of current PERT/CPM plan If SD-Task not finished, future segment of current PERT/CPM plan SC-WBS link

Table 7.19 – Calculations, PERT/CPM plans and analytical links required to update the object variables in activity (M2a), with PERT/CPM derived metrics (SD-Tasks schedules)

Metric	Calculation	Requirements
Update of SYMDB ***PERT/CPM derived metrics***		
(2) SD-Tasks		
(2.2) Effort	*These SYMDB data-points are calculated for each SD-Task* *(*) if SD-Task not finished this present data needs to be updated again at the end of re-planning*	
ACWP[present]	SUM of effort allocated to all tasks in the past segment of the PERT/CPM current plan, mapped to the SD-Task	Past Segment of current PERT/CPM plan SC-WBS link
ACWP breakdowns	Same as above, with the appropriate breakdown	Past Segment of current PERT/CPM plan SC-WBS link
BCWP[present]	SUM of initially planned budget (from initial PERT/CPM plan) times the actual % scope accomplished over-time (in current PERT/CPM plan), for all tasks in the initial PERT/CPM plan which are present in the past segment of the current PERT/CPM plan, and are mapped to the SD-Task	Initial PERT/CPM plan Past Segment of current PERT/CPM plan SC-WBS link SC-WBS link (initial)
BCWS[present]	SUM of initially planned budget allocation of all tasks in the initial PERT/CPM plan, which are mapped to the SD-Task and fall in the past segment of the project	Initial PERT/CPM plan SC-WBS link (initial)
CTC[present] (*)	CAC[present] – ACWP[present]	Past Segment of current PERT/CPM plan If SD-Task not finished yet, future segment of current PERT/CPM plan SC-WBS link
CAC[present] (*)	ACWP[present] + SUM of effort planned to be allocated to the PERT/CPM tasks in the future segment of the current PERT/CPM plan, which are mapped to the SD-Task	Past Segment of current PERT/CPM plan If SD-Task not finished yet, future segment of current PERT/CPM plan SC-WBS link

Table 7.20 – Calculations, PERT/CPM plans and analytical links required to update the object variables of SYMDB in activity (M2a), with PERT/CPM derived metrics (SD-Tasks budgets/costs)

Metric	Calculation	Requirements
Update of SYMDB ***PERT/CPM derived metrics***		
(2) SD-Tasks		
(2.3) Scope	*These SYMDB data-points are calculated for each SD-Task* *(*) if SD-Task not finished this present data needs to be updated again at the end of re-planning*	
SCAC[present] (*)	ASCWP[present] + SUM of scope planned to be allocated to the PERT/CPM tasks in the future segment of the current PERT/CPM plan, which are mapped to the SD-Task	Past segment of current PERT/CPM plan If SD-Task not finished yet, future segment of current PERT/CPM plan SC-WBS link
CSCC[present]	SUM of actual scope accomplished of all completed tasks in the past segment of the current PERT/CPM plan mapped to the SD-Task, minus the SUM of initially planned scope of all tasks in the initial PERT/CPM plan, which fall in the past segment of the project and are mapped to the SD-Task	Past segment of current PERT/CPM plan Initial PERT/CPM plan SC-WBS link SC-WBS link (initial)
ASCWP[present]	SUM of scope accomplished in all tasks in the past segment of the PERT/CPM current plan, mapped to the SD-Task	Past segment of current PERT/CPM plan SC-WBS link
SCTC[present] (*)	CAC[present] – ACWP[present]	Past segment of current PERT/CPM plan If SD-Task not finished yet, future Segment of current PERT/CPM plan SC-WBS link

Table 7.21 – Calculations, PERT/CPM plans and analytical links required to update the object variables of SYMDB in activity (M2a), with PERT/CPM derived metrics (SD-Tasks scope)

Metric	Calculation	Requirements
Update of SYMDB ***PERT/CPM derived metrics***		
(2) SD-Tasks		
(2.4) Resources	*These SYMDB data-points are calculated for each SD-Resource within each SD-Task* *(*) if SD-Task not finished this present data needs to be updated again at the end of re-planning*	
ASP[present]	SUM of actual present allocation levels of all resources mapped to the SD-Resource, which are allocated to all tasks in the past segment of the current PERT/CPM plan mapped to the SD-Task	Past segment of current PERT/CPM plan SC-WBS link SC-OBS link
PSP[present]	SUM of planned present allocation levels of all resources mapped to the SD-Resource, which are allocated to all tasks in the initial PERT/CPM plan mapped to the SD-Task	Initial PERT/CPM plan SC-WBS link (initial) SC-OBS link (initial)
CASP[present]	Total cumulative over-time allocation up to present of all resources mapped to the SD-Resource, which are allocated to all tasks in the past segment of the current PERT/CPM plan mapped to the SD-Task	Past segment of current PERT/CPM plan SC-WBS link SC-OBS link
CPSP[present]	Total cumulative over-time allocation up to present of all resources mapped to the SD-Resource, which allocated to all tasks in the initial PERT/CPM plan mapped to the SD-Task	Initial PERT/CPM plan SC-WBS link (initial) SC-OBS link (initial)
CSPAC[present] (*)	CASP[present] + total cumulative allocation of all resources mapped to the SD-Resource, planned to be allocated to tasks mapped to the SD-Task, in the future segment of the current PERT/CPM plan	Past segment of current PERT/CPM plan If SD-Task not finished yet, future segment of current PERT/CPM plan SC-WBS link SC-OBS link

Table 7.22 – Calculations, PERT/CPM plans and analytical links required to update the object variables of SYMDB in activity (M2a), with PERT/CPM derived metrics (SD-Tasks resource allocation)

Metric	Object variable	Calculation
Update of SYMDB ***Calculated metrics***		
(1) Project		
(1.2) Schedule	*Could have been derived from PERT/CPM model*	
Finish date	SYMDB.PERT/CPM derived metrics.Schedule.Project Schedule.Finish Date[present]	Latest finishing date of SD-Task
(1.3) Effort	*Could have been derived from PERT/CPM model*	
ACWP	SYMDB.PERT/CPM derived metrics.Effort.Project Effort.ACWP[present]	Sum of SD-Tasks ACWP
...	...	Sum of SD-Tasks ...
CAC	SYMDB.PERT/CPM derived metrics.Effort.Project Effort.CAC[present]	Sum of SD-Tasks CAC
(1.4) Scope	*Could have been derived from PERT/CPM model*	
SCAC	SYMDB.PERT/CPM derived metrics.Scope.Project Scope.SCAC[present]	Sum of SD-Tasks SCAC
...	...	Sum of SD-Tasks ...
SCTC	SYMDB.PERT/CPM derived metrics.Scope.Project Scope.SCTC[present]	Sum of SD-Tasks SCTC
(1.5) Quality	*Could have been derived from collected metrics*	
Defects detected	SYMDB.Collected metrics.Quality.Project Quality.Defects detected[present]	Sum of SD-Tasks defects detected
...	...	Sum of SD-Tasks defects
Defects awaiting rework	SYMDB.Collected metrics.Quality.Project Quality.Defects awaiting rework[present]	Sum of SD-Tasks defects awaiting rework

Table 7.23 – Object variables of SYMDB (project-wide results) to be updated in SYDPIM activity (M2a) from calculations involving other metrics already updated in the database

Metric	Object variable	Calculation
Update of SYMDB ***Calculated metrics***		
(2) Performance indices		
(2.1) Project indices	*These indices refer to the whole project performance*	
Earned value (EV)	SYMDB.Calculated metrics.Performance indices.Project Indices.EV[present]	CAC[t0] – CTC[present]
CPI	SYMDB.Calculated metrics.Performance indices.Project Indices.CPI[present]	BCWP / ACWP
SPI	SYMDB.Calculated metrics.Performance indices.Project Indices.SPI[present]	BCWP / BCWS
Cost variance (CV)	SYMDB.Calculated metrics.Performance indices.Project Indices.CV[present]	ACWP – BCWP
Schedule variance (SV)	SYMDB.Calculated metrics.Performance indices.Project Indices.SV[present]	BCWS – BCWP
Accounting variance (AV)	SYMDB.Calculated metrics.Performance indices.Project Indices.AV[present]	ACWP – BCWS
Time variance (TV)	SYMDB.Calculated metrics.Performance indices.Project Indices.TV[present]	Tpresent – Tk Tk : BCWS[Tk] = BCWP[Tpresent]
(2.2) Task indices	*These indices are calculated for each SD-Task*	
Earned value (EV)	SYMDB.Calculated metrics.Performance indices.Task Indices.EV[i][present]	CAC[t0] – CTC[present]
...	...	...
Time variance (TV)	SYMDB.Calculated metrics.Performance indices.Task Indices.TV[i][present]	Tpresent – Tk Tk : BCWS[Tk] = BCWP[Tpresent]

Table 7.24 – Object variables of SYMDB (performance indices) to be updated in SYDPIM activity (M2a) from calculations involving other metrics already updated in the database

Metric	Object variable	Calculation
Update of SYMDB ***Calculated metrics***		
(3) Process metrics		
(3.1) Productivity	*Productivity is measured in Scope / Effort (e.g. Tasks / Person-Month)*	
(3.1.1) Project	*These process metrics refer to the whole project*	
Gross	SYMDB.Calculated metrics.Process metrics.Productivity.Project productivity.Gross[present]	ASCWP / ACWP
Net	SYMDB.Calculated metrics.Process metrics.Productivity.Project productivity.Net[present]	ASCWP / ACWP-Eng-dev
(3.1.2) Task	*These process metrics are calculated for each SD-Task*	
Gross	SYMDB.Calculated metrics.Process metrics. Productivity.Task productivity.Gross[i][present]	ASCWP / ACWP
Net	SYMDB.Calculated metrics.Process metrics. Productivity.Task productivity.Net[i][present]	ASCWP / ACWP-Eng-dev
(3.2) Defect	*Defect indices are measure in Defect / Scope or Effort / Defect*	
(3.2.1) Project	*These defect metrics refer to the whole project*	
Detection index	SYMDB.Calculated metrics.Process metrics.Defect.Project defects.Detection index[present]	Defects detected / ASCWP
Cost to detect	SYMDB.Calculated metrics.Process metrics.Defect.Project defects.Cost to detect[present]	ASCWP-Eng-QA / Defects Detected
Cost to rework	SYMDB.Calculated metrics.Process metrics.Defect.Project defects.Cost to rework[present]	ASCWP-Eng-rework / Defects detected
(3.2.2) Task	*These defect metrics are calculated for each SD-Task*	
Detection index	SYMDB.Calculated metrics.Process metrics.Defect.Project defects.Detection index[i][present]	Defects detected / ASCWP
Cost to detect	SYMDB.Calculated metrics.Process metrics.Defect.Project defects.Cost to detect[i][present]	ASCWP-Eng-QA / Defects detected
Cost to rework	SYMDB.Calculated metrics.Process metrics.Defect.Project defects.Cost to rework[i][present]	ASCWP-Eng-rework / Defects reworked

Table 7.25 – Object variables of SYMDB (process metrics) to be updated in SYDPIM activity (M2a) from calculations involving other metrics already updated in the database

Pattern	Calculation	Requirements
Schedule		
SAC[t] (*)	For the whole project and per SD-Task. **SD-Task**: = for each time-point MAX planned finishing date of all tasks in the PERT/CPM plan of that time-point mapped to the SD-Task. **Project**: = for each data-point Max SAC[t] of all SD-Tasks.	Set of past versions of PERT/CPM plan (incl. Initial plan) Current PERT/CPM plan SC-WBS link SC-WBS link (past ver.)
Start Date[t] (*)	For each SD-Task. **SD-Task:** = for each time-point MIN planned start date of all tasks in the PERT/CPM plan of that time-point mapped to the SD-Task.	Set of past versions of PERT/CPM plan (incl. Initial plan) Current PERT/CPM plan SC-WBS link SC-WBS link (past ver.)

Table 7.26 – Calculation and requirements of the schedule patterns of past behaviour extracted from the PERT/CPM model.

Pattern	Calculation	Requirements
Effort		
ACWP[t]	For the whole project and per SD-Task. Can be split into engineering and management type of effort. **SD-Task**: = SUM of cumulative over-time effort allocated to all tasks in the past segment of the PERT/CPM current plan mapped to the SD-Task. **Project**: = SUM ACWP[t] of all SD-Tasks.	Current PERT/CPM plan SC-WBS link
BCWP[t]	For the whole project and per SD-Task. **SD-Task**: = SUM of initially planned budget times the planned % scope accomplished over-time, for all tasks in the initial PERT/CPM plan which are present in the past segment of the current PERT/CPM plan and are mapped to the SD-Task. **Project**: = SUM BCWP[t] of all SD-Tasks.	Current PERT/CPM plan Initial PERT/CPM plan SC-WBS link SC-WBS link (initial)
BCWS[t]	For the whole project and per SD-Task. **SD-Task**: = SUM of planned budget allocation over-time for all tasks in the initial PERT/CPM plan which are mapped to the SD-Task and fall in the past segment of the project. **Project**: = SUM BCWS[t] of all SD-Tasks.	Initial PERT/CPM plan SC-WBS link (initial)
CTC[t] (*)	For the whole project and per SD-Task. **SD-Task**: = CAC[t] – ACWP[t] **Project**: = SUM CTC[t] of all SD-Tasks.	Set of past versions of PERT/CPM plan (incl. Initial plan) Current PERT/CPM plan SC-WBS link SC-WBS link (past ver.)
CAC[t] (*)	For the whole project and per SD-Task. **SD-Task**: = for each time-point SUM of cumulative over-time effort allocated to all tasks in the PERT/CPM plan of that time-point mapped to the SD-Task. **Project**: = SUM CAC[t] of all SD-Tasks.	Set of past versions of PERT/CPM plan (incl. Initial plan) Current PERT/CPM plan SC-WBS link SC-WBS link (past ver.)

Table 7.27 – Calculation and requirements of the effort patterns of past behaviour extracted from the PERT/CPM model.

Pattern	Calculation	Requirements
Scope		
SCAC[t] (*)	For the whole project and per SD-Task. **SD-Task**: = for each time-point SUM of cumulative scope allocated to all tasks in the PERT/CPM plan of that time-point mapped to the SD-Task. **Project**: = SUM SCAC[t] of all SD-Tasks.	Set of past versions of PERT/CPM plan (incl. Initial plan) Current PERT/CPM plan SC-WBS link SC-WBS link (past ver.)
CSCC[t]	For the whole project and per SD-Task. Constant value – steady behaviour. **SD-Task**: = for each time-point SUM of actual scope accomplished of all completed tasks in the PERT/CPM plan of that time-point mapped to the SD-Task, minus the SUM of initially planned scope of all tasks in the initial PERT/CPM plan, scheduled to be completed at that time-point, mapped to the SD-Task **Project**: = SUM CSCC[t] of all SD-Tasks.	Set of past versions of PERT/CPM plan (incl. Initial plan) Current PERT/CPM plan SC-WBS link SC-WBS link (past ver.)
ASCWP[t]	For the whole project and per SD-Task. **SD-Task**: = SUM of cumulative actual scope accomplished in all tasks in the past segment of the PERT/CPM current plan mapped to the SD-Task **Project**: = SUM ASCWP[t] of all SD-Tasks.	Current PERT/CPM plan SC-WBS link
SCTC[t] (*)	For the whole project and per SD-Task. **SD-Task**: = SCAC[t] – ASCWP[t] **Project**: = SUM SCTC[t] of all SD-Tasks.	Set of past versions of PERT/CPM plan (incl. Initial plan) Current PERT/CPM plan SC-WBS link SC-WBS link (past ver.)

Table 7.28 – Calculation and requirements of the scope patterns of past behaviour extracted from the PERT/CPM model.

Pattern	Calculation	Requirements
Resources		
ASP[t]	Per SD-Resource, for the whole project and per SD-Task. **SD-Task**: = SUMT of actual allocation profiles of all resources mapped to the SD-Resource, allocated to all tasks in the past segment of the current PERT/CPM plan mapped to the SD-Task. **Project**: same as actual availability profiles of all resources in the current PERT/CPM plan	Current PERT/CPM plan SC-WBS link SC-OBS link
PSP[t]	Per SD-Resource, for the whole project and per SD-Task. **SD-Task**: = SUMT of planned allocation profiles of all resources mapped to the SD-Resource allocated to all tasks in the initial PERT/CPM plan which are mapped to the SD-Task and were scheduled to be completed at the present moment. **Project**: same as planned availability profiles of all resources in the initial PERT/CPM plan	Initial PERT/CPM plan SC-WBS link (initial) SC-OBS link (initial)
CASP[t]	Per SD-Resource, for the whole project and per SD-Task. **SD-Task**: = CUMULATIVE(ASP[t]) **Project**: = CUMULATIVE(ASP[t])	Current PERT/CPM plan SC-WBS link SC-OBS link
CPSP[t]	Per SD-Resource, for the whole project and per SD-Task. **SD-Task**: = CUMULATIVE(PSP[t]) **Project**: = CUMULATIVE(PSP[t])	Initial PERT/CPM plan SC-WBS link (initial) SC-OBS link (initial)
CSPAC[t] (*)	Per SD-Resource, for the whole project and per SD-Task. **SD-Task**: = for each time-point SUM of over-time cumulative resources allocated to all tasks in the PERT/CPM plan of that time-point mapped to the SD-Task **Project**: = for each time-point SUM of cumulative resources made available to the project in the PERT/CPM plan of that time-point	Set of past versions of PERT/CPM plan (incl. Initial plan) Current PERT/CPM plan SC-WBS link SC-WBS link (past ver.)

Table 7.29 – Calculation and requirements of the resource patterns of past behaviour extracted from the PERT/CPM model.

Appendix C – SYDPIM objects

C.1 – PERT/CPM model

Object: PERT/CPM model

PERT/CPM model =
- Initial date : date
- Current PERT/CPM plan =
  - Past segment : plan-segment
  - Future segment : plan-segment
- Initial PERT/CPM plan : plan-segment
- Past PERT/CPM plans : {plan-segment$_k$}
- PERT/CPM project behaviour =
  - Past segment : {behaviour-pattern$_k$}
  - Future segment : {behaviour-pattern$_k$}

Data-types:

plan-segment =
- Tasks : {task$_k$}
- Dependencies : {dependency$_k$}
- Resources : {resource$_k$}
- Plan_Resource-availability : {(SD-Resource$_k$,{(t_i,y)})}
- Actual_Resource-availability : {(SD-Resource$_k$,{(t_i,y)})}

dependency = (task$_i$, task$_j$)

behaviour-pattern = {(t_i, y)}

Task =
- planned start date : date
- actual start date : date
- planned completion date : date
- actual completion date : date
- planned duration : R+
- actual duration : R+
- planned budget : R+
- actual effort spent to date : R+
- planned scope : R+
- actual scope to date : R+
- planned resource profile : {(r_k, n)} or {(r_k,{(t_i, y)})}
- actual resource profile : {(r_k, n)} or {(r_k,{(t_i, y)})}
- task type : ENG | MAN

Formal specification of the SYDPIM object "PERT/CPM model"

C.2 – SD project model

Object: SD project model

SD project model =
- SD model architecture =
  - SD-Tasks : {SD-Task$_k$}
  - SD-Inter-Task-Dependencies : {SD-dependency$_k$}
  - SD-Resources : {SD-Resource$_k$}
- SD plan =
  - Initial plan : plan-initial
  - Past segment : plan-segment
  - Future segment : plan-segment
- Project conditions =
  - Past segment : {input-parameter$_k$}
  - Future segment : {input-parameter$_k$}
- Project behaviour =
  - Past segment : SD-Behaviour
  - Future segment : SD-Behaviour

Data-types:

Plan-initial =
- Schedule =
  - Project schedule =
    - Start Date : date
    - Finish Date : date
  - Task schedule =
    - Start date : {(SD-Task$_k$, date)}
    - Finish date : {(SD-Task$_k$, date)}
- Budget =
  - Project budget : R+
  - Task budget : {(SD-Task$_k$, R+)}
- Scope =
  - Project scope = : R+
  - Task scope : {(SD-Task$_k$, R+)}
- Resources =
  - Task Resources (*allocation*) : {(SD-Task$_k$, SD-Resource$_i$,{(t_l, y)})}
  - Project Resources (*availability*) : {(SD-Resource$_k$,{(t_l, y)})}

Plan-segment =
- Schedule =
  - Project schedule =
    - Finish date =
      - Decision-roles : {finish-date-adjustment-role$_i$}
      - Exogenous : {(t_l, date-adjustment)}
  - Task schedule =
    - Start date =
      - Decision-roles : {(SD-Task$_k$, {start-date-adjustment-role$_i$})}
      - Exogenous : {(SD-Task$_k$, {(t_l, date-adjustment)})}
    - Finish date =
      - Decision-roles : {(SD-Task$_k$, {finish-date-adjustment-role$_i$})}
      - Exogenous : {(SD-Task$_k$, {(t_l, date-adjustment)})}

Budget =
- Project budget =
  - Decision-roles : {budget-adjustment-role$_i$}
  - Exogenous : {(t_i,budget-adjustment)}
- Task budget =
  - Decision-roles : {(SD-Task$_k$, {budget-adjustment-role$_i$})}
  - Exogenous : {(SD-Task$_k$, {(t_i,budget-adjustment)})}

Scope =
- Project scope =
  - Decision-roles : {scope-adjustment-role$_i$}
  - Exogenous : {(t_i, scope-adjustment)}
- Task scope =
  - Decision-roles : {{SD-Task$_k$, {scope-adjustment-role$_i$}}
  - Exogenous : {SD-Task$_k$, {(t_i,scope-adjustment)}}

Resources =
- Task Resources (allocation) =
  - Decision-roles : {{SD-Task$_k$, {resource-adjustment-role$_k$}}
  - Exogenous : {SD-Task$_k$, SD-Resource$_i$, {(t_i, resource-adjust)}}
- Project Resources (availability) =
  - Decision-roles :{{SD-Resource$_k$, {resource-adjustment-role$_i$}}
  - Exogenous : {SD-Resource$_k$, {(t_i, resource-adjust)}}

SD-Behaviour =

Schedule =
- Project Schedule =
  - Finish date : {(t_i, date)}
- Tasks Schedule =
  - Start date : {(SD-Task$_k$, {(t_i, date)})}
  - Finish date : {(SD-Task$_k$, {(t_i, date)})}

Effort =
- Project Effort =
  - ACWP : {(t_i, R+)}
  - ACWP Engineering : {(t_i, R+)}
  - ACWP Management : {(t_i, R+)}
  - BCWP : {(t_i, R+)}
  - BCWS : {(t_i, R+)}
  - CTC : {(t_i, R+)}
  - CAC : {(t_i, R+)}
- Task Effort =
  - ACWP : {(SD-Task$_k$, {(t_i, R+)})}
  - ACWP Engineering : {(SD-Task$_k$, {(t_i, R+)})}
  - ACWP Management : {(SD-Task$_k$, {(t_i, R+)})}
  - BCWP : {(SD-Task$_k$, {(t_i, R+)})}
  - BCWS : {(SD-Task$_k$, {(t_i, R+)})}
  - CTC : {(SD-Task$_k$, {(t_i, R+)})}
  - CAC : {(SD-Task$_k$, {(t_i, R+)})}

Scope =
- Project Scope =
  - SCAC : {(t_i, R+)}
  - CSCC : {(t_i, R+)}
  - ASCWP : {(t_i, R+)}
  - SCTC : {(t_i, R+)}
- Task Scope =
  - SCAC : {(SD-Task$_k$, {(t_i, R+)})}
  - CSCC : {(SD-Task$_k$, {(t_i, R+)})}
  - ASCWP : {(SD-Task$_k$, {(t_i, R+)})}
  - SCTC : {(SD-Task$_k$, {(t_i, R+)})}

Resources =
  Project Resources (availability) =
    ASP : {(SD-Resource$_l$, {(t_i, R+)})}
    PSP : {(SD-Resource$_l$, {(t_i, R+)})}
    CASP : {(SD-Resource$_l$, {(t_i, R+)})}
    CPSP : {(SD-Resource$_l$, {(t_i, R+)})}
    CSPAC : {(SD-Resource$_l$, {(t_i, R+)})}
  Task Resources (allocation) =
    ASP : {(SD-Task$_k$, SD-Resource$_l$, {(t_i, R+)})}
    PSP : {(SD-Task$_k$, SD-Resource$_l$, {(t_i, R+)})}
    CASP : {(SD-Task$_k$, SD-Resource$_l$, {(t_i, R+)})}
    CPSP : {(SD-Task$_k$, SD-Resource$_l$, {(t_i, R+)})}
    CSPAC : {(SD-Task$_k$, SD-Resource$_l$, {(t_i, R+)})}
  Other Patterns : {behaviour-pattern$_k$}

SD-Task =
  Intra-task-curve : {(R+,R+)}
  Other elements : <*SD structure*>

SD-Dependency =
  Predecessor : SD-Task
  Successor : SD-Task
  Progress-curve : {(R+,R+)}
  Other elements : <*SD structure*>

SD-Resource = <*SD structure*>
start-date-adjustment-role$_k$ = <*SD structure*>
date-adjustment : *date*
finish-date-adjustment-role$_k$ = <*SD structure*>
budget-adjustment-role$_k$ = <*SD structure*>
budget-adjustment : R
scope-adjustment-role$_k$ = <*SD structure*>
scope-adjustment : R
resource-adjustment-role$_k$ = <*SD structure*>
resource-adjust : R
input-parameter = y | {(t_i, y)} | {(x, y)} | <*SD structure*>
behaviour-pattern = {(t_i, y)}

Formal specification of the SYDPIM object "SD model"

C.3 – SYDPIM metrics database (SYMDB)

Object: SYDPIM metrics database (SYMDB)

SYDPIM metrics database (SYMDB) =
- PERT/CPM derived metrics =
- Schedule =
  - Project Schedule =
    - Finish date : {(t_i, date)}
  - Task Schedule =
    - Start date : {(SD-Task$_k$, {(t_i, date)})}
    - Finish date : {(SD-Task$_k$, {(t_i, date)})}
- Effort =
  - Project Effort =
    - ACWP : {(t_i, R+)}
    - ACWP Engineering : {(t_i, R+)}
    - ACWP-Eng-QA : {(t_i, R+)}
    - ACWP-Eng-rework : {(t_i, R+)}
    - ACWP-Eng-dev : {(t_i, R+)}
    - ACWP Management : {(t_i, R+)}
    - ACWP-Man-HRM : {(t_i, R+)}
    - ACWP-Man-Control : {(t_i, R+)}
    - BCWP : {(t_i, R+)}
    - BCWS : {(t_i, R+)}
    - CTC : {(t_i, R+)}
    - CAC : {(t_i, R+)}
  - Task Effort =
    - ACWP : {(SD-Task$_k$, {(t_i, R+)})}
    - ACWP Engineering : {(SD-Task$_k$, {(t_i, R+)})}
    - ACWP-Eng-QA : {(SD-Task$_k$, {(t_i, R+)})}
    - ACWP-Eng-rework : {(SD-Task$_k$, {(t_i, R+)})}
    - ACWP-Eng-dev : {(SD-Task$_k$, {(t_i, R+)})}
    - ACWP Management : {(SD-Task$_k$, {(t_i, R+)})}
    - ACWP-Man-HRM : {(SD-Task$_k$, {(t_i, R+)})}
    - ACWP-Man-Control : {(SD-Task$_k$, {(t_i, R+)})}
    - BCWP : {(SD-Task$_k$, {(t_i, R+)})}
    - BCWS : {(SD-Task$_k$, {(t_i, R+)})}
    - CTC : {(SD-Task$_k$, {(t_i, R+)})}
    - CAC : {(SD-Task$_k$, {(t_i, R+)})}
- Scope =
  - Project Scope =
    - SCAC : {(t_i, R+)}
    - CSCC : {(t_i, R+)}
    - ASCWP : {(t_i, R+)}
    - SCTC : {(t_i, R+)}
  - Task Scope =
    - SCAC : {(SD-Task$_k$, {(t_i, R+)})}
    - CSCC : {(SD-Task$_k$, {(t_i, R+)})}
    - ASCWP : {(SD-Task$_k$, {(t_i, R+)})}
    - SCTC : {(SD-Task$_k$, {(t_i, R+)})}
- Resources =
  - Project Resources (availability) =
    - ASP : {(SD-Resource$_l$, {(t_i, R+)})}
    - PSP : {(SD-Resource$_l$, {(t_i, R+)})}
    - CASP : {(SD-Resource$_l$, {(t_i, R+)})}
    - CPSP : {(SD-Resource$_l$, {(t_i, R+)})}

CSPAC : {(SD-Resource$_l$, {(t_i, R+)})}
Task Resources (allocation) =
ASP : {(SD-Task$_k$, SD-Resource$_l$, {(t_i, R+)})}
PSP : {(SD-Task$_k$, SD-Resource$_l$, {(t_i, R+)})}
CASP : {(SD-Task$_k$, SD-Resource$_l$, {(t_i, R+)})}
CPSP : {(SD-Task$_k$, SD-Resource$_l$, {(t_i, R+)})}
CSPAC : {(SD-Task$_k$, SD-Resource$_l$, {(t_i, R+)})}

Collected metrics =
Quality =
Project Quality =
defects detected: {(t_i, R+)}
defects reworked : {(t_i, R+)}
cum defects detected : {(t_i, R+)}
cum defects reworked : {(t_i, R+)}
defects awaiting rework : {(t_i, R+)}
Task Quality =
defects detected: {SD-Task$_k$, (t_i, R+)}
defects reworked : {SD-Task$_k$, (t_i, R+)}
cum defects detected : {SD-Task$_k$, (t_i, R+)}
cum defects reworked : {SD-Task$_k$, (t_i, R+)}
defects awaiting rework : {SD-Task$_k$, (t_i, R+)}

Calculated metrics =
Performance indices =
Project indices =
EV : {(t_i, R+)}
CPI : {(t_i, R+)}
SPI : {(t_i, R+)}
CV : {(t_i, R+)}
SV : {(t_i, R+)}
AV : {(t_i, R+)}
TV : {(t_i, R+)}
Task indices =
EV : {SD-Task$_k$, (t_i, R+)}
CPI : {SD-Task$_k$, (t_i, R+)}
SPI : {SD-Task$_k$, (t_i, R+)}
CV : {SD-Task$_k$, (t_i, R+)}
SV : {SD-Task$_k$, (t_i, R+)}
AV : {SD-Task$_k$, (t_i, R+)}
TV : {SD-Task$_k$, (t_i, R+)}
Process metrics =
Productivity =
Project productivity =
Gross Productivity : {(t_i, R+)}
Net Productivity : {(t_i, R+)}
Task productivity =
Gross Productivity : {SD-Task$_k$, (t_i, R+)}
Net Productivity : {SD-Task$_k$, (t_i, R+)}
Defect =
Project Defects =
detection index : {(t_i, R+)}
cost to detect : {(t_i, R+)}
cost to rework : {(t_i, R+)}
Task Defects =
detection index : {SD-Task$_k$, (t_i, R+)}
cost to detect : {SD-Task$_k$, (t_i, R+)}
cost to rework : {SD-Task$_k$, (t_i, R+)}

Uncovered metrics (from SD diagnosis) =
- Defect =
  - Project defects =
    - Undetected : $\{(t_i, R+)\}$
    - cumulative generated : $\{(t_i, R+)\}$
    - generation rate : $\{(t_i, R+)\}$
    - undetected density : $\{(t_i, R+)\}$
    - cost to detect next : $\{(t_i, R+)\}$
  - Task defects =
    - Undetected : $\{(SD\text{-}Task_K, \{(t_i, R+)\})\}$
    - cumulative generated : $\{(SD\text{-}Task_K, \{(t_i, R+)\})\}$
    - generation rate : $\{(SD\text{-}Task_K, \{(t_i, R+)\})\}$
    - undetected density : $\{(SD\text{-}Task_K, \{(t_i, R+)\})\}$
    - cost to detect next : $\{(SD\text{-}Task_K, \{(t_i, R+)\})\}$
- Staff =
  - Project staff =
    - Fatigue : $\{(SD\text{-}Resource_K, \{(t_i, R)\})\}$
    - Experience : $\{(SD\text{-}Resource_K, \{(t_i, R)\})\}$
  - Task staff =
    - Fatigue : $\{(SD\text{-}Task_K, SD\text{-}Resource_L, \{(t_i, R)\})\}$
    - Experience : $\{(SD\text{-}Task_K, SD\text{-}Resource_L, \{(t_i, R)\})\}$
- Effects on process =
  - Project effects =
    - Productivity : $\{(effect_K, \{(t_i,y)\})\}$
    - defect generation : $\{(effect_K, \{(t_i,y)\})\}$
    - defect detection : $\{(effect_K, \{(t_i,y)\})\}$
    - defect rework : $\{(effect_K, \{(t_i,y)\})\}$
  - Task effects =
    - Productivity : $\{(SD\text{-}Task_K, effect_L, \{(t_i,y)\})\}$
    - defect generation : $\{(SD\text{-}Task_K, effect_L, \{(t_i,y)\})\}$
    - defect detection : $\{(SD\text{-}Task_K, effect_L, \{(t_i,y)\})\}$
    - defect rework : $\{(SD\text{-}Task_K, effect_L, \{(t_i,y)\})\}$

Data types:

SD-Task	= *<SD structure>*
SD-Resource	= *<SD structure>*
$effect_K$	= *<SD model variable>*

Formal specification of the SYDPIM object "SYMDB"

C.4 – PERT/CPM past behaviour

Object: PERT/CPM past behaviour

PERT/CPM past behaviour =

- Schedule =
  - Project Schedule =
    - Finish date : {(t_i, date)}
  - Tasks Schedule =
    - Start date : {(SD-Task$_K$, {(t_i, date)})}
    - Finish date : {(SD-Task$_K$, {(t_i, date)})}
- Effort =
  - Project Effort =
    - ACWP : {(t_i, R+)}
    - ACWP Engineering : {(t_i, R+)}
    - ACWP Management : {(t_i, R+)}
    - BCWP : {(t_i, R+)}
    - BCWS : {(t_i, R+)}
    - CTC : {(t_i, R+)}
    - CAC : {(t_i, R+)}
  - Task Effort =
    - ACWP : {(SD-Task$_K$, {(t_i, R+)})}
    - ACWP Engineering : {(SD-Task$_K$, {(t_i, R+)})}
    - ACWP Management : {(SD-Task$_K$, {(t_i, R+)})}
    - BCWP : {(SD-Task$_K$, {(t_i, R+)})}
    - BCWS : {(SD-Task$_K$, {(t_i, R+)})}
    - CTC : {(SD-Task$_K$, {(t_i, R+)})}
    - CAC : {(SD-Task$_K$, {(t_i, R+)})}
- Scope =
  - Project Scope =
    - SCAC : {(t_i, R+)}
    - CSCC : {(t_i, R+)}
    - ASCWP : {(t_i, R+)}
    - SCTC : {(t_i, R+)}
  - Task Scope =
    - SCAC : {(SD-Task$_K$, {(t_i, R+)})}
    - CSCC : {(SD-Task$_K$, {(t_i, R+)})}
    - ASCWP : {(SD-Task$_K$, {(t_i, R+)})}
    - SCTC : {(SD-Task$_K$, {(t_i, R+)})}
- Resources =
  - Project Resources (availability) =
    - ASP : {(SD-Resource$_I$, {(t_i, R+)})}
    - PSP : {(SD-Resource$_I$, {(t_i, R+)})}
    - CASP : {(SD-Resource$_I$, {(t_i, R+)})}
    - CPSP : {(SD-Resource$_I$, {(t_i, R+)})}
    - CSPAC : {(SD-Resource$_I$, {(t_i, R+)})}
  - Task Resources (allocation) =
    - ASP : {(SD-Task$_K$, SD-Resource$_I$, {(t_i, R+)})}
    - PSP : {(SD-Task$_K$, SD-Resource$_I$, {(t_i, R+)})}
    - CASP : {(SD-Task$_K$, SD-Resource$_I$, {(t_i, R+)})}
    - CPSP : {(SD-Task$_K$, SD-Resource$_I$, {(t_i, R+)})}
    - CSPAC : {(SD-Task$_K$, SD-Resource$_I$, {(t_i, R+)})}

Data types:

SD-Task = *<SD structure>*
SD-Resource = *<SD structure>*

Formal specification of the SYDPIM object "PERT/CPM past behaviour"

C.5 – Project past behaviour

Object: Project past behaviour

Project past behaviour =
IF SYMDB is being maintained THEN =
SYMDB – SYMDB.Uncovered Metrics
Expert judgement patterns =
Risk-related =
Risk-pattern-x : generic-pattern
Intangible issues =
Intangible-pattern-x : generic-pattern
ELSE
IF PERT/CPM model is updated THEN =
PERT/CPM past behaviour
Expert judgement patterns =
SYMDB patterns =
SYMDB.Collected Metrics.Quality
SYMDB.Calculated Metrics.Performance Indices
SYMDB.Calculated Metrics.Process Metrics
Other patterns =
Risk-related =
Risk-pattern-x : generic-pattern
Intangible issues =
Intangible-pattern-x : generic-pattern
ELSE
Expert judgement patterns =
SYMDB – SYMDB.Uncovered Metrics
Other patterns =
Risk-related =
Risk-pattern-x : generic-pattern
Intangible issues =
Intangible-pattern-x : generic-pattern

Data types:

generic-pattern =
$\{(t_i, R+)\}$ |
$\{(\text{SD-Task}_k, \{(t_i, R+)\})\}$ |
$\{(\text{SD-Resource}_l, \{(t_i, R+)\})\}$ |
$\{(\text{SD-Task}_k, \text{SD-Resource}_l, \{(t_i, R+)\})\}$

Formal specification of the SYDPIM object "Project past behaviour"

C.6 – PERT/CPM future behaviour

Object: PERT/CPM future behaviour

PERT/CPM future behaviour =
- Schedule =
  - Project Schedule =
    - Finish date : {(t_i, date)}
  - Tasks Schedule =
    - Start date : {(SD-Task$_k$, {(t_i, date)})}
    - Finish date : {(SD-Task$_k$, {(t_i, date)})}
- Effort =
  - Project Effort =
    - ACWP : {(t_i, R+)}
    - ACWP Engineering : {(t_i, R+)}
    - ACWP Management : {(t_i, R+)}
    - BCWP : {(t_i, R+)}
    - BCWS : {(t_i, R+)}
    - CTC : {(t_i, R+)}
    - CAC : {(t_i, R+)}
  - Task Effort =
    - ACWP : {(SD-Task$_k$, {(t_i, R+)})}
    - ACWP Engineering : {(SD-Task$_k$, {(t_i, R+)})}
    - ACWP Management : {(SD-Task$_k$, {(t_i, R+)})}
    - BCWP : {(SD-Task$_k$, {(t_i, R+)})}
    - BCWS : {(SD-Task$_k$, {(t_i, R+)})}
    - CTC : {(SD-Task$_k$, {(t_i, R+)})}
    - CAC : {(SD-Task$_k$, {(t_i, R+)})}
- Scope =
  - Project Scope =
    - SCAC : {(t_i, R+)}
    - CSCC : {(t_i, R+)}
    - ASCWP : {(t_i, R+)}
    - SCTC : {(t_i, R+)}
  - Task Scope =
    - SCAC : {(SD-Task$_k$, {(t_i, R+)})}
    - CSCC : {(SD-Task$_k$, {(t_i, R+)})}
    - ASCWP : {(SD-Task$_k$, {(t_i, R+)})}
    - SCTC : {(SD-Task$_k$, {(t_i, R+)})}
- Resources =
  - Project Resources (availability) =
    - ASP : {(SD-Resource$_l$, {(t_i, R+)})}
    - PSP : {(SD-Resource$_l$, {(t_i, R+)})}
    - CASP : {(SD-Resource$_l$, {(t_i, R+)})}
    - CPSP : {(SD-Resource$_l$, {(t_i, R+)})}
    - CSPAC : {(SD-Resource$_l$, {(t_i, R+)})}
  - Task Resources (allocation) =
    - ASP : {(SD-Task$_k$, SD-Resource$_l$, {(t_i, R+)})}
    - PSP : {(SD-Task$_k$, SD-Resource$_l$, {(t_i, R+)})}
    - CASP : {(SD-Task$_k$, SD-Resource$_l$, {(t_i, R+)})}
    - CPSP : {(SD-Task$_k$, SD-Resource$_l$, {(t_i, R+)})}
    - CSPAC : {(SD-Task$_k$, SD-Resource$_l$, {(t_i, R+)})}

Data types:

SD-Task	= *<SD structure>*
SD-Resource	= *<SD structure>*

Formal specification of the SYDPIM object "PERT/CPM future behaviour"

Appendix D – SYDPIM analytical links

D.1 – Structural links

Structural correspondence

SC-WBS: Structural correspondence of work breakdown

SC-WBS(PERT/CPM Plan, SD Model) =
- Relationships =
  - WBS-Map : {(SD-Task$_i$, {task$_k$,})}
- Operators =
  - Is_mapped : task$_k$ x SD-Task$_i$ → T | F
  - Mapped_to_SD : SD-Task$_i$ → {task$_k$}
  - Is_PERT_ENG : task$_k$ → T | F
  - Is_PERT_MAN : task$_k$ → T | F
  - Is_SD_ENG : SD-Task$_k$ → T | F
  - Is_SD_MAN : SD-Task$_k$ → T | F
  - Is_SD_HRM : SD-Task$_k$ → T | F
- Validity =
  - No_partial_map =
    - IF Is_mapped(task$_k$, SD-Task$_i$) THEN
    - WHATEVER SD-Task$_j$ NOT Is_mapped(task$_k$, SD-Task$_j$)
  - SD_mapping =
    - FOR EACH SD-Task$_k$ THERE IS AT LEAST ONE task$_i$
    - SO THAT Is_mapped(SD-Task$_k$, task$_i$)
  - PERT/CPM _mapping =
    - FOR EACH task$_k$ THERE IS AT LEAST ONE SD-Task$_i$
    - SO THAT Is_mapped(SD-Task$_i$, task$_k$)
  - Task_types:
    - IF Is_mapped(task$_k$, SD-Task$_i$) THEN
      - IF Is_PERT_ENG(task$_k$) THEN Is_SD_ENG(SD-Task$_i$)

Object variables:

task$_k$ = PERT/CPM Plan.<x>.Task[k]
- <x> = Past segment or Future segment

SD-Task$_k$ =SD Model.SD model architecture.SD Tasks[k]

SC-OBS: Structural correspondence of organisation breakdown

SC-OBS(PERT/CPM Plan, SD Model) =
- Relationships =
  - OBS-Map : {(SD-Resource$_i$, {resource$_k$,})}
- Operators =
  - Is_mapped : resource$_k$ x SD-Resource$_i$ → T | F
  - Mapped_to_SD : SD-Resource$_i$ → {resource$_k$}
- Validity =
  - No_partial_map =
  - IF Is_mapped(resource$_k$, SD-Resource$_i$) THEN
    - WHATEVER SD-Resource$_j$
      - NOT Is_mapped(resource$_k$, SD-Resource$_j$)
  - SD_mapping =
    - FOR EACH SD-Resource$_k$ THERE IS AT LEAST ONE resource$_i$
    - SO THAT Is_mapped(SD-Resource$_k$, resource$_i$)
  - PERT/CPM _mapping =
    - FOR EACH resource$_k$ THERE IS AT LEAST ONE SD-Resource$_i$
    - SO THAT Is_mapped(SD-Resource$_i$, resource$_k$)

Object variables:

resource$_k$ = PERT/CPM Plan.<x>.Resources[k]
- <x> = Past segment or Future segment

SD-Resource$_k$ =SD Model.SD model architecture.SD Resources[k]

SC-WD: Structural correspondence of work dependencies

SC-WD(PERT/CPM Plan, SD Model) =
Relationships =
Intra_SD-Task : {(SD-Task$_i$, {dependency$_k$,})}
Inter_SD-Task : {(SD-Dependency$_i$, {dependency$_k$,})}
Operators =
Is_mapped_intra : dependency$_k$ x SD-Task$_i$ → T | F
Is_mapped_inter : dependency$_k$ x SD-Dependency$_i$ → T | F
Mapped_to_SD_intra : SD-Task$_i$ → {dependency$_k$}
Mapped_to_SD_inter : SD-Dependency$_i$ → {dependency$_k$}

Validity =
No_partial_map_intra =
IF Is_mapped_intra(dependency$_k$, SD-Task$_i$) THEN
WHATEVER SD-Task$_j$
NOT Is_mapped_intra(dependency$_k$, SD-Task$_j$)
No_partial_map_inter =
IF Is_mapped_inter(dependency$_k$, SD-Dependency$_i$) THEN
WHATEVER SD-Dependency$_j$
NOT Is_mapped_inter(dependency$_k$, SD-Dependency$_j$)
Consistency_intra =
IF Is_mapped_intra(dependency$_k$, SD-Task$_i$) THEN
SC-WBS(PERT/CPM Plan, SD Model).Operators.Is_mapped(dependency$_k$[i],SD-Task$_i$)
AND
SC-WBS(PERT/CPM Plan, SD Model).Operators.Is_mapped(dependency$_k$[i][j],SD-Task$_i$)
Consistency_inter =
IF Is_mapped_inter(dependency$_k$, SD-Dependency$_i$) THEN
SC-WBS(PERT/CPM Plan, SD Model).Operators.Is_mapped(dependency$_k$[i], SD-Dependency$_i$.Predecessor)
AND
SC-WBS(PERT/CPM Plan, SD Model).Operators.Is_mapped(dependency$_k$[i][j], SD-Dependency.Successor$_j$)

Object variables:

dependency$_k$ = PERT/CPM Plan.<x>.Dependencies[k]
<x> = Past segment or Future segment

SD-Dependency$_k$ =SD Model.SD model architecture.SD-Inter-Task-Dependencies[k]

SD-Task$_k$ =SD Model.SD model architecture.SD Tasks[k]

Structural consistency

SCN-RA: Structural consistency of resource allocation

SCN-RA(PERT/CPM Plan, SD Model)=
Relationship =
IF PERT/CPM_Is_allocated($resource_k$, $task_i$) THEN
IF SC-WBS(PERT/CPM Plan, SD Model).Operators.Is_mapped($task_i$, SD-$Task_j$) AND SC-OBS(PERT/CPM Plan, SD Model).Operators.Is_mapped($resource_k$, SD-$Resource_l$)
THEN
SD_Is_allocated(SD-$Resource_l$, SD-$Task_j$)
Operators =
PERT/CPM_Is_allocated : $resource_k$ x $task_i$ → T | F
SD_Is_allocated : SD-$Resource_k$ x SD-$Task_i$ → T | F
Object variables:

$task_k$ = PERT/CPM Plan.<x>.Tasksk[k]
<x> = Past segment or Future segment

SD-$Task_k$ =SD Model.SD model architecture.SD Tasks[k]

$resource_k$ = PERT/CPM Plan.<x>.Resources[k]
<x> = Past segment or Future segment

SD-$Resource_k$ =SD Model.SD model architecture.SD Resources[k]

SCN-WD: Structural consistency of work dependencies

SCN-WD(PERT/CPM Plan, SD Model) =
Relationships =
Condition_intra =
IF SC-WD(PERT/CPM Plan, SD Model).Is_mapped_intra(dependency$_k$, SD-Task$_l$)
THEN
SC-WBS(PERT/CPM Plan, SD model).Operators.Is_mapped(dependency$_k$[i],SD-Task$_l$)
AND
SC-WBS(PERT/CPM Plan, SD Model).Operators.Is_mapped(dependency$_k$[i][j],SD-Task$_l$)
Condition_inter =
IF SC-WD(PERT/CPM Plan, SD Model).Is_mapped_inter(dependency$_k$, SD- Dependency$_l$)
THEN
SC-WBS(PERT/CPM Plan, SD Model).Operators.Is_mapped(dependency$_k$[i], SD-Dependency$_l$.Predecessor)
AND
SC-WBS(PERT/CPM Plan, SD Model).Operators.Is_mapped(dependency$_k$[i][j], SD-Dependency.Successor$_l$)

Object variables:

dependency$_k$ = PERT/CPM Plan.<x>.Dependencies[k]
<x> = Past segment or Future segment

SD-Dependency$_k$ =SD Model.SD model architecture.SD-Inter-Task-Dependencies[k]

SD-Task$_k$ =SD Model.SD model architecture.SD Tasks[k]

D.2 – Data links

Data exchange links

DEI Links

DEI-1: Initially Planned Profiles of Resources Availability

Definition:

For those profiles of resource availability defined in the PERT/CPM model at the same level of aggregation as in the SD model, and in both models these profiles also incorporate the same elementary PERT/CPM resources, then the following links can be established:

- *transfer initially planned profile of resources availability from PERT/CPM to SD*
- *transfer initially planned profile of resources availability from SD to PERT/CPM*

Usage: transfer initial project plan from one model to the other

Formal specification:

DEI-1(PERT/CPM model, SD project model) =
  Relationships =
    SD_to_PERT =
      PERT/CPM.Init_Plan _Res_Avai[r_i][t] := SD.Init_Plan_Res_Avail[r_i][t]
    PERT_to_SD =
      SD.Init_Plan_Res_Avail[r_i][t] := PERT/CPM.Init_Plan_Res_Avail[r_i][t]

Where: r_i = SD-Resource[i]

Short references:

DEI-1.SD-PERT = DEI-1().Relationships.SD_to_PERT
DEI-1.PERT-SD = DEI-1().Relationships.PERT_to_SD

Object variables:

PERT/CPM.Init_Plan_Res_Avail [r_i][t] =
PERT/CPM model.Initial plan.Plan_Resource_availability[r_i][t]

SD.Init_Plan_Res_Avail [r_i][t] =
SD project model.SD plan.Initial plan.Resources.Project Resources[r_i][t]

DEI-2: Project start date

Definition:

The project initial date which specifies the date when the whole project is initiated (planned if the project has not started yet; actual otherwise) is an input to both models and can be transferred from one mode to the other.

Usage: transfer initial project plan from one model to the other

Formal specification:

DEI-2(PERT/CPM model, SD project model) =
    Relationships =
        SD_to_PERT =
            PERT/CPM.Project_Initial_Date := SD.Project_Initial_Date
        PERT_to_SD =
            SD.Project_Initial_Date := PERT/CPM.Project_Initial_Date

Short references:

DEI-2.SD-PERT = DEI-2().Relationships.SD_to_PERT
DEI-2.PERT-SD = DEI-2().Relationships.PERT_to_SD

Object variables:

PERT/CPM.Project_Initial_Date =
PERT/CPM model.Initial date

SD.Project_Initial_Date =
SD project model.SD plan.Initial plan.Schedule.Project scehdule.Start date

DEI-3: SD Adjustment of Actual and Currently Planned Profiles of Resources Availability

Definition:

For those profiles of resource availability defined in the PERT/CPM model at the same level of aggregation as in the SD model, and in both models these profiles also incorporate the same elementary PERT/CPM resources, then the following links can be established:

- *produce SD input exogenous decision to readjust the past resource availability profile from the initially planned profile to the actual profile in the PERT/CPM model;*
- *produce SD input exogenous decision to readjust the present resource level produced by the SD model (present data-point only);*
- *produce SD input exogenous decision to readjust the future planned resource availability level from the initially planned profile to the currently planned profile in the PERT/CPM model.*

Usage: transfer project status or new project plan from PERT/CPM to SD model

Formal specification:

DEI-3(PERT/CPM model, SD project model) =
  Relationships =
    Past =
    SD.Exog_Adjust_Plan_Profile_Past[r_i][t] :=
      PERT/CPM.Actual_Res_Avai_Past[r_i][t] –
      SD.Init_Plan_Res_Avai[r_i][t]
    Present =
    SD.Exog_Adjust_Plan_Profile_Past[r_i][present] :=
      PERT/CPM.Actual_Res_Avai_Past[r_i][present] –
      SD.Actual_Res_Avai_Past[r_i][present]
    Future =
      Present_gap := (SD.Actual_Res_Avai_Past[r_i][present] –
        SD.Init_Plan_Res_Avai[r_i][present]))
    SD.Exog_Adjust_Plan_Profile_Future[r_i][t] :=
      PERT/CPM.Plan_Res_Avai_Future[r_i][t] –
      (SD.Init_Plan_Res_Avai[r_i][t] + Present_gap)

Where:

- r_i = SD-Resource[i]

Short references:

DEI-3.1.PERT-SD = DEI-3().Relationships.Future
DEI-3.2.PERT-SD = DEI-3().Relationships.Present
DEI-3.3.PERT-SD = DEI-3().Relationships.Past

Object variables:

PERT/CPM.Actual_Res_Avail_Past[r_i][t] =
PERT/CPM model.Current PERT/CPM plan.Past segment.Actual_Resource-availability[r_i][t]

PERT/CPM.Plan_Res_Avail_Future[r_i][t] =
PERT/CPM model.Current PERT/CPM plan.Future segment.Plan_Resource-availability[r_i][t]

SD.Exog_Adjust_Plan_Profile_Past[r_i][t] =
SD project model.SD plan.Past segment.Resources.Project Resources.Exogenous[r][t]

SD.Exog_Adjust_Plan_Profile_Future[r_i][t] =
SD project model.SD plan.Future segment.Resources.Project Resources.Exogenous[r][t]

SD.Init_Plan_Res_Avai [r_i][t] =
SD project model.SD plan.Initial plan.Resources.Project Resources.[r_i][t]

SD.Actual_Res_Avai_Past[r_i][t] =
SD project model.Project Behaviour.Past Segment.Resources.Project Resources.[r_i][t]

DEOI Links

DEOI-1: SD-Task Initially Planned Schedules

Definition:

For each SD-Task[i] in the SD model, the initially planned start and finishing dates can be transferred from the set of PERT/CPM tasks mapped to it, according to the SC-WBS link. The start date is the minimum start date of the PERT/CPM tasks in the set and the finishing date is the maximum finishing date of these tasks.

Usage: transfer initial project plan from PERT/CPM model to SD model

Formal specification:

DEOI-1(PERT/CPM Model, SD Model) =
Relationships =
Start_Date =
SD-Task[i].Init_Plan_Start_Date :=
MIN {PERT/CPM-Task[k].Init_Plan_Start_Date }
Finish_Date =
SD-Task[i].Init_Plan_Finish_Date :=
MAX {PERT/CPM-Task[k].Init_Plan_Finish_Date }
Validity =
SC-WBS(PERT/CPM Model.Initial PERT/CPM plan, SD Model).Operators.
Is_mapped(PERT/CPM-Task[k], SD-Task[i])

where:

- MIN is a function that selects the minimum value of a set
- MAX is a function that selects the maximum value of a set

Short references:

DEOI-1.PERT-SD = DEOI-1().Relationships
DEOI-1.Validity = DEOI-1().Validity

Object variables:

PERT/CPM-Task[k] =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k]

PERT/CPM-Task[k].Init_Plan_Start_Date =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k].planned start date

PERT/CPM-Task[k].Init_Plan_Finish_Date =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k].planned finish date

SD-Task[i] =
SD project model.SD model architecture.SD-Tasks[i]

SD-Task[i].Init_Plan_Start_Date =
SD project model.SD plan.Initial plan.Schedule.Task Schedule.Start date[i]

SD-Task[i].Init_Plan_Finish_Date =
SD project model.SD plan.Initial plan.Schedule.Task Schedule.Finish date[i]

DEOI-2: SD-Task Initially Planned Budget

Definition:

For each SD-Task[i] in the SD model, the initially planned budget can be transferred from the set of PERT/CPM tasks mapped to it, according to the SC-WBS link. This budget is the sum of the of the initially planned budgets of all PERT/CPM tasks in the set.

Usage: transfer initial project plan from PERT/CPM model to SD model

Formal specification:

DEOI-2(PERT/CPM Model,SD Model) =
Relationships =
Budget =
SD-Task[i].Init_Budget :=
SUM {PERT/CPM-Task[k].Init_Budget}
Validity =
SC-WBS(PERT/CPM Model.Initial PERT/CPM plan, SD Model).Operators.
Is_mapped(PERT/CPM-Task[k], SD-Task[i])

where:

- SUM is a function that sums the values in a set

Short references:

DEOI-2.PERT-SD = DEOI-2().Relationships
DEOI-2.Validity = DEOI-2().Validity

Object variables:

PERT/CPM-Task[k] =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k]

PERT/CPM-Task[k].Init_Budget =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k].planned budget

SD-Task[i] =
SD project model.SD model architecture.SD-Tasks[i]

SD-Task[i].Init_Budget =
SD project model.SD plan.Initial plan.Budget.Task budget[i]

DEOI-3: SD-Task Initially Planned scope

Definition:

For each SD-Task[i] in the SD model, the initially planned scope can be transferred from the set of PERT/CPM tasks mapped to it, according to the SC-WBS link. This scope is the sum of the of the initially planned scope of all PERT/CPM tasks in the set.

Usage: transfer initial project plan from PERT/CPM model to SD model

Formal specification:

DEOI-3(PERT/CPM Model,SD Model) =
Relationships =
Scope =
SD-Task[i].Init_Plan_Scope :=
SUM {PERT/CPM-Task[k].Init_Plan_Scope}
Validity =
SC-WBS(PERT/CPM Model.Initial PERT/CPM plan, SD Model).Operators.
Is_mapped(PERT/CPM-Task[k], SD-Task[i])

where:

- SUM is a function that sums the values in a set

Short references:

DEOI-3.PERT-SD = DEOI-3().Relationships
DEOI-3.Validity = DEOI-3().Validity

Object variables:

PERT/CPM-Task[k] =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k]

PERT/CPM-Task[k].Init_Plan_Scope =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k].planned scope

SD-Task[i] =
SD project model.SD model architecture.SD-Tasks[i]

SD-Task[i].Init_Plan_Scope =
SD project model.SD plan.Initial plan.Scope.Task scope[i]

DEOI-4: SD-Task Initially Planned Resource Allocation

Definition:

For each SD-Task[i] in the SD model, the initially planned profiles of resource allocation for each resource type can be transferred from the set of PERT/CPM tasks mapped to that SD-Task, according to the SC-WBS and SC-OBS links. For each SD resource type, the allocation profile is the sum of the planned profiles of each PERT/CPM resource mapped to it, of all PERT/CPM tasks in the set.

Usage: transfer initial project plan from PERT/CPM model to SD model

Formal specification:

DEOI-4(PERT/CPM model, SD project model) =
  Relationships =
    Resource =
      SD-Task[i].Init_Plan_Resource[r] :=
      SUMT {PERT/CPM-Task[k].Init_Plan_Resource [m]}
  Validity =
    SC-WBS(PERT/CPM Model.Initial PERT/CPM plan, SD Model).Operators.
    Is_mapped(PERT/CPM-Task[k], SD-Task[i])
  AND
    SC-OBS(PERT/CPM Model.Initial PERT/CPM plan, SD Model).Operators.
    Is_mapped(PERT/CPM-Resource[m], SD-Resource[r])

where:

- SUMT is a function that sums various over-time patterns in a set

Short references:

DEOI-4.PERT-SD = DEOI-4().Relationship
DEOI-4.Validity = DEOI-4().Validity

Object variables:

PERT/CPM-Task[k] =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k]

PERT/CPM-Task[k].Init_Plan_Resource[m] =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k].plan resource profile[m]

PERT/CPM-Resource[m] =
PERT/CPM model.Initial PERT/CPM plan.Resources[m]

SD-Task[i] =
SD project model.SD model architecture.SD-Tasks[i]

SD-Task[i].Init_Plan_Resource[r] =
SD project model.SD plan.Inital plan.Resources.Task resources[i,r]

SD-Resource[r] =
SD project model.SD model architecture.SD Resources[r]

DEOI-5: SD-Task Initially Planned Budget Breakdown

Definition:

For each SD-Task[i] in the SD model, the initially planned budget for each work activity can be transferred from the set of PERT/CPM tasks mapped to it, according to the SC-WBS link. The budget for each activity is the sum of the budgets of all PERT/CPM tasks mapped to the SD-Task which are of the same work type as the activity.

Usage: transfer initial project plan from PERT/CPM model to SD model

Formal specification:

DEOI-5(PERT/CPM Model,SD Model) =
  Relationships =
    Budget =
      SD-Task[i].Init_Budget_Activity [j] :=
      SUM {PERT/CPM-Task[k].Init_Budget}
  Validity =
    SC-WBS(PERT/CPM Model.Initial PERT/CPM plan, SD Model).Operators.
    Is_mapped(PERT/CPM-Task[k], SD-Task[i])
    AND
    [*PERT/CPM model.Operators.Is_Type(PERT/CPM-Task[k].SD-Activity[j])*]

where:

- SUM is a function that sums the values in a set

Short references:

DEOI-5.PERT-SD = DEOI-5().Relationships
DEOI-5.Validity = DEOI-5().Validity

Object variables:

PERT/CPM-Task[k] =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k]

PERT/CPM-Task[k].Init_Budget =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k].planned budget

SD-Task[i] =
SD project model.SD model architecture.SD-Tasks[i]

SD-Task[i].Init_Budget_Activity[j] =
SD project model.SD plan.Initial plan.Budget.Task Budget[i,j]

DEOI-6: Adjustment of SD-Task Currently Planned Schedules

Definition:

For each SD-Task[i] in the SD model, an exogenous decision can be generated to adjust the currently planned start and finishing dates according to the current PERT/CPM plan. This exogenous decision is equals the present gap between the dates produced by the SD model and the dates derived from the current PERT/CPM plan. The exogenous decision to adjust the start date can only be generated for SD-Tasks which have not started yet and the exogenous decision to adjust the finishing date can only be generated for SD-Tasks which are not complete yet.

Usage: transfer project status or new project plan from PERT/CPM model to SD model

Formal specification:

DEOI-6(PERT/CPM Model,SD Model) =
Relationships =
Start_Date =
SD.Exog_Adjust.Task[i].Plan_Start_Date[present] :=
MIN {PERT/CPM-Task[k].Plan_Start_Date} –
SD-Task[i].Plan_Start_Date[present]
Finish_Date =
SD.Exog_Adjust.Task[j].Plan_Finish_Date[present] :=
MAX {PERT/CPM-Task[m].Plan_Finish_Date} –
SD-Task[j].Plan_Finish_Date[present]
Validity =
SC-WBS(PERT/CPM Model.Current PERT/CPM plan, SD Model).Operators.
Is_mapped(PERT/CPM-Task[k], SD-Task[i])
AND
SC-WBS(PERT/CPM Model.Current PERT/CPM plan, SD Model).Operators.
Is_mapped(PERT/CPM-Task[m], SD-Task[j])
AND MIN {PERT/CPM-Task[k]. Plan_Start_Date} > TIME
AND MAX {PERT/CPM-Task[m].Plan_Finish_Date} > TIME

where:

- MIN is a function that selects the minimum value of a set
- MAX is a function that selects the maximum value of a set

Short references:

DEOI-6.PERT-SD = DEOI-6().Relationships
DEOI-6.Validity = DEOI-6().Validity

Object variables:

PERT/CPM-Task[k] =
PERT/CPM model.Current PERT/CPM plan.Future segment.Tasks[k]

PERT/CPM-Task[k].Plan_Start_Date =
PERT/CPM model.Current PERT/CPM plan.Future segment.Tasks[k].planned start date

PERT/CPM-Task[m] =
PERT/CPM model.Current PERT/CPM plan.<x>.Tasks[m]
<x> = Past segment + Future segment

PERT/CPM-Task[m].Plan_Finish_Date =
PERT/CPM model.Current PERT/CPM plan.<x>.Tasks[m].planned finish date
<x> = Past segment + Future segment

SD-Task[i] =
SD project model.SD model architecture.SD-Tasks[i]

SD-Task[i].Plan_Start_Date[present] =
SD project model.Project behaviour.Past segment.Schedule.Task Schedule.Start date[i][present]

SD.Exog_Adjust.Task[i].Plan_Start_Date[present] =
SD project model.SD plan.Past segment.Schedule.Task Schedule.Start_Date.Exogenous[i][present]

SD-Task[j] =
SD project model.SD model architecture.SD-Tasks[j]

SD-Task[j].Plan_Finish_Date[present] =
SD project model.Project behaviour.Past segment.Schedule.Task Schedule.Finish date[j][present]

SD.Exog_Adjust.Task[j].Plan_Finish_Date[present] =
SD project model.SD plan.Past segment.Schedule.Task Schedule.Finish_Date.Exogenous[j][present]

DEOI-7: Adjustment of SD-Task Currently Planned Cost at Completion (CAC)

Definition:

For each SD-Task[i] in the SD model, an exogenous decision can be generated to adjust the currently planned cost at completion according to the current PERT/CPM plan. This exogenous decision is equals the present gap between the CAC produced by the SD model at the present moment and the CAC derived from the current PERT/CPM plan. This exogenous decision can only be generated for SD-Tasks which are not complete yet.

Usage: transfer project status or new project plan from PERT/CPM model to SD model

Formal specification:

```
DEOI-7(PERT/CPM Model,SD  Model) =
  Relationships =
        Budget =
                SD.Exog_Adjust.Task[i].Plan_CAC[present] :=
                (SUM {PERT/CPM-Task[k].Actual_Cost} +
                 SUM {PERT/CPM-Task[m].Plan_Budget} ) –
                 SD-Task[i].Plan_CAC[present]
  Validity =
    SC-WBS(PERT/CPM Model.Current PERT/CPM plan, SD  Model).Operators.
    Is_mapped(PERT/CPM-Task[k], SD-Task[i])
    AND
    SC-WBS(PERT/CPM Model.Current PERT/CPM plan, SD  Model).Operators.
    Is_mapped(PERT/CPM-Task[m], SD-Task[i])
    AND
    MAX( MAX {PERT/CPM-Task[k].Plan_Finish_Date},
          MAX {PERT/CPM-Task[m].Plan_Finish_Date} ) > TIME
```

where:

- SUM is a function that sums the values in a set

Short references:

DEOI-7.PERT-SD = DEOI-7().Relationships
DEOI-7.Validity = DEOI-7().Validity

Object variables:

PERT/CPM-Task[k] =
PERT/CPM model.Current PERT/CPM plan.Past segment.Tasks[k]

PERT/CPM-Task[m] =
PERT/CPM model.Current PERT/CPM plan.Future segment.Tasks[k]

PERT/CPM-Task[k].Actual_Cost =
PERT/CPM model.Current PERT/CPM plan.Past segment.Tasks[k].actual effort spent to date

PERT/CPM-Task[k].Plan_Finish_Date =
PERT/CPM model.Current PERT/CPM plan.Past segment.Tasks[k].planned finish date

PERT/CPM-Task[m].Plan_Budget =
PERT/CPM model.Current PERT/CPM plan.Future segment.Tasks[m].planned budget

PERT/CPM-Task[m].Plan_Finish_Date =
PERT/CPM model.Current PERT/CPM plan.Future segment.Tasks[k].planned finish date

SD-Task[i] =
SD project model.SD model architecture.SD-Tasks[i]

SD-Task[i].Plan_CAC[present] =
SD project model.Project behaviour.Past segment.Effort.Task Effort.CAC[i][present]

SD.Exog_Adjust.Task[i].Plan_CAC[present] =
SD project model.SD plan.Past segment.Budget.Task Budget.Exogenous[i][present]

DEOI-8: Adjustment of SD-Task Currently Planned Scope at Completion (SCAC)

Definition:

For each SD-Task[i] in the SD model, an exogenous decision can be generated to adjust the currently planned scope at completion according to the current PERT/CPM plan. This exogenous decision is equals the present gap between the SCAC produced by the SD model at the present moment and the SCAC derived from the current PERT/CPM plan. This exogenous decision can only be generated for SD-Tasks which are not complete yet.

Usage: transfer project status or new project plan from PERT/CPM model to SD model

Formal specification:

```
DEOI-8(PERT/CPM Model,SD  Model) =
  Relationships =
        Scope =
                SD.Exog_Adjust.Task[i].Plan_SCAC[present] :=
                (SUM {PERT/CPM-Task[k].Actual_Scope} +
                 SUM {PERT/CPM-Task[m].Plan_Scope} ) –
                 SD-Task[i].Plan_SCAC[present]
  Validity =
    SC-WBS(PERT/CPM Model.Current PERT/CPM plan, SD  Model).Operators.
    Is_mapped(PERT/CPM-Task[k], SD-Task[i])
    AND
    SC-WBS(PERT/CPM Model.Current PERT/CPM plan, SD  Model).Operators.
    Is_mapped(PERT/CPM-Task[m], SD-Task[i])
    AND
    MAX( MAX {PERT/CPM-Task[k].Plan_Finish_Date},
          MAX {PERT/CPM-Task[m].Plan_Finish_Date} ) > TIME
```

where:

- SUM is a function that sums the values in a set

Short references:

DEOI-8.PERT-SD = DEOI-8().Relationships
DEOI-8.Validity = DEOI-8().Validity

Object variables:

PERT/CPM-Task[k] =
PERT/CPM model.Current PERT/CPM plan.Past segment.Tasks[k]

PERT/CPM-Task[k].Actual_Scope =
PERT/CPM model.Current PERT/CPM plan.Past segment.Tasks[k].actual scope to date

PERT/CPM-Task[k].Plan_Finish_Date =
PERT/CPM model.Current PERT/CPM plan.Past segment.Tasks[k].planned finish date

PERT/CPM-Task[m] =
PERT/CPM model.Current PERT/CPM plan.Future segment.Tasks[k]

PERT/CPM-Task[m].Plan_Scope =
PERT/CPM model.Current PERT/CPM plan.Future segment.Tasks[m].planned scope

PERT/CPM-Task[m].Plan_Finish_Date =
PERT/CPM model.Current PERT/CPM plan.Future segment.Tasks[k].planned finish date

SD-Task[i] =
SD project model.SD model architecture.SD-Tasks[i]

SD-Task[i].Plan_SCAC[present] =
SD project model.Project behaviour.Past segment.Scope.Task Scope.SCAC[i][present]

SD.Exog_Adjust.Task[i].Plan_SCAC[present] =
SD project model.SD plan.Past segment.Scope.Task Scope.Exogenous[i][present]

DEOI-10: PERT/CPM Adjustment of Actual and Currently Planned Profiles of Resources Availability

Definition:

For those profiles of resource availability defined in the PERT/CPM model at the same level of aggregation as in the SD model, and in both models these profiles also incorporate the same elementary PERT/CPM resources, then the following links can be established:

- *transfer actual profile of resources availability from SD to PERT/CPM*
- *transfer currently planned profile of resources availability from SD to PERT/CPM*

Usage: transfer project status or new project plan from SD model to PERT/CPM model

Formal specification:

DEOI-10(PERT/CPM model, SD project model) =
  Relationships =
    Past =
    PERT/CPM.Actual_Res_Avai_Past[r_i][t] :=
    SD.Actual_Res_Avai_Past[r_i][t]
    Future =
    PERT/CPM.Plan_Res_Avai_Future[r_i][t] :=
    SD.Plan_Res_Avai_Future[r_i][t]

Where:

- r_i = SD-Resource[i]

Short references:

DEOI-10.1.SD-PERT = DEOI-10().Relationships.Future
DEOI-10.2.SD-PERT = DEOI-10().Relationships.Past

Object variables:

PERT/CPM.Actual_Res_Avail_Past[r_i][t] =
PERT/CPM model.Current PERT/CPM plan.Past segment.Actual_Resource-availability[r_i][t]

PERT/CPM.Plan_Res_Avail_Future[r_i][t] =
PERT/CPM model.Current PERT/CPM plan.Future segment.Plan_Resource-availability[r_i][t]

SD.Actual_Res_Avai_Past[r_i][t] =
SD project model.Project Behaviour.Past Segment.Resources.Project Resources[r_i][t]

SD.Plan_Res_Avai_Future[r_i][t] =
SD project model.Project Behaviour.Future Segment.Resources.Project Resources[r_i][t]

Data consistency

DCI Links

DCI-1: Initially Planned Profiles of Project Resources Availability

Definition:

For those profiles of resource availability defined in the PERT/CPM model at the same level of aggregation as in the SD model, and in both models these profiles also incorporate the same elementary PERT/CPM resources, then the initially planned profiles input to the two models must be the same.

Usage: transfer initial project plan from one model to the other

Formal specification:

DCI-1(PERT/CPM model, SD project model) =
  Relationships =
    PERT/CPM.Init_Plan _Res_Avai[r_i][t] ~= SD.Init_Plan_Res_Avail[r_i][t]

Where: r_i = SD-Resource[i]

Short references:

DCI-1 = DCI-1().Relationships

Object variables:

PERT/CPM.Init_Plan_Res_Avail[r_i][t] =
PERT/CPM model.Init PERT/CPM plan.Plan_Resource_availability[r_i][t]

SD.Init_Plan_Res_Avail [r_i][t] =
SD project model.SD plan.Initial plan.Resources.Project Resources[r_i][t]

DCI-2: Project start date

Definition:

The project initial date which specifies the date where the whole project is initiated is an input to both models and must be the same.

Usage: transfer initial project plan from one model to the other

Formal specification:

DEI-2(PERT/CPM model, SD project model) =
    Relationships =
        PERT/CPM.Project_Initial_Date == SD.Project_Initial_Date

Short references:

DCI-2 = DEI-2().Relationships.SD_to_PERT

Object variables:

PERT/CPM.Project_Initial_Date = PERT/CPM model.Initial date

SD.Project_Initial_Date = SD project model.SD plan.Initial date

DCOI Links

DCOI-1: SD-Task Initially Planned Schedules

Definition:

For each SD-Task[i] in the SD model, their initially planned start and finishing dates must be the same as the initially planned earliest start date and latest finishing dates of the set of PERT/CPM tasks mapped to it according to the SC-WBS link.

Usage: transfer initial project plan from one model to the other

Formal specification:

```
DCOI-1(PERT/CPM Model,SD  Model) =
  Relationships =
        Start_Date =
                SD-Task[i].Init_Plan_Start_Date ==
                MIN {PERT/CPM-Task[k].Init_Plan_Start_Date }
         Finish_Date =
                SD-Task[i].Init_Plan_Finish_Date ==
                MAX {PERT/CPM-Task[k].Init_Plan_Finish_Date }
  Validity =
    SC-WBS(PERT/CPM Model.Initial PERT/CPM plan, SD  Model).Operators.
    Is_mapped(PERT/CPM-Task[k], SD-Task[i])
```

where:

- MIN is a function that selects the minimum value of a set
- MAX is a function that selects the maximum value of a set

Short references:

DCOI-1.PERT-SD = DCOI-1().Relationships
DCOI-1.Validity = DCOI-1().Validity

Object variables:

PERT/CPM-Task[k] =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k]

PERT/CPM-Task[k].Init_Plan_Start_Date =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k].planned start date

PERT/CPM-Task[k].Init_Plan_Finish_Date =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k].planned finish date

SD-Task[i] =
SD project model.SD model architecture.SD-Tasks[i]

SD-Task[i].Init_Plan_Start_Date =
SD project model.SD plan.Initial plan.Schedule.Task Schedule.Start date[i]

SD-Task[i].Init_Plan_Finish_Date =
SD project model.SD plan.Initial plan.Schedule.Task Schedule.Finish date[i]

DCOI-2: SD-Task Initially Planned Budget

Definition:

For each SD-Task[i] in the SD model, their initially planned budget must be the same as the sum of the initial budgets of the set of PERT/CPM tasks mapped to it according to the SC-WBS link.

Usage: transfer initial project plan from one model to the other

Formal specification:

DCOI-2(PERT/CPM model, SD project model) =
  Relationships =
    Budget =
      SD-Task[i].Init_Budget ==
      SUM {PERT/CPM-Task[k].Init_Budget}
  Validity =
    SC-WBS(PERT/CPM Model.Initial PERT/CPM plan, SD Model).Operators.
    Is_mapped(PERT/CPM-Task[k], SD-Task[i])

where:

- SUM is a function that sums the values in a set

Short references:

DCOI-2.PERT-SD = DCOI-2().Relationships
DCOI-2.Validity = DCOI-2().Validity

Object variables:

PERT/CPM-Task[k] =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k]

PERT/CPM-Task[k].Init_Budget =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k].planned budget

SD-Task[i] =
SD project model.SD model architecture.SD-Tasks[i]

SD-Task[i].Init_Budget =
SD project model.SD plan.Initial plan.Budget.Task budget[i]

DCOI-3: SD-Task Initially Planned scope

Definition:

For each SD-Task[i] in the SD model, their initially planned scope must be the same as the sum of the initially planned scopes of the set of PERT/CPM tasks mapped to it according to the SC-WBS link.

Usage: transfer initial project plan from one model to the other

Formal specification:

DCOI-3(PERT/CPM Model,SD Model) =
Relationships =
Scope =
SD-Task[i].Init_Plan_Scope ==
SUM {PERT/CPM-Task[k].Init_Plan_Scope}
Validity =
SC-WBS(PERT/CPM Model.Initial PERT/CPM plan, SD Model).Operators.
Is_mapped(PERT/CPM-Task[k], SD-Task[i])

where:

- SUM is a function that sums the values in a set

Short references:

DCOI-3.PERT-SD = DCOI-3().Relationships
DCOI-3.Validity = DEOI-3().Validity

Object variables:

PERT/CPM-Task[k] =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k]

PERT/CPM-Task[k].Init_Plan_Scope =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k].planned scope

SD-Task[i] =
SD project model.SD model architecture.SD-Tasks[i]

SD-Task[i].Init_Plan_Scope =
SD project model.SD plan.Initial plan.Scope.Task scope[i]

DCOI-4: SD-Task Initially Planned Resource Allocation

Definition:

For each SD-Task[i] in the SD model, their initially planned profiles of resource allocation must be the same as the sum of the initially planned profiles of the set of PERT/CPM tasks mapped to it according to the SC-WBS link. For each SD resource type the allocation profile is the sum of the allocation profiles in the PERT/CPM tasks of those PERT/CPM resources mapped to it according to the SC-OBS link.

Usage: transfer initial project plan from one model to the other

Formal specification:

DCOI-4(PERT/CPM model, SD project model) =
  Relationships =
      Resource =
          SD-Task[i].Init_Plan_Resource[r][t] ~=
          SUMT {PERT/CPM-Task[k].Init_Plan_Resource [m][t]}
  Validity =
      SC-WBS(PERT/CPM Model.Initial PERT/CPM plan, SD Model).Operators.
      Is_mapped(PERT/CPM-Task[k], SD-Task[i])
    AND
      SC-OBS(PERT/CPM Model.Initial PERT/CPM plan, SD Model).Operators.
      Is_mapped(PERT/CPM-Resource[m], SD-Resource[r])

where:

- SUMT is a function that sums various over-time patterns in a set

Short references:

DCOI-4.PERT-SD = DCOI-4().Relationship
DCOI-4.Validity = DCOI-4().Validity

Object variables:

PERT/CPM-Task[k] =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k]

PERT/CPM-Task[k].Init_Plan_Resource[m][t] =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k].plan resource profile[m][t]

PERT/CPM-Resource[m] =
PERT/CPM model.Initial PERT/CPM plan.Resources[m]

SD-Task[i] =
SD project model.SD model architecture.SD-Tasks[i]

SD-Task[i].Init_Plan_Resource[r][t] =
SD project model.SD plan.Inital plan.Resources.Task resources[i,r][t]

SD-Resource[r] =
SD project model.SD model architecture.SD Resources[r]

DCOI-5: SD-Task Initially Planned Budget Breakdown

Definition:

For each SD-Task[i] in the SD model, the initially planned budget for each work activity must be the same as sum of the initial budgets of all PERT/CPM tasks mapped to that SD-Task which are of the same work type as the activity.

Usage: transfer initial project plan from one model to the other

Formal specification:

DCOI-5(PERT/CPM Model,SD Model) =
Relationships =
Budget =
SD-Task[i].Init_Budget_Activity [j] ==
SUM {PERT/CPM-Task[k].Init_Budget}
Validity =
SC-WBS(PERT/CPM Model.Initial PERT/CPM plan, SD Model).Operators.
Is_mapped(PERT/CPM-Task[k], SD-Task[i])
AND
[*PERT/CPM model.Operators.Is_Type(PERT/CPM-Task[k].SD-Activity[j])*]

where:

- SUM is a function that sums the values in a set

Short references:

DCOI-5.PERT-SD = DCOI-5().Relationships
DCOI-5.Validity = DCOI-5().Validity

Object variables:

PERT/CPM-Task[k] =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k]

PERT/CPM-Task[k].Init_Budget =
PERT/CPM model.Initial PERT/CPM plan.Tasks[k].planned budget

SD-Task[i] =
SD project model.SD model architecture.SD-Tasks[i]

SD-Task[i].Init_Budget_Activity[j] =
SD project model.SD plan.Initial plan.Budget.Task Budget[i,j]

DCOI-6: Actual and Currently Planned Profiles of Resources Availability

Definition:

For those profiles of resource availability defined in the PERT/CPM model at the same level of aggregation as in the SD model, and in both models these profiles also incorporate the same elementary PERT/CPM resources, then the actual and the currently planned profiles must be the same in the two models.

Usage: transfer project status or new project plan from SD model to PERT/CPM model

Formal specification:

DCOI-6(PERT/CPM model, SD project model) =
  Relationships =
    Past =
    PERT/CPM.Actual_Res_Avai_Past[r_i][t] ~=
    SD.Actual_Res_Avai_Past[r_i][t]
    Future =
    PERT/CPM.Plan_Res_Avai_Future[r_i][t] ~=
    SD.Plan_Res_Avai_Future[r_i][t]

Where:

- r_i = SD-Resource[i]

Short references:

DCOI-6.1.SD-PERT = DCOI-6().Relationships.Future
DCOI-6.2.SD-PERT = DCOI-6().Relationships.Past

Object variables:

PERT/CPM.Actual_Res_Avail_Past[r_i][t] =
PERT/CPM model.Current PERT/CPM plan.Past segment.Actual_Resource-availability[r_i][t]

PERT/CPM.Plan_Res_Avail_Future[r_i][t] =
PERT/CPM model.Current PERT/CPM plan.Future segment.Plan_Resource-availability[r_i][t]

SD.Actual_Res_Avai_Past[r_i][t] =
SD project model.Project Behaviour.Past Segment.Resources.Project Resources[r_i][t]

SD.Plan_Res_Avai_Future[r_i][t] =
SD project model.Project Behaviour.Future Segment.Resources.Project Resources[r_i][t]

DCO Links

DCO-1: SD-Task and Project Actual and Currently Planned Schedules

Definition:

For each SD-Task[i] in the SD model the currently planned start and finishing dates over-time must be the same in the two models for both past and future segments of the project. For the whole project, the currently planned finishing date must be the same in the two models.

Usage: transfer project status or new project plan from one model to the other

Formal specification:

```
DCO-1(PERT/CPM model, SD project model) =
  Relationships =
     Future =
       Task_Start_Date =
              SD-Task[i].Plan_Start_Date_Future[t] ~=
              PERT/CPM.SD-Task[i].Plan_Start_Date_Future[t]
       Task_Finish_Date =
              SD-Task[i].Plan_Finish_Date_Future[t] ~=
              PERT/CPM.SD-Task[i].Plan_Finish_Date_Future[t]
       Project_Finish_Date =
              SD-Project.Plan_Finish_Date_Future[t] ~=
              PERT/CPM.Project.Plan_Finish_Date_Future[t]
     Past =
       Task_Start_Date =
              SD-Task[i].Plan_Start_Date_Past[t] ~=
              PERT/CPM.SD-Task[i].Plan_Start_Date_Past[t]
       Task_Finish_Date =
              SD-Task[i].Plan_Finish_Date_Past[t] ~=
              PERT/CPM.SD-Task[i].Finish_Date[t]
       Project_Finish_Date =
              SD-Project.Plan_Finish_Date_Past[t] ~=
              PERT/CPM.Project.Plan_Finish_Date_Past[t]
```

Short references:

DCO-1.1 = DCO-1().Relationships.Future
DCO-1.2 = DCO-1().Relationships.Past

Object variables:

PERT/CPM.SD-Task[i].Plan_Start_Date_Future[t]
PERT/CPM model.Project behaviour.Future segment.Schedule.Task Schedule.Start date[i][t]

PERT/CPM.SD-Task[i].Plan_Finish_Date_Future[t]

PERT/CPM model.Project behaviour.Future segment.Schedule.Task Schedule.Finish date[i][t]

PERT/CPM.Project.Plan_Finish_Date_Future[t]
PERT/CPMmodel.Project behaviour.Future segment.Schedule.Project Schedule.Finish date[i][t]

PERT/CPM.SD-Task[i].Plan_Start_Date_Past[t]
PERT/CPM model.Project behaviour.Past segment.Schedule.Task Schedule.Start date[i][t]

PERT/CPM.SD-Task[i].Plan_Finish_Date_Past[t]
PERT/CPM model.Project behaviour.Past segment.Schedule.Task Schedule.Finish date[i][t]

PERT/CPM.Project.Plan_Finish_Date_Past[t]
PERT/CPMmodel.Project behaviour.Past segment.Schedule.Project Schedule.Finish date[i][t]

SD-Task[i].Plan_Start_Date_Future[t] =
SD project model.Project behaviour.Future segment.Schedule.Task Schedule.Start date[i][t]

SD-Task[i].Plan_Finish_Date_Future[t] =
SD project model.Project behaviour.Future segment.Schedule.Task Schedule.Finish date[i][t]

SD-Project[i].Plan_Finish_Date_Future[t] =
SD project model.Project behaviour.Future segment.Schedule.Project Schedule.Finish date[i][t]

SD-Task[i].Plan_Start_Date_Past[t] =
SD project model.Project behaviour.Past segment.Schedule.Task Schedule.Start date[i][t]

SD-Task[i].Plan_Finish_Date_Past[t] =
SD project model.Project behaviour.Past segment.Schedule.Task Schedule.Finish date[i][t]

SD-Project[i].plan_Finish_Date_Past[t] =
SD project model.Project behaviour.Past segment.Schedule.Project Schedule.Finish date[i][t]

DCO-2: SD-Task and Project Actual and Currently Planned Effort .

Definition:

For each SD-Task[i] in the SD model and for the whole project the following effort indices over-time must be the same: ACWP, ACWP-Eng, ACWP-Man, BCWP, BCWS, CTC and CAC; for both past and future segments of the project.

Usage: transfer project status or new project plan from one model to the other

Formal specification:

```
DCO-2(PERT/CPM model, SD project model) =
  Relationships =
    Future =
        ACWP =
            SD-Task[i] ACWP_Future[t] ~= PERT/CPM.SD-Task[i].ACWP_Future[t]
            SD-Task[i].ACWP-Eng_Future[t] ~=
                PERT/CPM.SD-Task[i].ACWP-Eng_Future[t]
            SD-Task[i].ACWP-Man_Future[t] ~=
                PERT/CPM.SD-Task[i].ACWP-Man_Future[t]
            SD-Project.ACWP_Future[t] ~= PERT/CPM.Project.ACWP_Future[t]
            SD-Project.ACWP-Eng_Future[t] ~=
                PERT/CPM.Project.ACWP-Eng_Future[t]
            SD-Project.ACWP-Man_Future[t] ~=
                PERT/CPM.Project.ACWP-Man_Future[t]
        BCWP =
            SD-Task[i].BCWP_Future[t] ~= PERT/CPM.SD-Task[i].BCWP_Future[t]
            SD-Project.BCWP_Future[t] ~= PERT/CPM.Project.BCWP_Future[t]
        BCWS =
            SD-Task[i].BCWS_Future[t] ~= PERT/CPM.SD-Task[i].BCWS_Future[t]
            SD-Project.BCWS_Future[t] ~= PERT/CPM.Project.BCWS_Future[t]
        CTC =
            SD-Task[i].CTC_Future[t] ~= PERT/CPM.SD-Task[i].CTC_Future[t]
            SD-Project.CTC_Future[t] ~= PERT/CPM.Project.CTC_Future[t]
        CAC =
            SD-Task[i].CAC_Future[t] ~= PERT/CPM.SD-Task[i].CAC_Future[t]
            SD-Project.CAC_Future[t] ~= PERT/CPM.Project.CAC_Future[t]
    Past =
        ACWP =
            SD-Task[i].ACWP_Past[t] ~= PERT/CPM.SD-Task[i].ACWP_Past[t]
            SD-Task[i].ACWP-Eng_Past[t] ~=
                PERT/CPM.SD-Task[i].ACWP-Eng_Past[t]
            SD-Task[i].ACWP-Man_Past[t] ~=
                PERT/CPM.SD-Task[i].ACWP-Man_Past[t]
            SD-Project.ACWP_Past[t] ~= PERT/CPM.Project.ACWP_Past[t]
            SD-Project.ACWP-Eng_Past[t] ~=
                PERT/CPM.Project.ACWP-Eng_Past[t]
            SD-Project.ACWP-Man_Past[t] ~=
                PERT/CPM.Project.ACWP-Man_Past[t]
        BCWP =
            SD-Task[i].BCWP_Past[t] ~= PERT/CPM.SD-Task[i].BCWP_Past[t]
            SD-Project.BCWP_Past[t] ~= PERT/CPM.Project.BCWP_Past[t]
        BCWS =
            SD-Task[i].BCWS_Past[t] ~= PERT/CPM.SD-Task[i].BCWS_Past[t]
            SD-Project.BCWS_Past[t] ~= PERT/CPM.Project.BCWS_Past[t]
```

CTC =
SD-Task[i].CTC_Past[t] ~= PERT/CPM.SD-Task[i].CTC_Past[t]
SD-Project.CTC_Past[t] ~= PERT/CPM.Project.CTC_Past[t]
CAC =
SD-Task[i].CAC_Past[t] ~= PERT/CPM.SD-Task[i].CAC_Past[t]
SD-Project.CAC_Past[t] ~= PERT/CPM.Project.CAC_Past[t]

Short references:

DCO-2.1 = DCO-2().Relationships.Future
DCO-2.2 = DCO-2().Relationships.Past

Object variables:

SD-Task[i].ACWP_Future[t] =
SD project model.Project behaviour.Future segment.Effort.Task Effort.ACWP[i][t]

SD-Task[i].ACWP-Eng_Future[t] =
SD project model.Project behaviour.Future segment.Effort.Task Effort.ACWP Engineering[i][t]

SD-Task[i].ACWP-Man_Future[t] =
SD project model.Project behaviour.Future segment.Effort.Task Effort.ACWP Management[i][t]

SD-Task[i].ACWP_Past[t] =
SD project model.Project behaviour.Past segment.Effort.Task Effort.ACWP[i][t]

SD-Task[i].ACWP-Eng_Past[t] =
SD project model.Project behaviour.Past segment.Effort.Task Effort.ACWP Engineering[i][t]

SD-Task[i].ACWP-Man_Past[t] =
SD project model.Project behaviour.Past segment.Effort.Task Effort.ACWP Management[i][t]

SD-Project.ACWP[t] =
SD project model.Project behaviour.Future segment.Effort.Project Effort.ACWP[i][t]

SD-Task[i].BCWP_Future[t] =
SD project model.Project behaviour.Future segment.Effort.Task Effort.BCWP[i][t]

SD-Task[i].BCWP_Past[t] =
SD project model.Project behaviour.Past segment.Effort.Task Effort.BCWP[i][t]

SD-Project.BCWP[t] =
SD project model.Project behaviour.Future segment.Effort.Project Effort.BCWP[i][t]

SD-Task[i].BCWS_Future[t] =
SD project model.Project behaviour.Future segment.Effort.Task Effort.BCWS[i][t]

SD-Task[i].BCWS_Past[t] =
SD project model.Project behaviour.Past segment.Effort.Task Effort.BCWS[i][t]

SD-Project.BCWS[t] =

SD project model.Project behaviour.Future segment.Effort.Project Effort.BCWS[i][t]

SD-Task[i].CTC_Future[t] =
SD project model.Project behaviour.Future segment.Effort.Task Effort.CTC[i][t]

SD-Task[i].CTC_Past[t] =
SD project model.Project behaviour.Past segment.Effort.Task Effort.CTC[i][t]

SD-Project.CTC[t] =
SD project model.Project behaviour.Future segment.Effort.Project Effort.CTC[i][t]

SD-Task[i].CAC_Future[t] =
SD project model.Project behaviour.Future segment.Effort.Task Effort.CAC[i][t]

SD-Task[i].CAC_Past[t] =
SD project model.Project behaviour.Past segment.Effort.Task Effort.CAC[i][t]

SD-Project.CAC[t] =
SD project model.Project behaviour.Future segment.Effort.Project Effort.CAC[i][t]

PERT/CPM.SD-Task[i].ACWP_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Effort.Task Effort.ACWP[i][t]

PERT/CPM.SD-Task[i].ACWP-Eng_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Effort.Task Effort.ACWP Engineering[i][t]

PERT/CPM.SD-Task[i].ACWP-Man_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Effort.Task Effort.ACWP Management[i][t]

PERT/CPM.SD-Task[i].ACWP_Past[t] =
PERT/CPM model.Project behaviour.Past segment.Effort.Task Effort.ACWP[i][t]

PERT/CPM.SD-Task[i].ACWP-Eng_Past[t] =
PERT/CPM model.Project behaviour.Past segment.Effort.Task Effort.ACWP Engineering[i][t]

PERT/CPM.SD-Task[i].ACWP-Man_Past[t] =
PERT/CPM model.Project behaviour.Past segment.Effort.Task Effort.ACWP Management[i][t]

PERT/CPM.Project.ACWP[t] =
PERT/CPM model.Project behaviour.Future segment.Effort.Project Effort.ACWP[i][t]

PERT/CPM.SD-Task[i].BCWP_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Effort.Task Effort.BCWP[i][t]

PERT/CPM.SD-Task[i].BCWP_Past[t] =
PERT/CPM model.Project behaviour.Past segment.Effort.Task Effort.BCWP[i][t]

PERT/CPM.Project.BCWP[t] =
PERT/CPM model.Project behaviour.Future segment.Effort.Project Effort.BCWP[i][t]

PERT/CPM.SD-Task[i].BCWS_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Effort.Task Effort.BCWS[i][t]

PERT/CPM.SD-Task[i].BCWS_Past[t] =
PERT/CPM model.Project behaviour.Past segment.Effort.Task Effort.BCWS[i][t]

PERT/CPM.Project.BCWS[t] =
PERT/CPM model.Project behaviour.Future segment.Effort.Project Effort.BCWS[i][t]

PERT/CPM.SD-Task[i].CTC_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Effort.Task Effort.CTC[i][t]

PERT/CPM.SD-Task[i].CTC_Past[t] =
PERT/CPM model.Project behaviour.Past segment.Effort.Task Effort.CTC[i][t]

PERT/CPM.Project.CTC[t] =
PERT/CPM model.Project behaviour.Future segment.Effort.Project Effort.CTC[i][t]

PERT/CPM.SD-Task[i].CAC_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Effort.Task Effort.CAC[i][t]

PERT/CPM.SD-Task[i].CAC_Past[t] =
PERT/CPM model.Project behaviour.Past segment.Effort.Task Effort.CAC[i][t]

PERT/CPM.Project.CAC[t] =
PERT/CPM model.Project behaviour.Future segment.Effort.Project Effort.CAC[i][t]

DCO-3: SD-Task and Project Actual and Currently Planned Resources

Definition:

For each SD-Task[i] in the SD model and for the whole project the following resource indices over-time must be the same: ASP, PSP, CASP, CPSP and CSPAC; for both past and future segments of the project.

Usage: transfer project status or new project plan from one model to the other

Formal specification:

```
DCO-3(PERT/CPM model, SD project model) =
  Relationships =
    Future =
        ASP =
              SD-Task[i] ASP_Future[t] ~= PERT/CPM.SD-Task[i].ASP_Future[t]
              SD-Project.ASP_Future[t] ~= PERT/CPM.Project.ASP_Future[t]
        PSP =
              SD-Task[i].PSP_Future[t] ~= PERT/CPM.SD-Task[i].PSP_Future[t]
              SD-Project.PSP_Future[t] ~= PERT/CPM.Project.PSP_Future[t]
        CASP =
              SD-Task[i].CASP_Future[t] ~= PERT/CPM.SD-Task[i].CASP_Future[t]
              SD-Project.CASP_Future[t] ~= PERT/CPM.Project.CASP_Future[t]
        CPSP =
              SD-Task[i].CPSP_Future[t] ~= PERT/CPM.SD-Task[i].CPSP_Future[t]
              SD-Project.CPSP_Future[t] ~= PERT/CPM.Project.CPSP_Future[t]
        CSPA =
              SD-Task[i].CSPA_Future[t] ~= PERT/CPM.SD-Task[i].CSPA_Future[t]
              SD-Project.CSPA_Future[t] ~= PERT/CPM.Project.CSPA_Future[t]
    Past =
        ASP =
              SD-Task[i].ASP_Past[t] ~= PERT/CPM.SD-Task[i].ASP_Past[t]
              SD-Project.ASP_Past[t] ~= PERT/CPM.Project.ASP_Past[t]
        PSP =
              SD-Task[i].PSP_Past[t] ~= PERT/CPM.SD-Task[i].PSP_Past[t]
              SD-Project.PSP_Past[t] ~= PERT/CPM.Project.PSP_Past[t]
        CASP =
              SD-Task[i].CASP_Past[t] ~= PERT/CPM.SD-Task[i].CASP_Past[t]
              SD-Project.CASP_Past[t] ~= PERT/CPM.Project.CASP_Past[t]
        CPSP =
              SD-Task[i].CPSP_Past[t] ~= PERT/CPM.SD-Task[i].CPSP_Past[t]
              SD-Project.CPSP_Past[t] ~= PERT/CPM.Project.CPSP_Past[t]
        CSPA =
              SD-Task[i].CSPA_Past[t] ~= PERT/CPM.SD-Task[i].CSPA_Past[t]
              SD-Project.CSPA_Past[t] ~= PERT/CPM.Project.CSPA_Past[t]
```

Short references:

DCO-3.1 = DCO-3().Relationships.Future
DCO-3.2 = DCO-3().Relationships.Past

Object variables:

SD-Task[i].ASP_Future[t] =
SD project model.Project behaviour.Future segment.Resources.Task Resources.ASP[i][t]

SD-Task[i].ASP_Past[t] =
SD project model.Project behaviour.Past segment.Resources.Task Resources.ASP[i][t]

SD-Project.ASP[t] =
SD project model.Project behaviour.Future segment.Resources.Project Resources.ASP[i][t]

SD-Task[i].PSP_Future[t] =
SD project model.Project behaviour.Future segment.Resources.Task Resources.PSP[i][t]

SD-Task[i].PSP_Past[t] =
SD project model.Project behaviour.Past segment.Resources.Task Resources.PSP[i][t]

SD-Project.PSP[t] =
SD project model.Project behaviour.Future segment.Resources.Project Resources.PSP[i][t]

SD-Task[i].CASP_Future[t] =
SD project model.Project behaviour.Future segment.Resources.Task Resources.CASP[i][t]

SD-Task[i].CASP_Past[t] =
SD project model.Project behaviour.Past segment.Resources.Task Resources.CASP[i][t]

SD-Project.CASP[t] =
SD project model.Project behaviour.Future segment.Resources.Project Resources.CASP[i][t]

SD-Task[i].CPSP_Future[t] =
SD project model.Project behaviour.Future segment.Resources.Task Resources.CPSP[i][t]

SD-Task[i].CPSP_Past[t] =
SD project model.Project behaviour.Past segment.Resources.Task Resources.CPSP[i][t]

SD-Project.CPSP[t] =
SD project model.Project behaviour.Future segment.Resources.Project Resources.CPSP[i][t]

SD-Task[i].CSPA_Future[t] =
SD project model.Project behaviour.Future segment.Resources.Task Resources.CSPA[i][t]

SD-Task[i].CSPA_Past[t] =
SD project model.Project behaviour.Past segment.Resources.Task Resources.CSPA[i][t]

SD-Project.CSPA[t] =
SD project model.Project behaviour.Future segment.Resources.Project Resources.CSPA[i][t]

PERT/CPM.SD-Task[i].ASP_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Resources.Task Resources.ASP[i][t]

PERT/CPM.SD-Task[i].ASP_Past[t] =
PERT/CPM model.Project behaviour.Past segment.Resources.Task Resources.ASP[i][t]

PERT/CPM.Project.ASP[t] =
PERT/CPM model.Project behaviour.Future segment.Resources.Project Resources.ASP[i][t]

PERT/CPM.SD-Task[i].PSP_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Resources.Task Resources.PSP[i][t]

PERT/CPM.SD-Task[i].PSP_Past[t] =
PERT/CPM model.Project behaviour.Past segment.Resources.Task Resources.PSP[i][t]

PERT/CPM.Project.PSP[t] =
PERT/CPM model.Project behaviour.Future segment.Resources.Project Resources.PSP[i][t]

PERT/CPM.SD-Task[i].CASP_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Resources.Task Resources.CASP[i][t]

PERT/CPM.SD-Task[i].CASP_Past[t] =
PERT/CPM model.Project behaviour.Past segment.Resources.Task Resources.CASP[i][t]

PERT/CPM.Project.CASP[t] =
PERT/CPM model.Project behaviour.Future segment.Resources.Project Resources.CASP[i][t]

PERT/CPM.SD-Task[i].CPSP_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Resources.Task Resources.CPSP[i][t]

PERT/CPM.SD-Task[i].CPSP_Past[t] =
PERT/CPM model.Project behaviour.Past segment.Resources.Task Resources.CPSP[i][t]

PERT/CPM.Project.CPSP[t] =
PERT/CPM model.Project behaviour.Future segment.Resources.Project Resources.CPSP[i][t]

PERT/CPM.SD-Task[i].CSPA_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Resources.Task Resources.CSPA[i][t]

PERT/CPM.SD-Task[i].CSPA_Past[t] =
PERT/CPM model.Project behaviour.Past segment.Resources.Task Resources.CSPA[i][t]

PERT/CPM.Project.CSPA[t] =
PERT/CPM model.Project behaviour.Future segment.Resources.Project Resources.CSPA[i][t]

DCO-4: SD-Task and Project Actual and Currently Planned Scope.

Definition:

For each SD-Task[i] in the SD model and for the whole project the following resource indices over-time must be the same: SCAC, CSCC, ASCWP, and SCTC; for both past and future segments of the project.

Usage: transfer project status or new project plan from one model to the other

Formal specification:

```
DCO-4(PERT/CPM model, SD project model) =
  Relationships =
    Future =
        SCAC =
              SD-Task[i].SCAC_Future[t] ~= PERT/CPM.SD-Task[i].SCAC_Future[t]
              SD-Project.SCAC_Future[t] ~= PERT/CPM.Project.SCAC_Future[t]
        CSCC =
              SD-Task[i].CSCC_Future[t] ~= PERT/CPM.SD-Task[i].CSCC_Future[t]
              SD-Project.CSCC_Future[t] ~= PERT/CPM.Project.CSCC_Future[t]
        ASCWP =
              SD-Task[i].ASCWP_Future[t] ~=
                PERT/CPM.SD-Task[i].ASCWP_Future[t]
              SD-Project.ASCWP_Future[t] ~= PERT/CPM.Project.ASCWP_Future[t]
        SCTC =
              SD-Task[i].SCTC_Future[t] ~= PERT/CPM.SD-Task[i].SCTC_Future[t]
              SD-Project.SCTC_Future[t] ~= PERT/CPM.Project.SCTC_Future[t]
    Past =
        SCAC =
              SD-Task[i].SCAC_Past[t] ~= PERT/CPM.SD-Task[i].SCAC_Past[t]
              SD-Project.SCAC_Past[t] ~= PERT/CPM.Project.SCAC_Past[t]
        CSCC =
              SD-Task[i].CSCC_Past[t] ~= PERT/CPM.SD-Task[i].CSCC_Past[t]
              SD-Project.CSCC_Past[t] ~= PERT/CPM.Project.CSCC_Past[t]
        ASCWP =
              SD-Task[i].ASCWP_Past[t] ~= PERT/CPM.SD-Task[i].ASCWP_Past[t]
              SD-Project.ASCWP_Past[t] ~= PERT/CPM.Project.ASCWP_Past[t]
        SCTC =
              SD-Task[i].SCTC_Past[t] ~= PERT/CPM.SD-Task[i].SCTC_Past[t]
              SD-Project.SCTC_Past[t] ~= PERT/CPM.Project.SCTC_Past[t]
```

Short references:

DCO-4.1 = DCO-4().Relationships.Future
DCO-4.2 = DCO-4().Relationships.Past

Object variables:

SD-Task[i].SCAC_Future[t] =
SD project model.Project behaviour.Future segment.Scope.Task Scope.SCAC[i][t]

SD-Task[i].SCAC_Past[t] =
SD project model.Project behaviour.Past segment.Scope.Task Scope.SCAC[i][t]

SD-Project.SCAC[t] =
SD project model.Project behaviour.Future segment.Scope.Project Scope.SCAC[i][t]

SD-Task[i].CSCC_Future[t] =
SD project model.Project behaviour.Future segment.Scope.Task Scope.CSCC[i][t]

SD-Task[i].CSCC_Past[t] =
SD project model.Project behaviour.Past segment.Scope.Task Scope.CSCC[i][t]

SD-Project.CSCC[t] =
SD project model.Project behaviour.Future segment.Scope.Project Scope.CSCC[i][t]

SD-Task[i].ASCWP_Future[t] =
SD project model.Project behaviour.Future segment.Scope.Task Scope.ASCWP[i][t]

SD-Task[i].ASCWP_Past[t] =
SD project model.Project behaviour.Past segment.Scope.Task Scope.ASCWP[i][t]

SD-Project.ASCWP[t] =
SD project model.Project behaviour.Future segment.Scope.Project Scope.ASCWP[i][t]

SD-Task[i].SCTC_Future[t] =
SD project model.Project behaviour.Future segment.Scope.Task Scope.SCTC[i][t]

SD-Task[i].SCTC_Past[t] =
SD project model.Project behaviour.Past segment.Scope.Task Scope.SCTC[i][t]

SD-Project.SCTC[t] =
SD project model.Project behaviour.Future segment.Scope.Project Scope.SCTC[i][t]

PERT/CPM.SD-Task[i].SCAC_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Scope.Task Scope.SCAC[i][t]

PERT/CPM.SD-Task[i].SCAC_Past[t] =
PERT/CPM model.Project behaviour.Past segment.Scope.Task Scope.SCAC[i][t]

PERT/CPM.Project.SCAC[t] =
PERT/CPM model.Project behaviour.Future segment.Scope.Project Scope.SCAC[i][t]

PERT/CPM.SD-Task[i].CSCC_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Scope.Task Scope.CSCC[i][t]

PERT/CPM.SD-Task[i].CSCC_Past[t] =
PERT/CPM model.Project behaviour.Past segment.Scope.Task Scope.CSCC[i][t]

PERT/CPM.Project.CSCC[t] =
PERT/CPM model.Project behaviour.Future segment.Scope.Project Scope.CSCC[i][t]

PERT/CPM.SD-Task[i].ASCWP_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Scope.Task Scope.ASCWP[i][t]

PERT/CPM.SD-Task[i].ASCWP_Past[t] =
PERT/CPM model.Project behaviour.Past segment.Scope.Task Scope.ASCWP[i][t]

PERT/CPM.Project.ASCWP[t] =
PERT/CPMmodel.Project behaviour.Future segment.Scope.Project Scope.ASCWP[i][t]

PERT/CPM.SD-Task[i].SCTC_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Scope.Task Scope.SCTC[i][t]

PERT/CPM.SD-Task[i].SCTC_Past[t] =
PERT/CPM model.Project behaviour.Past segment.Scope.Task Scope.SCTC[i][t]

PERT/CPM.Project.SCTC[t] =
PERT/CPM model.Project behaviour.Future segment.Scope.Project Scope.SCTC[i][t]

D.3 – Data-structural links

Data-structural readjustment

DSR-1: Derivation of intra-task dependencies of SD-Task

Definition:

For each SD-Task[i] in the SD model, the progress curve of its intra-task dependency can be derived from the planned cumulative scope accomplishment taken from the PERT7CPM model. The progress curve is derived based on the assumption that the work is being compressed within the task up to the limit of technical feasibility.

Usage: transfer initial plan or new project plan from PERT/CPM model to SD model

Formal specification:

DSR-1(PERT/CPM Model,SD Model) =
Relationships =
ASCWP%[t] := PERT/CPM.SD-Task[i].ASCWP_Future[t] /
PERT/CPM.SD-Task[i].SCAC_Future[t_0]
Scope_That_Can_Start[t] :=
IF
ASCWP%[t] == 0 THEN ASCWP%[DELAY * UT]
ELSE
ASCWP%[t + DELAY * UT]
SD-Task[i].Intra-Dependency[x,y] :=
{(x, Scope_That_Can_Start[t_i]) : ASCWP%[t_i] == x}

where:

- UT is the time unit considered within the SD model
- DELAY is the average work processing delay within the task

Short references:

DSR-1.PERT-SD = DSR-1().Relationships

Object variables:

SD-Task[i].Intra-Dependency[x,y] =
SD project model.SD model architecture.SD-Tasks[i].Intra-task-curve[x,y]

PERT/CPM.SD-Task[i].ASCWP_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Scope.Task Scope.ASCWP[i][t]

PERT/CPM.SD-Task[i].SCAC_Future[t_0]
PERT/CPM model.Project behaviour.Future segment.Scope.Project Scope.SCAC[i][t_0]

DSR-2: Derivation of inter-task dependencies of SD-Task

Definition:

For each SD-Dependency[k] in the SD model linking the SD-Task[i] (predecessor) and SD-Task[j] (successor), its progress curve can be derived from the planned cumulative scope accomplishment of the these two SD-Tasks taken from the PERT7CPM model. The progress curve is derived based on the assumption that the tasks are being overlapped up to the limit of technical feasibility.

Usage: transfer initial plan or new project plan from PERT/CPM model to SD model

Formal specification:

DSR-2(PERT/CPM Model,SD Model) =
Relationships =
ASCWP_P%[t] := PERT/CPM.SD-Task[i].ASCWP_Future[t] /
PERT/CPM.SD-Task[i].SCAC_Future[t_0]
ASCWP_S%[t] := PERT/CPM.SD-Task[j].ASCWP_Future[t] /
PERT/CPM.SD-Task[j].SCAC_Future[t_0]
Scope_That_Can_Start[t] :=
IF
ASCWP_P%[t] == 0 THEN 0
ELSE
ASCWP_S%[t + DELAY * UT]
SD.Inter-Task-Dependency[k][x,y] :=
{(x, Scope_That_Can_Start[t_i]) : ASCWP_P%[t_i] == x}

where:

- UT is the time unit considered within the SD model
- DELAY is the average work processing delay within the successor task

Short references:

DSR-2.PERT-SD = DSR-2().Relationships

Object variables:

SD.Inter-Task-Dependency[k][x,y] =
SD project model.SD model architecture.SD-Inter-Task-Dependencies[k].Progress-curve[x,y]

PERT/CPM.SD-Task[i].ASCWP_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Scope.Task Scope.ASCWP[i][t]

PERT/CPM.SD-Task[i].SCAC_Future[t_0]
PERT/CPM model.Project behaviour.Future segment.Scope.Project Scope.SCAC[i][t_0]

Data-structural consistency

DSC-1: Derivation of intra-task dependencies of SD-Task

Definition:

For each SD-Task[i] in the SD model, the progress curve of its intra-task dependency must be consistent with the planned cumulative scope accomplishment taken from the PERT/CPM model. It is assumed that the work is being compressed within the task up to the limit of technical feasibility.

Usage: transfer initial plan or new project plan from PERT/CPM model to SD model

Formal specification:

DSC-1(PERT/CPM Model,SD Model) =
Relationships =
ASCWP%[t] := PERT/CPM.SD-Task[i].ASCWP_Future[t] /
PERT/CPM.SD-Task[i].SCAC_Future[t_0]
Scope_That_Can_Start[t] :=
IF
ASCWP%[t] == 0 THEN ASCWP%[DELAY * UT]
ELSE
ASCWP%[t + DELAY * UT]
SD-Task[i].Intra-Dependency[x,y] ==
{(x, Scope_That_Can_Start[t_i]) : ASCWP%[t_i] == x}

where:

- UT is the time unit considered within the SD model
- DELAY is the average work processing delay within the task

Short references:

DSC-1 = DSC-1().Relationships

Object variables:

SD-Task[i].Intra-Dependency[x,y] =
SD project model.SD model architecture.SD-Tasks[i].Intra-task-curve[x,y]

PERT/CPM.SD-Task[i].ASCWP_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Scope.Task Scope.ASCWP[i][t]

PERT/CPM.SD-Task[i].SCAC_Future[t_0]
PERT/CPM model.Project behaviour.Future segment.Scope.Project Scope.SCAC[i][t_0]

DSC-2: Derivation of inter-task dependencies of SD-Task

Definition:

For each SD-Dependency[k] in the SD model linking the SD-Task[i] (predecessor) and SD-Task[j] (successor), its progress curve must be consistent with the planned cumulative scope accomplishment of the these two SD-Tasks taken from the PERT7CPM model. It is assumed that the tasks are being overlapped up to the limit of technical feasibility.

Usage: transfer initial plan or new project plan from PERT/CPM model to SD model

Formal specification:

DSC-2(PERT/CPM Model,SD Model) =
Relationships =
ASCWP_P%[t] := PERT/CPM.SD-Task[i].ASCWP_Future[t] /
PERT/CPM.SD-Task[i].SCAC_Future[t_0]
ASCWP_S%[t] := PERT/CPM.SD-Task[j].ASCWP_Future[t] /
PERT/CPM.SD-Task[j].SCAC_Future[t_0]
Scope_That_Can_Start[t] :=
IF
ASCWP_P%[t] == 0 THEN 0
ELSE
ASCWP_S%[t + DELAY * UT]
SD.Inter-Task-Dependency[k][x,y] :=
{(x, Scope_That_Can_Start[t_i]) : ASCWP_P%[t_i] == x}

where:

- UT is the time unit considered within the SD model
- DELAY is the average work processing delay within the successor task

Short references:

DSC-2 = DSC-2().Relationships

Object variables:

SD.Inter-Task-Dependency[k][x,y] =
SD project model.SD model architecture.SD-Inter-Task-Dependencies[k].Progress-curve[x,y]

PERT/CPM.SD-Task[i].ASCWP_Future[t] =
PERT/CPM model.Project behaviour.Future segment.Scope.Task Scope.ASCWP[i][t]

PERT/CPM.SD-Task[i].SCAC_Future[t_0]
PERT/CPM model.Project behaviour.Future segment.Scope.Project Scope.SCAC[i][t_0]

Appendix E – A personal view of the project management process

Overview

The author's own view of the project management process is here presented. Some of the ideas may not be found in the literature described in the same way. a linear and more static view is often adopted for pedagogical purposes. The main purpose of the description here presented is to provide a dynamic framework of the project management process, wherein continuous iteration, refinements, rework and interactions play a major role (as they do in the real world). This more dynamic perspective allows for an easier understanding of the use and integration of System Dynamics models, within the project management process.

Basic control mechanism

The simplest way to consider the project management process is as a generic control process. Just as a thermostat controls the temperature of a room based on the pre-specified target and on generating a reaction to deviations. The two main functions of the thermostat are (i) to determine whether or not the heater will produce heat in the next time interval, and (ii) monitor the temperature of the room and compare against the target. Likewise, a project also has pre-specified objectives of cost, quality, time and requirements. Whenever the project outcome deviates from these targets, management takes decisions in order to recover the project back to the targets.

As a control mechanism, project management also has two main functions: (1) (re)-planning and (2) monitoring. (Re)-planning specifies what work should be implemented by whom, in the following period of time (e.g. month). This is based on the perceived project status and on the current targets. The monitoring function assesses and develops a description of this project status. This simplistic view of project management addresses the core of the whole process in the real world. It identifies the three main types of activities that take place when a project is implemented: planning, implementation and monitoring. Planning and monitoring

are management functions, because they do not represent work directly involved in developing the product. On the other hand, implementing the work specified in the plan is primarily aimed at developing the product. This product development activity is part of an engineering process. The project implementation process can therefore be viewed as a dual process of management and engineering, where these two major sub-processes interact continuously through planning and monitoring. This is illustrated in figure E.1 below.

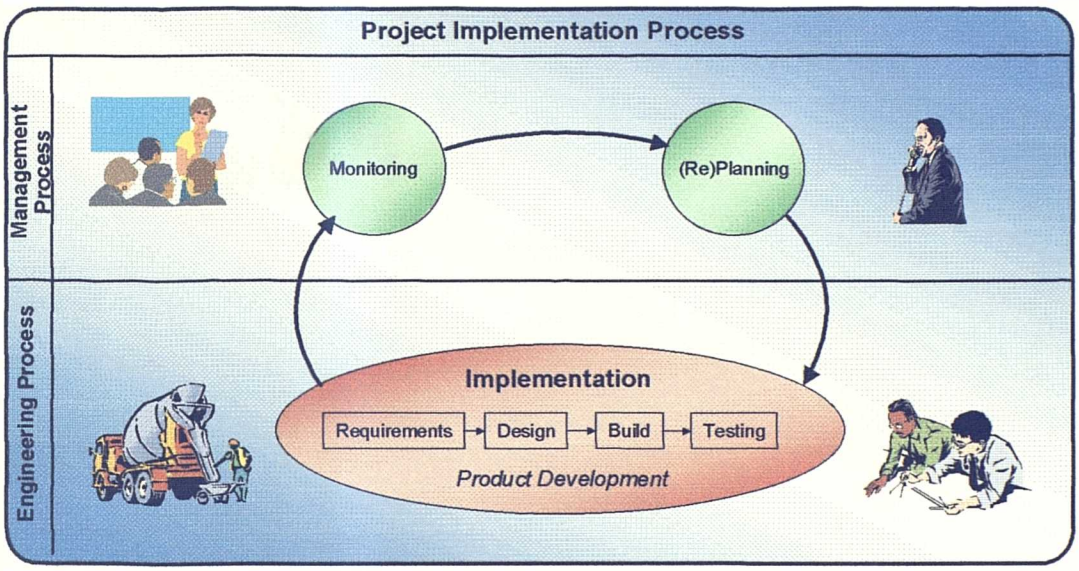


Figure E.1 – The project implementation process as a basic control mechanism

In practice, the project plan also includes the specification of management type of work (budgets, schedules and resources). As consequence, implementing or executing the plan also comprises carrying out the work of the monitoring and planning functions. In theory, this leads to a recursive process where a monitoring task monitors its own status. However, as it dos so, it changes its own status – the Heisenberg's principle of uncertainty applies! The same can be argued for planning – when you plan the planning action, you have already done it! For the sake of simplicity, and because in practice this recursive phenomenon is not relevant, it can be assumed that monitoring and planning focuses primarily on the product development work that takes place within the engineering process. The accomplishment of this work is assumed to be the key determinant of the project progress against the objectives. While management work can play a major part of the project cost, this takes the form of overheads imposed by the success of the product development work. Delays, requirements and quality problems may also

result from management work. However, these generally take the form of impacts over the product development work. Another possible approach is to consider that management is an important part of project execution. In this case, management performance is monitored itself and the management work is re-planned as appropriate (e.g. if decisions are taking too long to be produced, the management hierarchy may be changed). This often happens in the real world. In this case, the plan implementation activity should include explicitly management work as suggested by figure E.2 below.

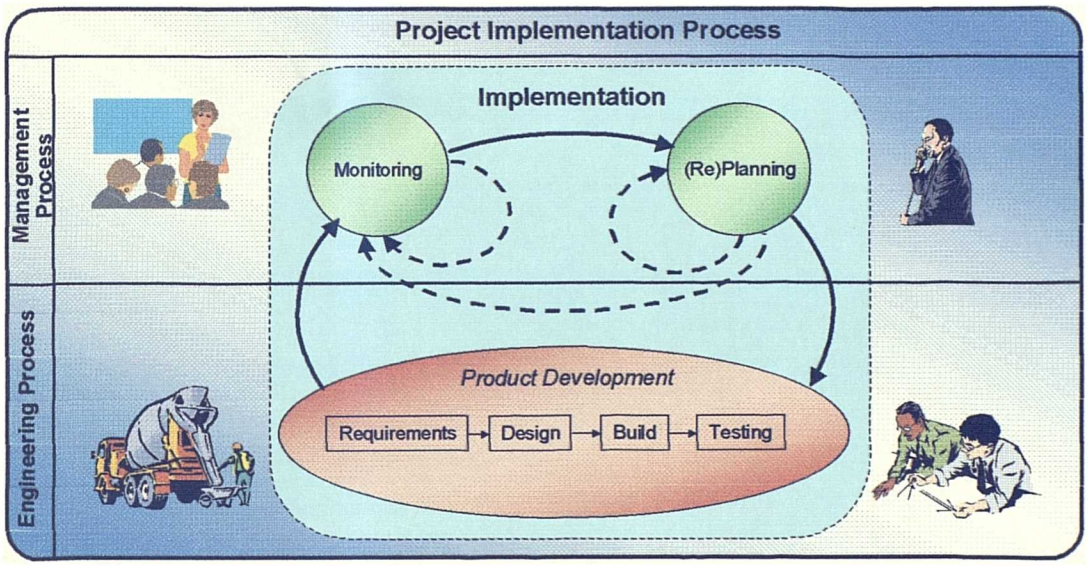


Figure E.2 – The project implementation process with management as part of plan implementation

In this scenario, the project work plan includes specifies the work required to carry out the management activities of monitoring and re-planning. The dotted arrows suggest that the monitoring function also assesses the state of the monitoring work as well as the planning work. Equally, the planning function also specifies how the planning and monitoring work should be carried out in the future. According to this view, the management process is assessed and re-structured as appropriate. In practice, this could take the form of monitoring how much effort has been spent in management work and, in face of over-runs, make changes to the management hierarchy and decision processes in order to save effort in the future.

While in reality the performance of management work can have a major impact on the project outcome, the product development process is the primary concern of

the project management control process. In general, it is the progress achieved in this process that determines mostly the overall progress in the project. Therefore, the simplified control cycle of figure E.1 is generally considered as the core project management process. In this research, this simplified view will be assumed in most situations. The more complete view of figure E.2 will be considered explicitly whenever appropriate. The most important aspect to retain is that project implementation comprises the two main processes of management and engineering, which interact continuously throughout the project life-cycle; as a control mechanism, the management process comprises the two main functions of monitoring and (re)planning; the engineering process comprises various product development activities according to the specific product development life-cycle.

Additional management functions

Although project management consists of a control mechanism, it includes more complex functions than just the basic ones of progress monitoring and work (re)planning, as shown in figures E.1 and E.2 Managing a project is far more complex than controlling the temperature of a room. The system being controlled is complex. Project are growing more and more complex (Williams 1997). Their social nature of project systems implies that a wide range of subjective human issues play a major role. Projects have a high degree of interaction with their surrounding environment which increases even more the complexity of the problems that need to be handled. As described in the previous sub-section, project are typically subjected to various and often conflicting interests of the different parties involved. In order to cope with the different types of issues that need to be controlled in a project, various individual functions can be considered, in addition to planning and monitoring. There is no standard set of such functions proposed in the body of knowledge. However, various typical functions can be identified in the literature (e.g. Nicholas 1990, Turner 1993, Kerzner 1998).

It is proposed in this research that a management function related to project control can be considered as an individual management sub-process with a well specified objective and aimed at handling a particular aspect of the project. Such management function should focus primarily on that particular aspect, over which it

will hold a primary responsibility above all other management functions. For example, human resource management can be considered as an individual management function. Its objective is to ensure that right resources will be available throughout the project life-cycle as required. It focuses primarily on staff recruitment and training. While other management functions may interact with this process, supporting it in various ways, human resource management has the primary responsibility over it. A management function will often be performed by a project sub-team, headed by a first-line manager (e.g. HRM manager). Without intending to be exhaustive, the following set of main management functions is here considered:

(1) *risk management* – aimed at identifying and evaluating risks (probability and impact), monitoring their occurrence, devising mitigating actions and controlling their implementation;

(2) change management – aimed at identifying major changes in the project which often imply a deep review of the project objectives and mission. It can focus on the product functional definition;

(3) *sub-contract management* – aimed at ensuring that sub-contracted work, product sub-components or intermediate sub-products is delivered on time, within budget, requirements and desired quality. It is based on the continuous monitoring of the sub-contractors' work. It is based on contract management. It may included procurement management;

(4) *client management* – aimed at handling all aspects related to the client behaviour and requirements. This includes keeping the client well informed about the progress in the project, continuous re-negotiation of the project objectives and other contractual conditions, continuous review and clarification of the product requirements and managing the client actions to prevent disruptive affects on the project;

(5) *quality management* – aimed at ensuring that all quality requirements for the project are satisfied. Typically, this is done through the implementation of a quality system, which is often certified according to certain standards (e.g. ISO s900 family of standards).

(6) configuration management – aimed at ensuring the control of the life-cycle of documents, sub-products and final product. These elements typically evolve throughout various versions version along the project life-cycle. Configuration

management ensures that this evolution is recorded for all elements and hence that all changes are traceable at any moment in time. Configuration management is generally a key requirement of a quality system;

(7) *resources management* – aimed at ensuring that al the required human and material resources are available throughout the project life-cycle as required. a separate human resource management function handles human resource requirements. Depending on the type of industry, other material resources, may also be critical to the project and hence may require an individual management function.

Depending on the type of organisational structure adopted for the project, these functions may be implemented by a dedicated individual management team or they may be implemented by existing departments in the organisation. In some cases, some management functions may be incorporated into a single function – e.g. configuration management can be included in quality management. Whether there is an explicit management function specified, the controlling implied in each of the functions generally needs to be implemented within the management process and by a certain function. Each of the seven functions listed above suggest a first-line manager in a "projectized" organisation: risk manager, change manager, sub-contracting manager, client manager, quality manager, configurations manager and (human) resources manager. The basic functions of monitoring and re-planning are generally implemented by the planning team headed by the planning manager.

These additional management functions have important interactions with the other two basic functions of monitoring and re-planning. The resources actually allocated to the project need to be monitored for comparison against the plan. All decisions undertaken within these functions have an impact and need to be considered against the project plan. For example, the implementation of a mitigating action requires that the project work plan is adjusted accordingly. These management functions may also interact directly with the product development process. For example, the quality system monitors closely the implementation of all the required quality standards. Configuration management imposes important rules to the development process (e.g. recording of all versions in the information system with the required data, like author, links to other documents, date, among others).

Another important function within the project management process, which does not have to do directly with product development, is the establishment and maintenance of an effective project management information system (PMIS) (Nicholas 1990). The aim of a PMIS is to ensure that all project information is available to the project team in a timely manner. The features offered by the PMIS is nowadays based on information technology, which may include simple relational databases to more sophisticated data mining and on-line analytical processing tools (OLAP) (Berson and Smith 1997).

A top-down perspective of initial planning

According to the author's opinion, the development of the initial project plan should follows a top-down approach. This is a key issue in project management. The project schedules and budgets are established first at higher levels of aggregation, and are then dis-aggregated into more detail, down to the operational level. The planning at the detailed level is therefore carried out with the primary aim of satisfying the schedule and budgets established at the aggregate level. This is opposed to planning first at the detailed level in order to find out the appropriate targets at the aggregate level. The latter bottom-up approach of the world is more aligned with the logical view of the world deployed by the classical sciences, like physics. This is also the way in which humans are generally educated to perceived the world – the macro-outcome results from the micro-events. As a consequence, the top-down approach may appear counter-intuitive: it is not the estimates at the bottom levels that determine the results at the top levels, but the other way around. However, this view has the great virtue of addressing the practical problems of project planning in the real world: commitments with the Client need to be made before effort has been spent in the development of a detailed plan; when high-level estimates are required in the early stages, it is not yet appropriate to impose planning decisions at the detailed level; this is because there is not enough information available to take these decisions, or because some required conditions are not gathered yet. In other words, the project manager needs to take important planning decisions at the macro level, without having to decide about the details at the micro level. Detailed planning therefore works to satisfy the conditions imposed

by aggregate planning. Nevertheless, the bottom-up "laws of physics" are not totally ignored. The overall initial planning process is iterative. Once the project plan is developed at the detailed level to satisfy the aggregate targets, not always this is feasible. Furthermore, the analysis at the detailed level provides new insights about implementation risks and about why aggregate targets might not be feasible. This way, feedback is provided to the aggregate level where targets and milestones are reviewed, re-negotiated with the client and eventually readjusted. The process iterates between aggregate planning and detailed planning, until a satisfactory work plan is achieved at all levels. Overall, the initial planning process can be described a U-curve process as shown in figure E.3 below. Iteration to previous steps can occur at any stage in the process, as suggested by the dotted circles.

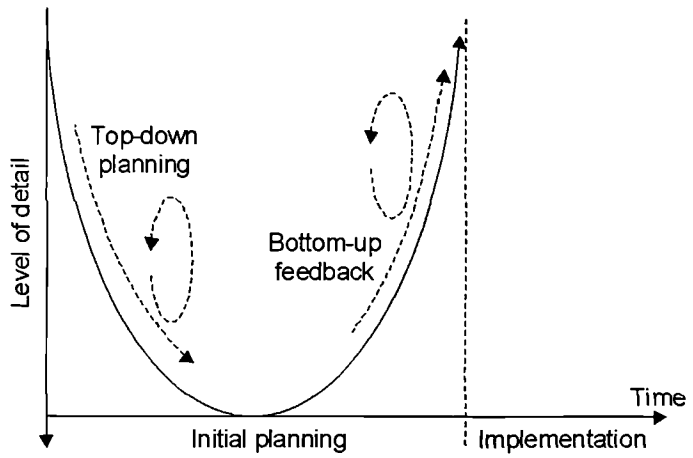


Figure E.3 – The U-curve process of initial planning

In much of the literature, the planning process is presented as static linear sequence of steps which lead to the final plan. While useful for pedagogical purposes, it is the author's opinion that in the real world planning needs to be much more flexible and follows a dynamic process as just described above. Continuous Iteration, refinement, and rework of the project plan is particularly important when various planning tools and techniques are being used together, in an articulated and integrated manner.

Appendix F – Basic project management tools and techniques

Overview

The basic project management tools and techniques were briefly described in section 2.2.2. A more detailed description is here provided. This includes some comments and suggestions about how they should be better applied within the project management process, their importance, strengths and weaknesses. These comments reflect the author's own view.

Product breakdown structure (PBS)

This is a simple technique aimed at decomposing the project output product into elementary sub-components. The PBS consists of a progressive breakdown where the product is decomposed into various levels, until the final sub-components are considered so simple and functionally self-contained, that further decomposition is not perceived useful. This specification is often confused with the project work breakdown structure (WBS). However, there is a fundamental difference between the product components and the work required to developed them. In particular, the scope of the project work is wider than the product specification. On the other hand, some project work is not aimed at developing directly the product (e.g. administrative and management work), and the development of some product sub-components requires different types of work.

Figure F.1 below provides and example of a very simple PBS, which specifies a telecommunications software system.

Product Breakdown Structure (PBS)

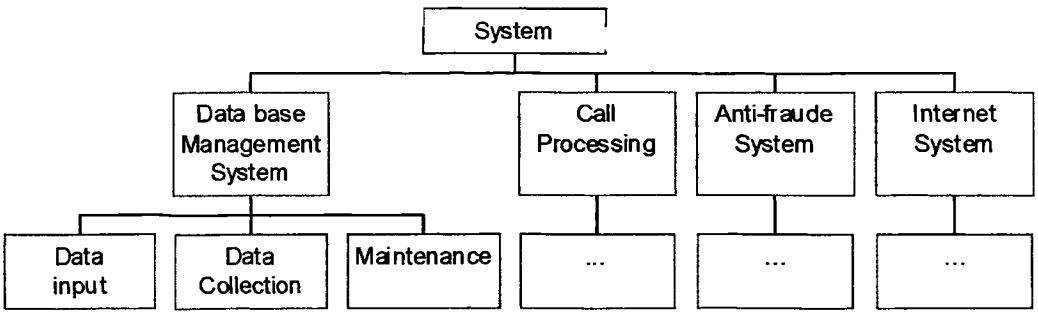


Figure F.1 – Example of a simple product breakdown structure

In each level of breakdown, a decomposition criteria is applied. In most cases, functionality is the criteria used. However, other criteria like Client (e.g. a product being developed for few specific Clients) or geography (e.g. parts of the product being developed at different sites).

The decomposition of the product in this way provides various benefits to the planning process, in particular:

- provides an overall view of the product architecture, its sub-components and basic functional relationships;
- it helps to identify the required project work and break it down into sub-tasks;
- resource allocation, in particular identification of resources expertise in the various functional areas;
- communication with the Client about the product definition and functional requirements;
- identification and planning of deliverables to the Client and project milestones;
- identification of an appropriate development process.

The first specification of a PBS often takes place at the beginning of the project, as part of the contractor's proposal, or even in the Client's request for proposal (RFP).

This initial version of the PBS is then refined and used to develop other planning elements.

Milestones and deliverables chart

This is a very simple technique, which is valuable in many ways throughout the project, but is often disregarded. A milestones and deliverables chart specifies the planned dates for these two elements, throughout the whole project. Milestones are major events considered as landmarks of project progress, like the completion of the design phase. Deliverables are specific sub-products which delivered to the Client for various purposes. A deliverable can be a prototype, design documentation, an intermediate product release or the even final product. Milestones often have associated deliverables.

A milestone chart can be considered as a high-level strategic plan and a contract with the Client. It specifies the major achievements planned throughout the project, which are often the basis of the contractual agreements with the Client regarding progress. At the eyes of the Client, the project proceeds with success as planned if milestones and deliverables are achieved on schedule. In the contract, these are often the basis of Client payments as well as penalties.

A milestones and deliverables chart should specify the various dates associated with each milestone and deliverable, and other relevant information. Ideally, the deliverables should be identified from the PBS specification – as already mentioned, all planning techniques should be implemented in an integrated manner. A simple example is shown in table F.1 below.

No.	Milestone	Deliverables	Planned date	Estimate date	Actual date	Delay
1	Design completion	System design for review	01/01/2000	20/01/2000	20/01/2000	20
2	Prototype completion	Prototype	01/03/2000	01/04/2000	NA	30
...	...	...	...	...	...	...
n	FAT	Final version	20/04/2002	20/04/2002	NA	0

Table F.1 – Simple example of a milestone and deliverables chart

This planning element is refined and updated continuously throughout the initial planning process and throughout project implementation. As the plan is detailed, the milestones and deliverables can also be refined into more detail. The detailed planning may also identify the need for additional milestones. As the project plan is implemented, actual and estimated dates are updated as required.

Risk register

The risk register is a technique similar to the milestones and deliverables chart. The risk register is the central element of the whole risk management process. It is aimed at identifying, analysing, monitoring, controlling and mitigating the relevant project risks.

The risk register consists of a table listing the currently identified risks, ranked by "seriousness". A risk is more serious as its probability of occurrence and estimated impact on the project are greater. Mathematically, a risk' s "seriousness" can be taken as the product of the probability (in %) with the impact (in $) – in this way "seriousness" is measured as an expected monetary value (EMV). Where these quantitative estimates are not available, a qualitative scale can also be used. While this criteria is commonly used to rank risks, Williams (1996) argues that probability and impact must be considered separately and that relying solely on the EMV can be misleading. The information recorded in the risk register vary according to the project management needs. Typically, it includes the rank, a certain classification (e.g. source), probability, impact and mitigating actions. The identification of mitigating actions shows the pro-active nature of risk management: before a risk occurs, mitigating actions are devised to counter its impacts.

Mitigating actions can be reactive or proactive. Reactive actions are those which are only implemented if the risk occurs. Proactive actions are implemented prior to the risk occurring, typically as soon as its probability and seriousness are considered too high. The risk register may also include information about the status of mitigating actions (e.g. needs detailing, ready to implement, partially implemented, under implementation, implemented). For illustrative purposes, a simple example of a risk register is shown in figure F.2 below.

Rank	Description	Type	Prob.	Impact	Mitigating Actions	Status
1	Requirements changes	Client	High	High	Start development of unstable areas at latest possible start	Implemented
2	Lack of specialised resources	Resources	Medium	High	Start recruitment program earlier	If probability becomes high
3	Product complexity	Product	Medium	Low	Increase level of detail of the designs	Develop contingency plan
...	...	...	...	...	...	...

Figure F.2 – Simple example of a risk register

As shown in this figure, colours are often used to enhance the visual nature of a risk register: the risks at the top of the table are the ones where management should concentrate most of the attention. These risks need close monitoring of occurrence and may require the implementation of the associated proactive mitigating actions.

Front-end estimating techniques

Front-end estimating techniques are aimed at providing estimates for the project cost, schedules and resources required, prior to the development of a detailed work plan. These techniques are aligned with a top-down approach to planning, discussed below in this appendix (see figure E.3). These techniques do not require detailed information about the project, which in general is not available at the early stages.

There are various possible techniques available for front-end estimating. Some are more structured than others. Most of them can be grouped into one of the following three categories:

- *expert judgement* – various opinions from experts with experience in the field are combined to generate the estimates. This is the only source of information used. The way in which the opinions are combined can be more or less formal. Structured techniques, like the Delphi method (Wright 1985, Kerzner 1998), can be used for this purpose;
- *by analogy* – the results achieved in the most similar past project are used as estimates for the new project. If such similar past project is available, this technique provides an easy and quick estimate. However, such estimate can be highly biased by the specific issues of the past project, including poor management;
- *empirical mathematical models based on regression analysis* – these models provide a more rigorous way to use information about past similar projects in order to estimate the likely results of new projects. They require the availability of a fairly extensive database of past projects (description and results). Regression analysis is carried out to identify the relevant project characteristics that affect the project outcome and how this is affected (i.e. regression curves). These characteristics are quantified as input parameters to the model, when estimates for a new project as produced. The accuracy of these models depends on the database of past projects (amount of projects and quality of the data).

Empirical models are the more structured and rigorous technique since it is based on mathematical statistical analysis. As such, they can be expected to be more accurate than the other two. These models are used in various industries but have become particularly popular in the software industry. The first model proposed was the COCOMO model (COnstructive COst MOdel, Boehm 1980). Later, two specific models were improved converted into powerful software tools: the KnowledgePLAN (Jones 1998) from Software Productivity Research (SPR), and the SLIM model (Putnam and Myers 1997) from Quantitative Software Measurement (QSM). These models are "black-box" type of model, since the calculation process is not visible to the user, as illustrated by figure F.3 below.

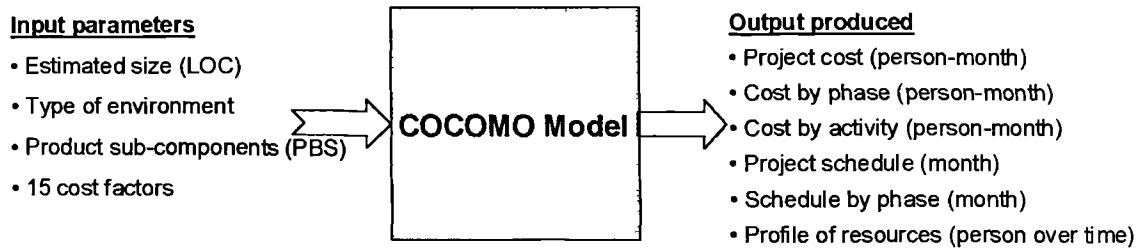


Figure F.3 – Overview of the COCOMO model

The COCOMO model considers that the software development process within the project follows the classical life-cycle of phases and continuous activities (Pressman 1996). It provides estimates regarding effort required (i.e. cost), schedule and resource requirements over-time (i.e. profiles). It breaks down the effort by phase and activity and the schedule by phase. The resource requirements are also decomposed by activity within each phase. As inputs, the COCOMO model has three level of formality. In each it requires more inputs. At the detailed level, it requires the estimate size of the software system decomposed by product component (the product decomposition can be taken from the PBS previously specified), a description of the type of development environment (three possibilities are considered), and about 15 cost factors, which describe various aspects of the project including both soft and hard factors (e.g. staff experience, product complexity).

The estimates produced by front-end estimating techniques at this stage may counter contractual agreements with the Client, as described in milestones chart. For example, the project completion date may be later than agreed. It can be argued that the same technique could have been applied when the dates were to be agreed by the Client. While that is often the case, the information available is generally less in the early planning stages. For example, with the COCOMO model, often there is only information available to implement the basic level when dates need to be agreed with the Client. As more information becomes available, the model is implemented again at a more detailed level, providing new estimates. Where the new estimates counter the results agreed with the Client, either the milestones are re-negotiated or the project manager assumes the risk of optimistic estimates. These risks are often dis-aggregated into optimistic assumptions. The

empirical model can be used for this purpose: for example, if higher staff experience is considered the estimates produced by the model might be as desired. Whether re-negotiation takes place or risks are assumed, the milestones and deliverables chart and the risk register should be updated accordingly. This highlights the ideal integrated nature of project planning process in using the various planning techniques.

Work breakdown structure (WBS)

The work breakdown structure (WBS) is central to the whole process of project planning and control. The WBS specifies all the work that needs to be performed, so that the project objectives are achieved. Whatever needs to be incorporated into the product, or needs to be accomplished to satisfy the Client's expectations, must be translated into project work and thereby specified in the WBS. The project scope and product functionality are therefore directly related to the WBS.

The WBS specifies the project work as a decomposition tree, in a similar way to the PBS described above. The whole project is decomposed from a single task down to various sub-tasks, throughout various levels of breakdown. The aim is to achieve a set of elementary tasks, which are simple and can be related to past experiences, so that they are easy to implement within estimated targets. The underlying principle behind a WBS is that if all of these elementary tasks are successfully managed, the whole project will be completed on time and within budget. This is the core principle underlying the whole traditional approach to project management (Rodrigues and Bowers 1996a). Figure F.4 provides a simple example of a WBS for a software development project.

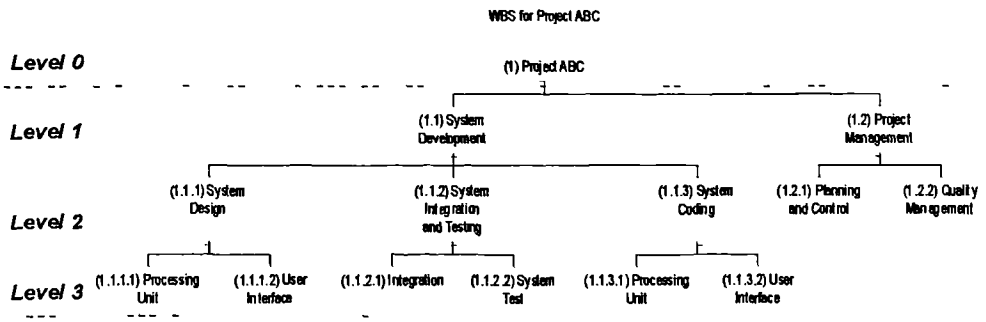


Figure F.4 – A simple example of a WBS for a software development project

This WBS breaks down the whole project work into 8 elementary sub-tasks. Overall this decomposition considered four levels of breakdown (i.e. level 0 to 3). Not all elementary tasks need to be in the same level. All tasks in the WBS should given a unique identification label or code. In the example above, a coding scheme was adopted which helps to identify the level of breakdown in which the specific task is. Another important issue is that the structure of a WBS does not imply any logical sequencing of the tasks over time.

In a real complex project, a WBS can have hundreds to thousands of tasks. The WBS is specified in a formal document and generally also implemented in a software system.

The project WBS plays an important role within the project planning and control process:

- it provides the elementary tasks to be scheduled, and which are the basis of resource allocation;
- it provides a basis for scope specification and scope control;
- it provides a basis for budgeting and cost control;
- it provides a basis to monitor performance in the various project areas.

Once developed, the WBS is continuously updated throughout the structure as more scope (and work) is added to or removed from the project. In this way, scope changes must be translated into changes in the WBS. The tasks in the WBS are also updated in terms of budget and costs. The actual costs incurred are entered in the elementary tasks and existing budgets may be revised. As the WBS evolves throughout the project life-cycle, it is important to keep record of its past versions.

An important issue regarding the WBS is how to developed the right WBS for the specific project at hand. Ultimately, the WBS is a model that represents the project work. There is no right or wrong WBS for a project. There can be different WBS's for the same project. The appropriateness of a WBS depends on how well addresses the specific problems and issues of the project and therefore the

managerial needs. Whatever the WBS developed for a project there are two basic principles that must be respected:

- *completeness* – all work carried out in the project, with no exception, must be mapped to a certain elementary task in the WBS. The WBS must not be partial and must provide a complete representation of the project work;
- *hierarchical* – the work, budget or actual cost of each non-elementary tasks in the WBS must be equals to the sum of the work, budget and cost of all of its sub-tasks. The overall project work, budget and actual cost is equals the sum of these elements of all elementary tasks.

If any of these two principles is not respected the WBS will not be useful to the project management process. The development of a WBS results from applying certain decomposition criteria whenever tasks are further decomposed into sub-tasks. This is, the decomposition is not arbitrary. There are various criteria that can be applied, in particular: product oriented, time based, functional, geographical, development process, type of work and client oriented. For example, in figure F.4 above the criteria used from level 0 to level 1 was clearly the type of work. The decomposition of task (1.1) from level 1 to level 2 was the product development process, whereas the decomposition of task (1.2) from level 1 to level 2 was based on a functional criteria. The decomposition of task (1.1.1) from level 2 to level 3 was based on the product structure. Various criteria are used down the WBS. The decision about what criteria to apply at any stage should be based on the management concerns. The most important problems should be addressed at the higher levels of the WBS. For example, a complex product made up of large components which can be developed in parallel, calls for the application of the product structure criteria.

Another important decision is the level of breakdown. There is no universal rule to determine the ideal level. The more detailed the WBS the simpler the elementary tasks will be, thus easier to manage. However, too much detail can be counter-productive and implies much more effort required to updated and maintain the WBS – this is, a more expensive WBS. In general, the elementary tasks should be:

- easy to estimate in terms of budget, schedule and resources required;
- the work contained in them must be easy to understand and as much "self-contained" as possible (i.e. few interactions while with other tasks while the work is underway);
- "finish-to-start" precedence relationships must be easy to identify.

Another important factor that affects the structure of the structure of the WBS is the type of organisational system adopted. As discussed in the previous sub-section, functional systems imply that the WBS is tailored for the functional structure of the existing organisation. In this case, the dominant criteria applied in decomposing the project is the functional criteria. On the other hand, in "projectized" organisations the product structure and the development process tend to be the dominant criteria.

The WBS is often used as a model for a bottom-up cost estimating process. The elementary tasks are first budgeted individually. The estimated project cost is then calculated aggregating these budgets up through the hierarchy of the WBS. However, in most projects when a WBS is developed the overall project budget is already determined (see milestones charts and front-end estimating techniques described above). In this case, a top-down process is followed instead where the project budget is decomposed down the WBS to the elementary tasks. In each level of breakdown, certain criteria are required to decompose the budget of a task into its sub-tasks. Metrics available from past projects are useful to carry out this process in the higher levels of aggregation. For example, in the WBS of figure F.4 past metrics could indicate that 30% of the project budget should be allocated to project management work. Expert judgement is also a very important source of information. At the very bottom level of the WBS, the allocation of the budget to

the elementary tasks is often based on a negotiation process with the specific teams which will accomplish the various tasks. The allocation the project budget to the WBS tasks is an important step in the initial planning process. As the budget is decomposed down to the elementary tasks it becomes more evident whether the budget is appropriate for the project work scope and whether it is being properly distributed among the different areas of the project. This is therefore an iterative process where the allocation is revised and where the overall project budget itself may need to be renegotiated.

The WBS is a key element within the traditional project management framework. Therefore it must be developed with most attention. Its structure must address the managerial concerns and needs that steam from the specific issues of the project. The quality of the WBS will determine the performance of the project management process through the life-cycle and therefore the project success.

Organisation breakdown structure (OBS)

The organisation breakdown structure (OBS) is also an important technique used in the project planning process. The OBS specifies the structure of the organisation which will accomplish the project work. Similarly to the PBS and WBS, the OBS is a hierarchical structure specified in the form of a tree, where individuals at the top manage the individuals below. The OBS can have a great influence on the project outcome because it addresses important organisational issues of human nature within the project. In particular, the OBS establishes:

- relationships of authority between individuals;
- teams of individuals, which will have to work together.

The relationships of authority determine who should report progress to who, and who is responsible to monitor the work progress of others. It is very important that persons with the right profile are given the position of leader. Poor leadership is often a major cause of problems. Equally, the individuals grouped in a common team must be compatible one another.

Like with the WBS there is no right or wrong OBS for a project. The structure of a OBS must address the various specific issues of a project, from technical to human factors. However, like with the there are two basic principles that must apply:

- *completeness* – all individuals that work in the project must be specified in the OBS, regardless of the amount of effort they spend in the project. This ensures that there is no work accomplished in the project for which there is no one responsible;
- *hierarchical* – any individual has direct authority over the individuals immediately below them. The individuals below report progress and any type of problem to the individual immediately above them. This brings discipline to the communication process and ensures that responsibilities are easy to identify and to handle (as opposed to everyone demanding explanations from everyone).

There are various factors that should be considered in the development of the project OBS: Like with the WBS, various criteria can adopted to split authority and form working teams: product, geography, client and functional among others. For example, separate teams can be formed to work in different parts of the product, or in different locations. The specification of the OBS also depends on the type of organisational system. For example, in a "projectized" organisation the OBS is tailored to fit the logic of the WBS. In a functional organisation, the dominant criteria is functional (e.g. production, marketing, human resources).

Throughout the project life-cycle the project team can change in many ways. New individuals may be brought in into the project while others may leave. Sometimes, responsibilities change and new leaders are appointed. Sub-teams and disbanded and new ones are formed to address the changing resource needs of the product development process. The OBS should be updated accordingly and thus always reflect the current status of the project organisation. Like with the WBS, it is important that the various past versions of the OBS are recorded in the project information system.

Overall the OBS is an important element in the project management process because it specifies the resources available in the project, it is the basis of the

"vertical" communication within the project and identifies the responsibilities among the project team members.

Responsibility matrix (WBS x OBS)

The OBS specifies the responsibilities among the project team members – i.e. who is responsible for whom. Another important type of responsibility is the relationship between the team members and the project work – i.e. who is responsible for what, and how. This type of responsibilities is specified using a technique called the project responsibility matrix.

This technique consists in developing a matrix by crossing the WBS against the OBS. Each cell of the matrix corresponds to a couple (team member, work task), in which the type of responsibility is specified. There can be different types of responsibilities that individuals can have over the project work. The basic responsibility is "execution", which means that the individual is responsible for executing the work specified in the specific project task (e.g. developing the design of a system component) . The responsibility "approval" means that the individual is responsible for approving the completion of the work carried out in the task (e.g. approving the design developed). Each cell in the responsibility matrix can specify more than one type of responsibility that each individual has over a specific project work task. For example, one individual can be responsible for designing and approving the final design. There is no standard set of types of responsibilities that can be considered. Depending on the specific needs of the project, particular types of responsibilities can be considered. However, some types of responsibilities are common to most projects. Table F.2 below compares different sets of responsibilities proposed by different authors – the responsibilities in the same line of the table are proposed as similar but not necessarily equivalent.

PMI (1996)	Kerzner (1998)	Turner (1993)	Nicholas (1990)
P – participant	Operations man.	X – executes	P – primary respons.
A – accountable	General management	P – manages progress	S – secondary respons.
R – review required	Must be notified	I – must be informed	N – must be notified
I – input required	Must be consulted	C – must be consulted	
S – sign-off required	Must approve	D – takes decision	A – must give approval
	Specialised	d – partial decision	
	May be consulted	A – available to advise	
		T – provides tuition	

Table F.2 – Different types of standard responsibilities used in the responsibility matrix

A simplex example of responsibility matrix developed for a software project is shown in figure F.5, using Turner's notation. The WBS is identified in the left column and the OBS is identified in the top line of the matrix.

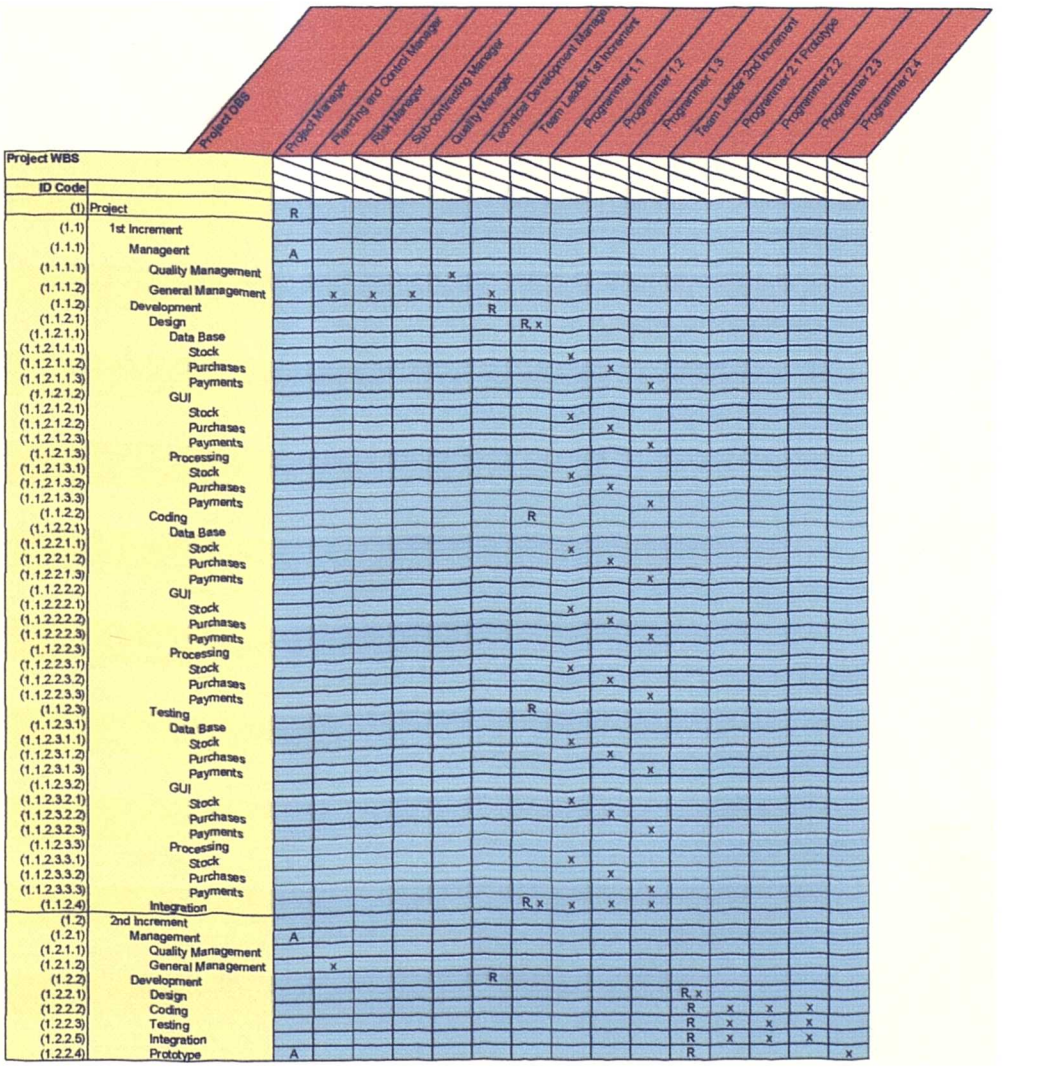


Figure F.5 – Example of a simple responsibility matrix

As expected there are many cells in the matrix with no responsibilities specified, meaning that certain individuals have no responsibility at all for certain areas of the project work.

In practice, the responsibility matrix is developed at different levels of detail, considering different hierarchical levels in the OBS and WBS. At the full detailed level, the responsibility matrix can have thousands of cells. This can be implemented in a document, where the matrix is divided into various sub-matrices.

The responsibility matrix plays an important role in the project management process. In the first place, it is the basis of the "horizontal" communication within the project team. By looking at the matrix, any team member knows to whom certain matters should discussed and clarified with. This is particular important in medium-large projects where new staff is frequently joining the project. The responsibility matrix also plays an important role in the diagnosis of problems and identification of responsibilities. It allows the project manger to know who was responsible for executing, providing advice and approving a certain part of the project work.

Like with the WBS and OBS, the responsibility matrix will change throughout the project life-cycle. If changes on the WBS and OBS occur then the responsibility matrix must be updated accordingly – e.g. new tasks need to be assigned to some one. Furthermore, as the project managerial and technical needs change throughout the life-cycle, the responsibilities assigned to individuals should be readjusted in order to better respond to the new project conditions. Like with the WBS and OBS, it is important to keep a historical record of the various versions of the responsibility matrix.

Gantt charts and PERT/CPM networks

Once the project work, the project team, and the responsibilities have been specified, the final step into developing an operational plan is the allocation of the resources to the project tasks, and the scheduling of these tasks. In order to carry

out this work, there are two main techniques commonly used, which are specialised for project management purposes: Gantt charts and PERT/CPM networks (also referred to as critical path logical networks). These techniques are the basis of the whole project planning function (see figure E.1). There is extensive literature available on this topic, in particular regarding the use of PERT/CPM based models. Complex mathematical enhancements have been proposed over-time. It is not the purpose of this research to explain them in detail. However, since these techniques are central to project planning, it is fundamental to understand the basic underlying principles and concepts.

In principle, the tasks to be scheduled and to which resources are to be allocated at the operational level are the elementary tasks of the WBS. However, sometimes the WBS is not specified at sufficient level of detail and further decomposition id required. There are various factors that need to be considered in order to determine the schedule of a task:

- the resources allocated affect the task duration;
- the initiation of the task can depend on the completion of other tasks;
- the initiation of the task can depend on external deliverables (e.g. Client approval, sub-contracted work).

Gantt charts were proposed by Henry L Gantt in the early XX century (Nicholas 1990). They provide a simple time-based representation of the tasks schedules. The Y-axis of the chart identifies the tasks and the X-axis represents time. For each task in the Y-axis, a bar chart is displayed from its starting to its finishing scheduled dates. Figure F.6 below provides a simple example regarding a software project.

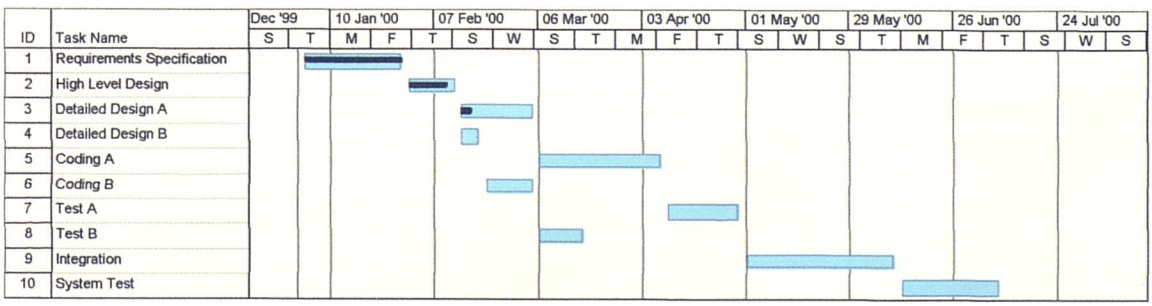


Figure F.6 – Example of a Gantt chart showing the work plan of a software project

Gantt charts are generally developed at different levels of detail, forming a hierarchy of charts. High level Gantt charts are used to show the big picture of the project, identifying the major milestone dates (e.g. project phases). Detailed Gantt charts are used to plan the work of the individual project sub-teams. The representation shown in figure F.6 above is the basic type of display of a Gantt charts. Nowadays, software tools allows for the development of more elaborated charts, where other elements of planning are shown (e.g. resources names, external deliverables). These display features depends on the specific tool. Gantt charts are also used for progress monitoring. Typically, some form of representation is used to identify progress within the tasks. Figure F.7 provides an example: the black lines within the tasks identify the work progress and the red vertical line identifies the current status date of the project. It is easy to see that task 2 is behind schedule whereas task 3 is ahead of schedule.

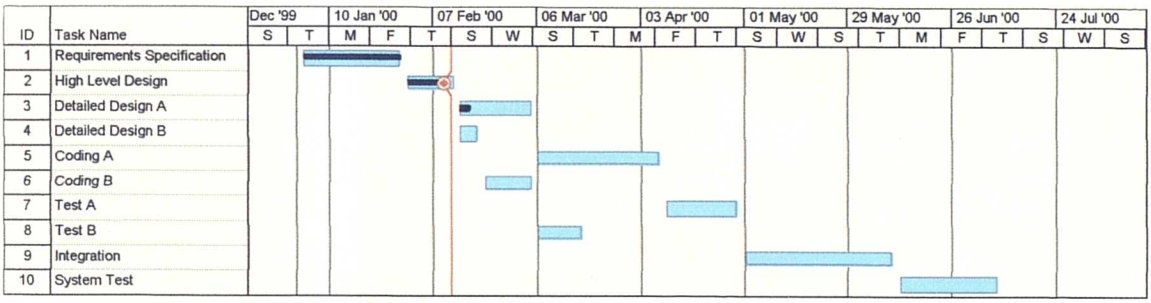


Figure F.7 – Example of a Gantt chart showing progress information

Gantt charts have advantages and disadvantages. The major advantages are that they are easy to developed and understand and provide an intuitive view of the project schedules. Because they can be developed at different levels of aggregation that can be used to provide a high-level view of the project and are an excellent tool to communicate and negotiate with the client. The major disadvantage of Gantt charts is that they do not consider the factors that restrain the initiation of the tasks, in particular their precedence relationships (i.e. which need to be completed so that other can start). Without this element, Gantt charts cannot be used as models to analyses and forecast the project outcome. The start and finishing dates are simply imposed on the tasks regardless of technical feasibility. It is not possible to determine what is the overall impact on the project of

a delay in a certain task nor to identify which are the more important tasks (the ones which delays are likely to have a greater impact).

In order to overcome the limitations of Gantt charts, the PERT/CPM technique was developed in the mid-1950s (Project Evaluation and Reviewing Technique / Critical path Method). PERT and CPM are not exactly the same technique and there are important differences between the two. However, over-time the two techniques have been used together and hence labelled as a single technique PERT/CPM. This technique is based on the concept of logical network which focuses on the precedence relationships that exist between the project tasks. Because these relationships often have to do with technological constraints, they are often referred to as "technological dependencies". The PERT/CPM technique links all the project tasks into a network according to these "finish-to-start" dependencies. There are two different types of notation: "activity on the arc" (AOA) and "activity on the node" (AON). Figure F.8 provides a very simple example of these notations.

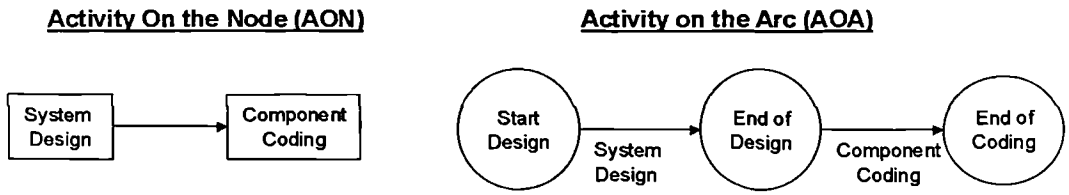


Figure F.8 – The two alternative notations to represent PERT/CPM logical networks

The AOA notation was preferred in the early days of project management due to some advantages regarding the calculations involved in determining the tasks' schedules. However, with the use of computer applications this advantage became irrelevant. On the other hand, it has the great disadvantages of requiring the use of fictitious tasks (with zero duration). In part because of this, they less intuitive to understand. The AON notation is nowadays preferred and it is used in most software tools. Figure F.9 below shows an example of a PERT/CPM network, which corresponds to the Gantt chart in figure F.6.

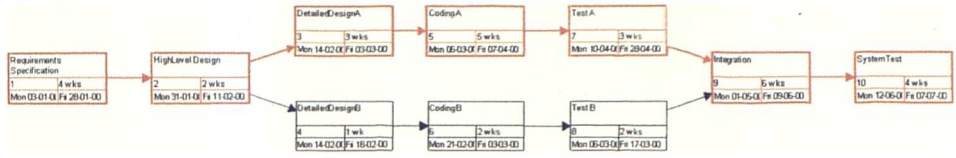


Figure F.9 – Example of a simple PERT/CPM network plan (AON notation) representing a software project plan

The PERT/CPM networks generally display the project tasks from the left to the right, suggesting an over-time sequence of implementation. However, unlike Gantt charts they are not strictly time-oriented. Instead, they focus on the logic of implementation based on the precedence relationships. If a task depends directly or indirectly on another task then it will start later. Otherwise, if there is no dependency between two tasks, then the graphical display does not impose any time- sequence. For example, in the network of figure F.9, the task "Coding B" starts mush later than "Coding A", yet the graphical display suggests that would start at the same time.

Unlike the Gantt charts the PERT/CPM networks can be used as forecasting models, producing the likely outcome of the project as well as other risk information. In this way, they support "what-if" analysis and therefore can be used for experimentation of different planning alternatives. The project plan can therefore be improved in an iterative manner, prior to implementation, until a satisfactory outcome is achieved. The way in which a PERT/CPM network is used as a test-bed model is based on some main concepts and calculations. Extensive research in this field been carried out to over-time to consider a wider range of factors and conditions (e.g. GERT, Nicholas 1990). The result are increasingly complex models, often not easy to understand by managers and which require extensive sets of data, thereby threatening their practical implementation. It is not the purpose of his research to explain the whole "PERT/CPM based theory", as well as the complex mathematics involved. The core relevant concepts are here briefly described, assuming the very basic version of PERT/CPM networks. These are as follows:

- *finish-to-start dependency* – establishes that a certain task can only start after another task is 100% complete. For example, in figure F.9 there is a finish-to-start dependency from "Requirements specification" and "High-level design";
- *successor* – task which start is restrained by a "finish-to-start" dependency. For example, "Detailed design " is successor of "Requirements specification";
- *predecessor* – task which restrains the start of another task in a "finish-to-start" dependency. For example, "Requirements specification" is predecessor of "High-level design";
- *path* – complete sequence of tasks, from the beginning of the project to the end of the project, links through "finish-to-start" dependencies. For example, in figure F.9 two paths can be identified;
- *path duration* – sum of the duration of the tasks that make the path. It is the time that would take to complete the sequence of tasks if there were no other restrictions;
- *critical path* – path in the whole PERT/CPM network which has the longest duration. This is the path that imposes the duration of the whole project (e.g. path in red in figure F.9);
- *earliest start* – earliest date that is feasible to start a task (i.e. without violating any "start-to-finish" dependency);
- *latest start* – latest date that a task can start without the delaying the project completion date;
- *earliest finish* – earliest date that is feasible to complete a task;
- *latest finish* – latest date that is feasible to complete a task without delaying the project duration;
- *total float* – time the completion date of a task can be delayed without delaying the completion of the whole project;
- *local float* – time that the completion date of a task can be delayed without delaying the start date of any of its successors (and thus of any other task);
- *critical task* – task that belongs to the critical path;
- *near-critical path* – path which tasks have very small total floats and thus, a minor delay can transform the path into a critical path;
- *near-critical task* – task with a very small total float.

The proposed definitions are simplified as much as possible. For example, floats can be split into "floats to the left" and "floats to the right". If other types of dependencies are considered, like "start-to-start" (i.e. a task can only start after another task has also started), then the definitions above would have to be revised and new concepts would e considered. Nevertheless, any enhancement of the PERT/CPM model is based on core set of concepts proposed above.

The use of the PERT/CPM model is based on the following calculations:
(1) earliest start and earliest finish of each task;
(2) latest finish and latest start of each task;
(3) total and local float of each task.

The calculations in (1) are based on the "scanning" of the network from the left to the right. The duration of the tasks in each path are accumulated generating the earliest start and finishing dates. The calculations in (2) are based on the "scanning" from the right to the left, where the duration of each tasks is subtracted from the project completion date, generating the latest finishing and start dates. The calculation of the total float for each task in (3) is simply equals the difference between its latest and earliest dates. The calculation of the local float is equals the minimum of the earliest start dates of its successors, minus its earliest finish. At the end of the process the critical path is identified (i.e. path with the longest duration), the estimated project completion date is determined (i.e. completion date of last task in the network), and the critical tasks are identified (i.e. the ones that belong to the critical path and have no total float).

The results from this analysis determined the dates when project tasks should start and finish, as well as the overall over-time profiles of resources required. It also provides information which can help the project manager. The recommendations can be summarised as follows:

- *critical tasks should be given priority.* This includes resource allocation, quality of resources employed, and rigour of managerial control. If these tasks are delayed, the whole project will get delayed and resource requirements over-time change;

- *near-critical tasks should be given special attention.* These tasks have a very small float and hence deserve to be treated almost as if they were critical;
- *tasks with a considerable total float but with no or small local float require some attention.* If these tasks are delayed, the whole project may not get delayed but this will have an impact on the start of other tasks, which in turn can have important implications regarding resource scheduling;
- *a project plan where, overall, tasks have short floats can be risky.* This often means that the plan was "compressed" as much as possible in order to reduce the overall project duration. A plan like this makes the project very sensitive to problems and delays in individual tasks. The whole project can easily get delayed and the profile over-time of resource requirements also changes – in both cases, there can be serious cost impacts; various quality impacts can also result from this. Indices like "average float" across all tasks can be produced to provide an indication of the overall project risk level.

As mentioned, the PERT/CPM model can be used as a "test-bed" model to assess the outcome of alternative plans and planning decisions. The project plan can be changed in many different ways. There are some typical re-planning actions, some of which can be automated, or semi-automated, using software tools which implement goal-seeking and trial-and-error algorithms. These are as follows:

- *keep the overall resource availability and reduce the project duration to a minimum.* This re-planning action is based on an iterative process of transferring resources from non-critical tasks to the critical tasks. This is based on the assumption that the tasks' duration reduces as more resources are allocated to them. As critical tasks get more resources, the overall critical path duration is reduced. At the same time, non-critical tasks get their floats reduced. Throughout the process, the critical path can change several times. This leads to a "compressed plan" with a minimum shorter duration, but much more sensitive to delays in the individual tasks, hence more risky. No additional resources are added to the project;
- *"crashing" the plan* – the project duration is reduced by adding resources to the project and allocating them to the critical tasks. This process is similar to the previous ones except that resources are added to the project and hence leads to a more expensive solution. However, the overall floats might not be so small

and hence the plan might be less risky. It is generally assumed that there is a limit to reduce the duration of each individual task , and hence there is a minimum possible level of "crashing";

- *"crashing" tasks* – the elementary action of adding more resources to an individual tasks, reducing its duration;
- *resource levelling* – this re-planning action making use of the available floats, the tasks are re-scheduled so that the project duration does not change but the overall profile of resources required over-time is "levelled". This means that the profiles will be more stable and have less variations (i.e. less "up and downs"). There are various advantages in having stable nearly-constant resources profiles. Sometimes, acceptable solution require the project duration to be extended;
- *resource loading* – same as the previous one, but the aim this time is to prevent that a specified maximum level of resources is not exceeded. This can result, for example, from space constraints or from equipment availability. The solution leads to a resource profile which does not exceed a certain maximum level, regardless of its shape over-time. Sometimes, a feasible solution may require the project duration to be extended.

The specific re-planning actions that can be implemented in a plan are numerous. Apart from changing resource allocation to the project and resource scheduling among the tasks, and changing the tasks start and finishing dates, there are other possible re-planning actions: tasks can be added or removed from the project plan, tasks can be aggregated or dis-aggregated to increase work concurrency, technological dependencies can also be added or removed. The PERT/CPM network may also consider dependencies from external deliverables (e.g. sub-contracted work), which can also be the basis to conceive re-planning decisions. All these actions will have a certain impact on the project outcome. The advantage of the PERT/CPM network as a "what-if" model is that it provides a forecast of the likely impacts on the project outcome. Within this context, the PERT/CPM network plan can be considered as a "black-box model", as shown in figure F.10 below. The output of the model provides results in terms of time, cost, resources and floats (which can be seen as indication of risk). Therefore the model can be used

to identify solutions that best balance these four aspects of the project outcome, in particular to assess cost-time trade-offs.

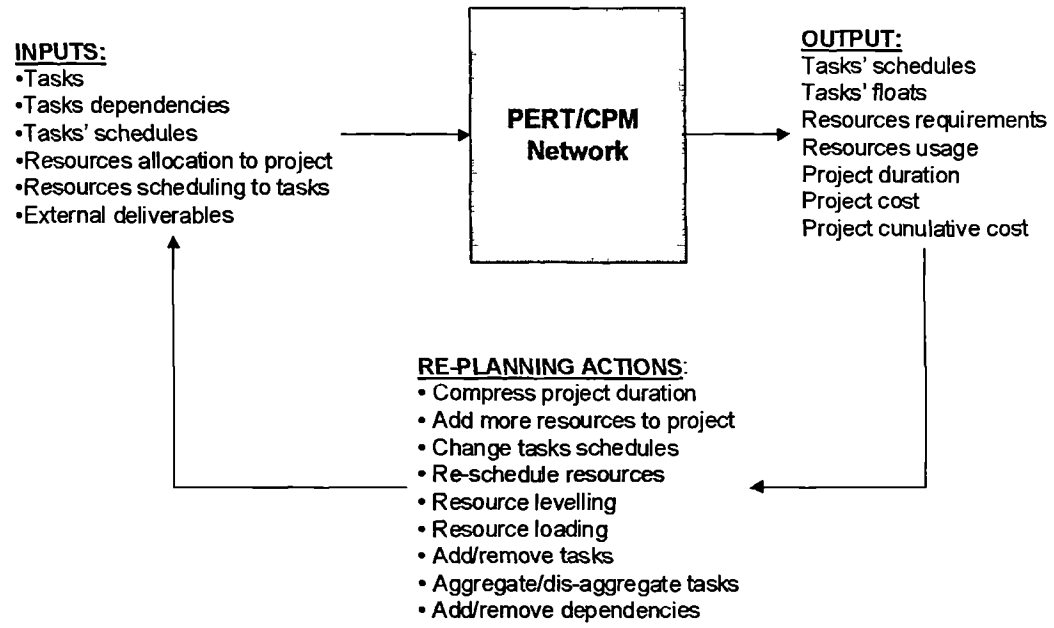


Figure F.10 – The PERT/CPM network plan as a "black-box" model for "what-if" analyses

The PERT/CPM technique as described above is centred around the identification of the critical path and on the assumption that the tasks' duration can be reduced by employing more resources. This allows to developed alternative plans on the basis of cost-time trade-offs. The concept of "float" is important because it provides an objective indication of risks and is the basis to devise efficient re-planning actions (e.g. less duration with the same resources).

As already mentioned, PERT and CPM are different techniques. As described above, the PERT/CPM analysis corresponds to CPM. Overall, the CPM analyses suggests that the project cost follows a U-curve, as a function of the duration. This is illustrated in figure F.11 below. The total project cost includes all the types of cost, direct, indirect and penalties for delays. The more resources are added to the project in order to crash the activities, the project duration is reduced but at the expense of more cost. This is only feasible down to a lower limit of duration. On the other hand, if less resources are allocated to the project, the project cost reduces at the expense of more time. After a certain point, the project cost tends

to increase again due to high penalties for late completion. In theory the optimum plan that CPM aims at identifying is the one that leads to the minimum cost – i.e. the optimum trade-off between cost and time.

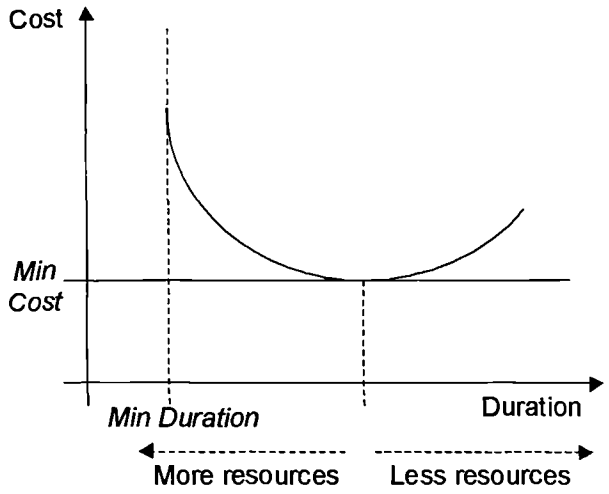


Figure F.11 – The project cost U-curve derived from CPM analysis

The PERT technique was conceived to model large-scale R&D projects, where there was a high uncertainty regarding the tasks' duration. Like CPM, PERT is based on the logical network of tasks and on the critical path. The main difference between PERT and CPM is that PERT considers that the tasks' duration follows a probabilistic distribution, instead of a deterministic estimate. This is, once resources have been allocated to the project tasks, the likely duration of each task is uncertain. PERT assumes that this uncertainty can be properly modelled by a three-point time estimates of duration: (a) optimistic, (m) most likely and (b) pessimistic. Therefore, instead of a single estimate, three estimates are proposed for each task. PERT then assumes that the duration of each task follows a beta distribution (Nicholas 1990). The expected duration (U) and associated variance (V) of each task are calculated as follows:

- U = (a+4*m+b)/6
- V = ((b-a)/6)^2

The project critical path is identified as in CPM based on the expected duration (U). Once identified, it is considered that the overall project duration follows a normal random distribution with the following parameters:

- Mean = Sum of U of all tasks;
- Variance = Sum of V of all tasks.

Based on this probability distribution for the duration of the whole project, the cumulative distribution is then used to answer two types of questions:

- what is the probability that the project is completed before day x?
- what is the project duration that ensures p% of success?

Based on this analysis, the project manager can decide about a specific project duration for the project, while being aware of the probability of success. Tasks are then scheduled as appropriate in order to meet this target. The PERT analysis can also be applied to sub-phases of the project, allowing the project manager to estimate the probability of schedule related risks (e.g. what is the probability of the design phase to be late a certain number of weeks). Re-planning actions can be devised so that the project schedule becomes more likely (i.e. higher probability of success). The overall process of PERT analysis is illustrated in figure F.12. The overall process is iterative as results from the analysis are used to improve the plan until a satisfactory outcome is achieved.

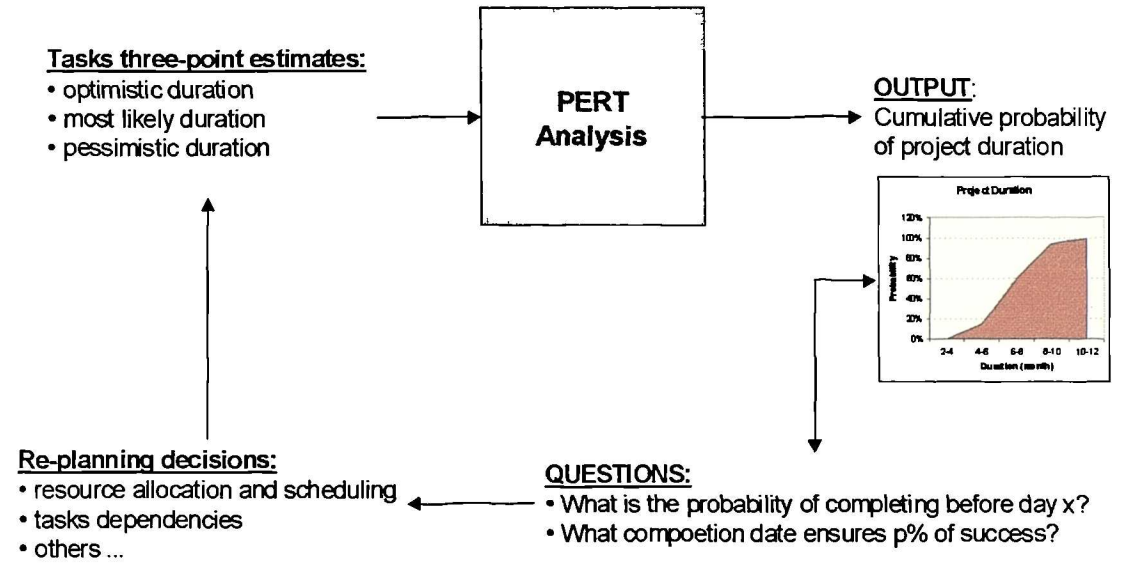


Figure F.12 – Overview of PERT analysis process

The PERT analysis technique was developed to provide the project manager with a probabilistic analysis of the project, without requiring too many calculations. However, while a useful approach, and because of its simplicity, the PERT analysis has two main limitations: (i) it models uncertainty in all critical tasks only in the form of a beta distribution, and (ii) it considers uncertainty only in the critical path based on expected tasks' duration. It can be argued that depending on the nature of the work carried out within the tasks, the schedule uncertainty does not follow always a bet type of random distribution. Other types of distributions can be more appropriate. The second limitation is more serious: if all tasks in the network have an uncertain duration, then all paths have a potential to become critical (not just the one identified as critical based on the expected deterministic values). For example, the probability of the project being finished beyond week 100 is equals the sum of the individual probability all paths in the network exceeding that date. Not surprisingly, the PERT analysis provides optimistic estimates (Nicholas 1990). These two limitations of PERT were perhaps acceptable when the computing time required for statistical calculations was very expensive. Nowadays, this is no longer a problem.

In order to overcome the limitations of PERT, the Monte Carlo simulation technique applied to the logical network became a popular alternative. The Monte Carlo technique is based on a sampling simulation process as follows:

- any task in the logical network can be modelled by any type of random distribution (from typical analytical functions, like the normal or exponential distributions, user-defined functions or frequency histograms);
- "project occurrences" are simulated by taking a sample duration from all uncertain tasks. Each occurrence is a project sample;
- for each project sample, the critical path duration is calculated and stored in a results frequency histogram;
- the project is sampled many times until the results frequency histogram becomes stable. The histogram describes the probability function of the project duration;
- the cumulative probability function is generated. A PERT like analysis can be carried out based on this function: "what is the probability of the project duration being less than x weeks?", "what is the completion date that ensures a p% probability of success?".

The iterative process of applying the Monte Carlo technique is similar to the PERT process shown in figure F.12. The main differences are in the inputs and in the calculation process that generates the project duration probability function. Instead of three-point estimates, the project manager provides a specific probability function for the duration of the uncertain task (which can be a beta-function, if appropriate). Unlike PERT, all tasks can be given an uncertain duration. The calculation of the project duration probability function is based on a sampling process that considers all tasks in the network, and thereby occurrences of all of its paths. Overall, the Monte Carlo technique allows for a potentially more rigorous and flexible modelling of the project uncertainties and produces more accurate (less optimistic) results than the PERT technique. It requires a much more computing power. Nowadays, many PERT/CPM software tools support the implementation of the Monte Carlo analysis.

Because of its probabilistic analysis, PERT and Monte Carlo are often referred to as risk analysis techniques. They can be used to identify risks and provide

important quantitative information to be included in the "risk register" described above.

Gantt charts and PERT/CPM are nowadays used as a single tool. Because PERT/CPM considers precedence relationships as an input and because tasks' schedules must respect these constraints, Gantt charts became an output of the PERT/CPM model. In other words, they are used as a friendly and intuitive way of displaying the project schedule generated by the PERT/CPM network.

Like all other techniques described so far, Gantt charts and PERT/CPM networks are not intended to be static throughout the project life-cycle. They are the ultimate description of the project operational plan. As such, they are continuously updated and readjusted to reflect the implementation of re-planning actions. Once more, it is a good practise to keep a record of past versions in the project management information system. Project performance is often assessed through a comparison against a previous version of the plan, typically referred to as "baseline plan".

The set of techniques described so far are primarily used within the project management planning function. They are first used to developed the initial work plan for the project. As this plan is implemented they are generally updated with actual results. As a new forecast of the future is produced, they are then used to re-plan the remaining project work, if necessary. The last set of standard techniques commonly used in project management are used to monitor progress. Their role is to assess and evaluate the project status against the targets.

"Earned value" and other control metrics / indices

Project control is based on the continuous monitoring of the project status, identification of deviations, and implementation of re-planning actions. The techniques described above are used to support the development of re-planning actions. There are also techniques developed for project management purposes, which are used to support the monitoring of the project status and identification of deviations. These techniques are based on the measurement of the project status and comparison against the targets. This implies the collection of data and further

production of metrics and indices. The overall technique of project control which specifies these metrics and indices is commonly referred to as Earned Value Management (EVM).

Project control can be thought of controlling individually each of the project objectives: cost, schedule, scope or functionality, and quality. Product functionality and quality are difficult to measure. Traditional project management control techniques therefore focus on cost and schedule control. Scope is also the focus of control but often this is not considered the same as product functionality.

Controlling the product functionality is aimed at ensuring that the right product is being developed. This means, the product will do what the Client asked for. This is obviously a project objective difficult to measure. This control process is based on the continuous revision of the product requirements specification document, and on the observation of the product. The requirements specifications document specifies what the product is supposed to do and is the basis of the product technical design. This document is continuously revised in internal meetings within the project team and in external meetings with the Client. Changes to this document should be reflected in changes the product under development. As the product becomes more tangible throughout the development life-cycle, its actual or potential functionality is observed and compared against the specifications. It is nowadays generally recognised that the Client should be involved as much as possible in this process. Experience shows that most problems with this project objective are due to differences in the interpretation of the requirements between the Client and the contractor, which tend to remain unperceived until the later stages of the life-cycle (Rodrigues and Williams 1998).

Scope defined as the sum of deliverables and services provided by the project is mainly controlled based on the WBS technique previously described. All the work required to developed the products and services specified in the contract must be specified explicitly in the WBS. Accomplishment of this work means that the scope of the project is also being delivered as planned. Changes in the project scope must be translated to changes in the WBS.

Controlling the product quality is aimed at ensuring that the product is being developed in the right way. This means, the product performs the functionality correctly. As an overall project objective, quality is also difficult to measure. This is mainly achieved through the implementation of a quality system. This system may already exist within the parent organisation or may be set up on purpose for the project. A quality system implies establishing a set of standards for the project regarding both the product being developed and the development process. It specifies a series of quality assurance (QA) activities, which are aimed at verifying and ensuring that the planned quality standards are being met and that quality system itself is being implemented. As part of QA, there are quality control activities (QC) which are aimed at monitoring project results and comparing them against the quality targets. QC activities focus mainly on the product being developed. Within the context of the quality system, quality planning is also an important activity, which is carried out mainly at the beginning of the project, and is aimed at identifying and establishing the appropriate quality standards for the project. The establishment of a quality system is often based on the family of ISO 9000 standards, and is required for certification. There are various techniques which can be used to support QA and QC activities. These are generally not specialised in project management and are therefore "borrowed" from the general quality management field. The more common ones include: tables of raw data about quality, Pareto analysis, correlation analysis, trend analysis and control charts (Kerzner 1998, PMI 1998).

Controlling the cost and schedule is aimed at ensuring that the project is completed on time and within budget. These are quantifiable and highly inter-related project objectives. These are the focus on the project management specialised technique called Earned Value Management (EVM). EVM is based on a bottom-up approach: the detailed tasks at the bottom of the WBS, which were scheduled in the PERT/CPM network, are monitored individually. The results are then aggregated up to the top of the project. This process ensures that deviations from the targets are first identified at the detailed level and it is therefore easier to identify the causes. For each WBS elementary task the following "status metrics" are calculated:

- *ACWP – actual cost spent with the work already performed.* This is obtained from all types of costs which were booked to the task. This is what was actually spent up to present;
- *BCWP – budgeted cost of work already performed.* This is obtained by identifying the amount of work already accomplished in the task and then identifying the budget which in the initial plan was allocated to this work . This is what should have been spent up to present if the work had cost as planned;
- *BCWS – budgeted cost of work scheduled.* This is obtained by identifying the amount of work which according to the initial plan should have been accomplished up to present, and from there identify the budget allocated to it in that plan. This is what should have been spent up to present if the work had been accomplished ad had cost as planned.

Once calculated for the elementary tasks in the WBS, these metrics can be calculated up the WBS for the other aggregate tasks. Based on these three status metrics, the following performance metrics are calculated for each task:

- *CV (cost variance) = BCWP – ACWP.* This indicates whether the work already accomplished is costing more or less than planned. A positive value indicates a cost saving and a negative value a cost overrun;
- *SV (schedule variance) = BCWP – BCWS.* This indicates whether more or less work has been accomplished against the schedule. A positive value indicates the budget value of the work, which is ahead of schedule. A negative value indicates the budget value of the work, which is behind schedule;
- *AV (accounting variance) = BCWS – ACWP.* This indicates whether more or less effort has been spent up to present than initially planned. A positive value means less effort and a negative value means more effort has been spent than planned;
- *TV (time variance) = present – t, where BCWP[t] = BCWS[present].* This indicated how much time the work is ahead or behind schedule. A positive value indicates that the work is ahead and a negative value indicates that the work is behind schedule.

The three status metrics ACWP, BCWP and BCWS are generally displayed in a graph over-time, as shown in figure F.13. This provides an intuitive overview of

how performance has been evolving in each task and in the project overall. The performance metrics CV, SV and AV mentioned above can be easily identified by comparing the vertical differences between the respective curves at any moment in time. The TV metric can be identified as the horizontal difference between the BCWP and BCWS, in relation to the present value of BCWS. In this example, after 20 days the work was ahead of schedule (around 5 days), the work accomplished was costing less than planned, and less effort was spent than planned for that moment in time.

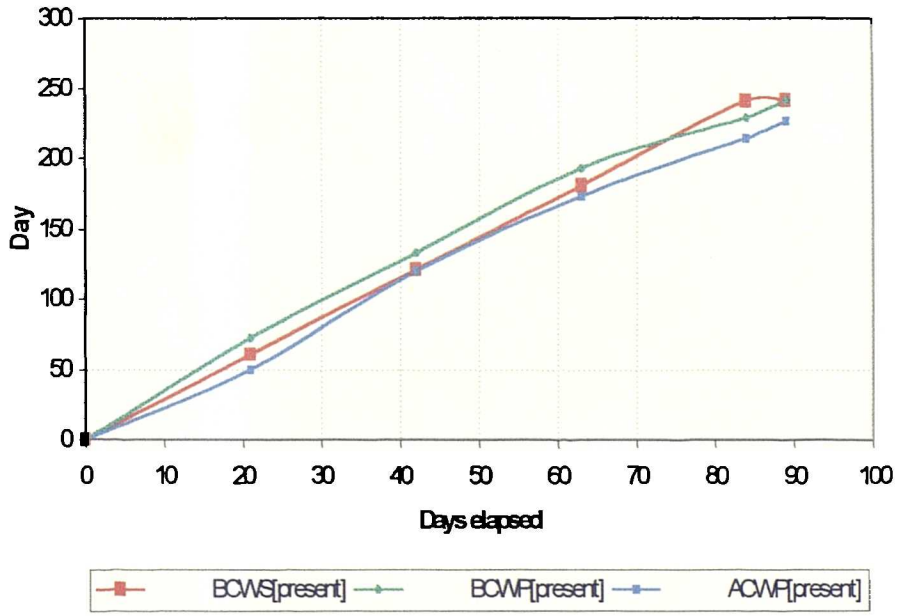


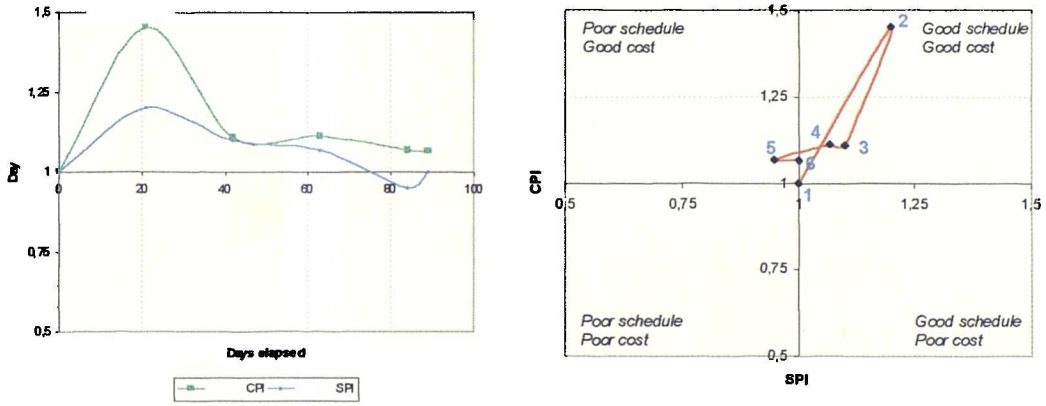
Figure F.13 – Display of project status metrics over-time

There are also two additional performance indices which are commonly used:

- *CPI (cost performance index) = BCWP / ACWP*. If greater than 1 means good cost performance, as the work is costing less than planned;
- *SPI (schedule performance index) = BCWP / BCWS* . If greater than 1 means good schedule performance, as work is ahead schedule.

These two indices provide relative measure of project performance in the two dimensions of schedule and cost. These indices can also be displayed over-time as shown in figure F.14(a). However, they provide an even more intuitive overview of joint cost-schedule performance if displayed in a XY graph, as show n in figure F.14(b). This alternative representation provides a "radar-like" display, with four

areas of performance. The ideal area of performance is at the top-left and the worse at the bottom-right. The performance as planning is at the centre at the graph. The numbers in the curve identify the moment in time when the performance as assessed and plotted in the graph. It is therefore possible to observe how the joint cost-time performance of the specific task or project evolves over-time.



(a) Plot of CPI and SPI over-time *(b) "Radar-like" XY plot of SPI and CPI*

Figure F.14 – Visual display of project performance indices SPI and CPI

Overall, these indices and previous status and performance metrics provide the project manager with a certain visibility of the project status. Because they are calculated for each task in the WBS, it is possible to know which areas of the project are exhibiting good performance and where problems are occurring. Used in this way, they can also provide some guidance to the possible causes of delays and over-runs. The overall technique is called Earned Value Management because it is based on the metric BCWP, which often referred to as the value which was earned from the initial budget with the work currently accomplished. Together with the metrics above, two estimates are generally produced: the estimated cost to complete the remaining work of the project (CTC), and the estimated cost at completion (CAC). There different ways of producing these two estimates, but the condition CAC = ACWP + CTC must be respected. The earned value is often calculated as: Initial budget – CTC. This gives a measure of how much the work accomplished so far really values against the initial budget, given the effort remaining. This is not necessarily the same as the BCWP and therefore constitutes an alternative calculation and interpretation of the concept. The overall

EMV approach was developed on purpose for project management. It focuses on cost and time, but it does not address quality and product functionality issues explicitly.

Appendix G – Detailed Project Management literature review

Overview and structure of this appendix

In this appendix a detailed literature review is carried out as the basis to discuss the state of the art in the two areas of project management and System Dynamics.

While the literature review in project management is not exhaustive, the brief description of the project management process is further extended in appendices E and F. There are two main reasons for this: first, the current state-of-art in project management has been compiled over-time in a number of good books. The ones here considered include Nicholas (1990), Turner (1993), the PMBOK (PMI 1996) and Kerzner (1998) (all well recognised pieces of work). The PMBOK is updated regularly by the Project Management Institute (PMI), USA, with the primary aim of compiling this state-of-art. The extensive description of the project management process presented in appendix E is the author's own. It is primarily aimed at providing a dynamic framework, wherein the application of System Dynamics will be easier to conceptualise and understand. The literature review also identifies some of the relevant developments, provides evidence that project failure has been a major problem, discusses and analyses the causes of this failure, and thereby tries to identify what is missing in the traditional approach. In the theoretical arena, the development of new techniques and improvement of existing ones is quite extensive. It is not the purpose of this research to cover all these developments in detail. Some relevant ones were selected, many of which are detailed in Williams (1997). Regarding evidence of project failure, there is an excellent piece of work from Morris and Hough (1987), which diagnosis some major projects. There are other studies available in the literature which provide evidence that project failure has been a problem – for this purpose, two recent surveys are referenced in this section. The aim was to identify reasonable evidence that project failure is a current problem and that the main causes are of strategic, human and systemic nature. This argument is put forward in this chapter and is a core motivation for the research here presented.

The appendix focuses exclusively on reviewing the "state-of-art" of project management. First, a brief discussion of what are projects and project management is presented. This covers the definition of some basic concepts, the main characteristics of projects, the project life-cycle, the parties involved and the project organisational issues. The following sub-section provides a review of the traditional project management framework. This covers the two main aspects of project management: (i) the process, and (ii) the tools, procedures and techniques employed within. For the sake of simplicity, the process presented is based on the standard framework proposed by the Project Management Institute (PMI 1996). The author's own view is presented in appendix E, which is important in the context of this research. A more exhaustive description of the tools and techniques identified is also presented in appendix F. The following sub-section provides a brief description of the current scenario in project management, which comprises two main issues: "management by projects" and "project failure". This highlights the urgent need for improvements. But why do projects fail? The following section provides a brief rationale discussion regarding the nature of project failure. Strategic issues and human factors are the core of this discussion. Project management is not stagnated field. Since the emergence of the first techniques in the late 1920s, further developments have been underway. Are these addressing the causes of project failure? The next section provides a review of the more relevant developments and discusses whether the real causes of failure are being tackled. The final sub-section raises the question of what might be missing in the project management discipline. It is concluded that a more systemic view is required. Recent applications of System Dynamics to project management suggest that this technique has the potential to address this need. This will be analysed in detail in the following main section of this chapter.

What are projects and project management?

Overview

This research focuses on the development of an enhanced project management methodology, which will allow organisations to implement projects more

successfully. It is therefore important to clarify at this stage the basic concepts of *project* and *project management.*

Project management is an increasingly crucial activity for most companies' businesses, and to the society in general. Social welfare and companies' success depend primarily on the success with which projects are implemented. Since the early days of humanity that projects constitute the vehicle to implement change. *Change* is here referred to in the positive and subjective sense of improving human welfare. From the pyramids of Egypt, the Greek intellectual developments and military campaigns, the Roman bridges, roads and aqueducts, through the navigation techniques first developed by the Portuguese and their sea-based military campaigns to the East, the whole expansion of the British empire, to the Soviet and American travels to the space beyond the Earth, projects have been the way through which humanity has progressed.

But what really *are* projects and project management? How different is the challenge of managing a project nowadays than it was many centuries ago? The economic environment and its demands on organisations' performance certainly makes nowadays projects much more challenging in certain ways than before. In particular, time constrains are much tighter. In order to respond effectively to these increasingly adverse conditions, a whole science for managing projects, made up of various elements of knowledge, has been developing over time. This is commonly referred to as the science of *Project Management.*

There is some excellent updated literature available where more or less detailed discussions about the concept of project and project management are presented (e.g. Nicholas 1990, Turner 1993, Kerzner 1998). It is not the purpose of this research to develop a further exhaustive conceptual analyses of these definitions nor to advance the science in this respect. The purpose of this section is just to provide a summary of the key aspects commonly agreed in the dominant literature, in principle shared by practitioners, and which are relevant for the purpose of the research herein presented. This includes the following topics: a basic definition of the concepts "project", "program" and "project management", the project main characteristics, the project life-cycle, the parties involved or stakeholders of a

project, and the organisational project issues. These topics will be briefly discussed separately in the following sub-sections. The project management process and the procedures, tools and techniques employed within this process are discussed in the following section. This constitutes the traditional project management framework,

Basic definitions: project, program and project management

There is at least one good reason to develop a rigorous definition of what a project is: to recognise those endeavours where the application of project management is appropriate. In fact, Project Management emerged as a science of its own because the application of the conventional functional management approach to certain type of efforts had proven very limited. On the other hand, applying project management to efforts which are not projects will probably produce bad results. A second reason to develop a rigorous definition is the deeper understanding it provides about the complexities involved in managing a project.

Various definitions of a project can be found in the literature. For example, the Project Management Institute (PMI) (1996) provides the following definition: "... a temporary endeavour undertaken to create a unique product or service." This is a useful simple definition, but it certainly has some limitations. It identifies two key characteristics of a project: temporary and unique. However, not all temporary and unique human endeavours are projects. In particular, the concept of uniqueness is a subtle one: on the one hand it is relative to the entity that carries out the endeavour, and on the other hand, strictly speaking, every event is unique as nothing is repeated. Kerzner (1998) proposes that a project can be defined based on the concept of a program, which NASA defines as: "A relative series of undertakings that continuous over a period of time (normally years) and that are designed to accomplish a broad, scientific or technical goal...". A project is one of these undertakings, which has a scheduled beginning and end and involves some primary purpose (Kerzner 1998). Nicholas (1990), instead of proposing a specific definition identifies a relatively long list of characteristics that every project will have. Turner (1993) explores this problem into some detail and proposes a more thorough and complete definition: "*An endeavour in which human, material and*

financial resources are organised in a novel way, to undertake an unique scope of work, of given specification, within constrains of cost and time, so as to achieve a beneficial change defined by quantitative and qualitative objectives." This definition will be used as the basic concept for the purpose of this research. A sub-set of Turner's projects will be considered, narrowing the scope of the definition down to tangible projects. This simplification is considered as useful because it creates clear boundaries for the scope of this research. The work here proposed may well be validly extended to other Turner's projects, but that will not be of explicit concern hereafter. The definition proposed for the scope of this research is therefore as follows:

Project:
a complex and unique undertaking aimed at the design, realization and delivery of a tangible product.

This definition implies that the projects herein considered deliver a tangible product. The development of this product comprises the two main phases of design followed by realization. Design comprises the work required to conceive the product and specify what it is. This includes identifying the functionality it delivers to the user and the general technical characteristics required to support such functionality. The latter phase of "realization" comprises the work required to actually "physically" build the product. In a particular project, this can be construction, production, or any another "realization" type of activity, depending on the specific industry. Within this generic definition various types of projects an be considered, like software development, conceptualisation and implementation of information systems, shipbuilding, civil construction, general R&D, among others. It is not the purpose of this research to specify the full set of real world projects which would fall within the proposed definition. It is assumed that, in the face of a specific project, this definition is objective enough for someone to decide whether it falls in the context of this research.

The term "program" is often commonly used as an alternative to "project". However, programs and projects are different. There are at least two reasons why the terms are often used interchangeably: programs often result from complex

projects, and for the sake of management the control framework used, including tools and techniques, is the same in both cases. Various definitions of program management can be found in the literature differentiating programs from projects (e.g. see Reiss in Williams 1997). Kerzner (1998) proposes NASA's definition mentioned above, and uses the terms interchangeably. On the other hand, Nicholas (1990) stresses the key differences between the two concepts. Despite the similarities, he argues, these differences have some implications for the management process which must be taken into account. Turner (1993) also examines the differences between the two concepts. He defined programs as "...a group of projects which are managed in a co-ordinated way, to deliver benefits that would not be possible were the projects managed independently." This definition stresses the important concept that a program is more than just the simple sum of various projects, and thereby that managing a program is not the same as managing a set of projects independently one another. Programs therefore have a lot to do with exploring the synergies and all types of relationships among projects, including the competition for various types of resources. Turner (1993) also stresses that programs have more diffuse objectives, hence less concrete than in projects. He further puts programs in the context of setting business strategies and directing various projects towards those strategic objectives. This implies that programs have a longer term perspective than projects. An important result of this definition is that program management involves certain management activities not present in a project alone: selecting projects, co-ordinating the projects by creating and managing interfaces, and assigning and controlling priorities among the projects. Creating programs through the development of "projects plans" is focused on the strategic management of the company's business activity. Managing interfaces and priorities is something which is not the concern of individual the management of a project. Not only programs have some differences in terms of management, as Turner's definition also suggests that what makes a program is not just the complexity of a project. Williams (1997) discusses the concept of complex projects in some depth and under various perspectives (e.g. structural, organisational, time-related, uncertainty). This discussion does not suggests that a complex project is a program. Nicholas (1990) provides a good discussion of the practical differences between the two concepts, which he summaries in three aspects:

- programs extend over a longer time-horizon (five or more years);
- projects as individual events are dissolved after the end-product is delivered. On the other hand, program tend to cover the whole life-cycle of the various systems delivered, so that the strategic objectives can be achieved;
- program management requires the explicit consideration that the project manager will change over the course of the program. No one single person can therefore be accountable responsible for the outcome.

In this research, the object of analysis is the individual management of a single project. However, just like most elements o the project management framework are applicable to program management, most of the output and conclusions of the research here presented is transferable to the management of a program.

What is project management? The PMI proposes the following definition: ".. the application of knowledge, skills, tools and techniques to project activities in order to meet or exceed stakeholders' needs and expectations of a project." In simpler terms, project management is a continuous undertaking primarily aimed at ensuring that the project is completed within the objectives, and which takes place within the context of the project. This definition has some important implications: (i) it requires effort, (ii) it is focused on the project objectives, (iii) it takes place continuously throughout the project life-cycle and (iv) it is part of the project implementation process.

The PMI's definition also suggests that the ultimate objectives are to meet or exceed stakeholders' expectations. It also suggests the employment of a certain pre-existing knowledge about the problem.

Regarding the objectives, most project management literature refers to the project objectives as comprising cost, time and quality (Nicholas 1990). This implies an operational perspective of the management process, focused on the product. Turner (1993) suggests that the project objectives should be broader, also including scope and organisation. Whether the particular objectives of a project refer to product related issues, to the business outcome, or even to social issues, it will be considered in this research that they can be mapped into four main

dimensions: cost, time, requirements (what was achieved) and quality (how well was it achieved).

Turner (1993) argues that the management process can be considered at three different levels: integrative, strategic and operational. The first level addresses the need to have the project outcome in line with the company's business objectives. The second level has to do with devising the right strategies within the project, so that it progresses as desired. The operational level has to do with specifying and controlling the specific tasks required to accomplish the project work according to those strategies. This research will focus on the last two levels of project management.

Just as the PMI assumes the pre-existence of some knowledge about how to manage a project, Turner (1993) also argues in favour of a process. He identifies various approaches, from the classical view of generic problem solving, to the "project-specialised" view focused on the transient nature of a project, which follows a generic life-cycle. This integrated view of process and knowledge (including tools and techniques) leads to the traditional project management framework, which is acknowledged and shared by most authors and organisations. This framework is the foundation upon which the various knowledge areas are developed and equipped with tools and techniques. For the purpose of this research, this traditional approach will be assumed, as described in the next section.

The main characteristics of a project

As important as recognising that we are in the presence of a project, it is also essential to be aware of some of the key project characteristics. These have important management implications.

Projects have certain specific characteristics which altogether make them different from other types of human endeavours. The distinctive characteristics of projects are typically contrasted with the other type of common human activity: routine operations. As already mentioned, projects are the vehicle for humans to

implement change. In contrast, routines can be seen as a way to preserve what is created with projects. As complementary human efforts, both project and routines form the core of human activity which creates the continuous, progressive and sustainable change in which we live in.

Nicholas (1990) identifies the following main characteristics in a project:

- a single definable purpose measured in terms of cost, schedule and performance requirements;
- cuts across organisational lines because of complexity;
- uniqueness;
- unfamiliarity;
- temporary;
- it is a working process that follows a life-cycle.

These are all characteristics which one would recognise to be present in most projects. They are necessarily related to the definition of the concept discussed in the previous subsection. However, what are the crucial ones? The important issue to address is to consider those which have a great impact on our ability to implement a project successfully.

Turner (1990) contrasts the characteristics of project against the characteristics of routine operations, as shown in table G.1 below. At the core of the differences is the fact that projects are *unique* endeavours, whereas routines are repetitive. Some of the project characteristics are also present in the list presented by Nicholas (1990). In developing the table, Turner (1990) makes reference to other various authors which also attempt to identify those characteristics that make something to be a project.

Projects	Operations
Unique Finite *Revolutionary Change* Disequilibrium Unbalanced Objectives Transient Resources	**Repetitive** Eternal Evolutionary Change Equilibrium Balanced Objectives Stable Resources
Flexibility Effectiveness Goals	Stability Efficiency Roles

Table G.1 – Projects versus operations (Turner 1990)

Turner (1990) also argues that these different characteristics bring about some important "cultural" differences, as shown in table G.2 below.

Projects	Operations
Flexible Environment	Stable Environment
Effectiveness must be present to the achievement of the objectives	Trough HII* become increasingly more efficient
People's behaviour is goal oriented	Each person fulfils a role defined by precedent

* - Habitual Incremental Improvement

Table G.2 – Projects versus operations: cultural differences that arise from the different characteristics (Turner 1990)

Perhaps a key issue missed out in these discussions is that some of the most important characteristics of a project are relative to the entity which will carry out the endeavour. For example, let us consider what appears to be one of the crucial characteristics: a project is unique. However, ultimately every human effort is unique. Therefore, for an effort to become a project it has to be "unique enough". This means, different enough from everything else possibly similar that has been done before. Uniqueness is not a binary concept. Instead, it belongs to a scale. Figure G.1 below illustrates this concept.

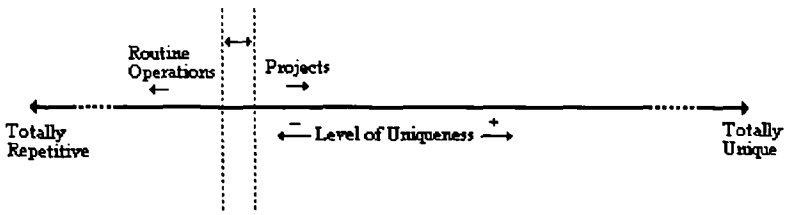


Figure G.1 – The level of uniqueness as a determinant of a project

A second issue about uniqueness has to do with the potential different dimensions of the concept, like:

- the product or service to be developed;
- the objectives imposed on the project;
- the entity or entities that carry out the project;
- the Client and other stakeholders with interest in the project;
- the work required to accomplish the project;
- the way in which the project work is going to be accomplished;
- the location, society, legislation and other environmental conditions within which the project is going to be accomplished.

In many cases, it may be necessary that only one of these dimensions brings enough uniqueness to an endeavour so that it can be considered as a project (and thereby managed as such). Nevertheless, uniqueness on its own does not turn a human effort into a project.

Another crucial characteristic present in these studies is the concept of complexity. A project is complex, which has various implications in the way it needs to be accomplished. For example, it requires some level of organisation. However, this is another relative concept. Like uniqueness, complexity can be considered in various dimensions: the product can be the source of complexity; or a Client with unclear requirements and disruptive behaviour can be the source of complexity; or it can be that complexity stems from the conflicts of interests among the various stakeholders. Complexity brings about other characteristics of projects, like the fact that it crosses various organisational lines. However, like uniqueness, complexity on its own does not turn a human effort into a project. For example, some routine operations performed in certain production systems are complex. But they are not projects because they have been repeated many times before (i.e. in a very similar way), and so there is knowledge, *a priori*, about how to best implement them.

However, combining complexity with uniqueness leads to certain types of human efforts most of which will be projects. Doing something never done before, which is

difficult to anticipate how to do it with success, looks like a project. Being both relative concepts, complexity and uniqueness can be considered as generating a two-dimensional space of human efforts. Within this space, some organisations will classify some human efforts as projects. Two different organisations may classify the same human effort differently. This is illustrated in figure G.2. All projects need a minimum level of complexity and uniqueness (MinComp and MinUniqu respectively in figure G.2). Depending on their personal perceptions, the organisations which will invest effort in implementing the human effort (referred to as "investors" in figure G.2), will position such effort somewhere in this two-dimensional space of "complexity x uniqueness". A possible line in this space separates operations from projects. For a certain human effort to be a project , the investor must perceive enough complexity and uniqueness so that this is positioned beyond this line. For example, in figure G.2 investor A considers a certain human effort as a project whereas investor B considers it as routine.

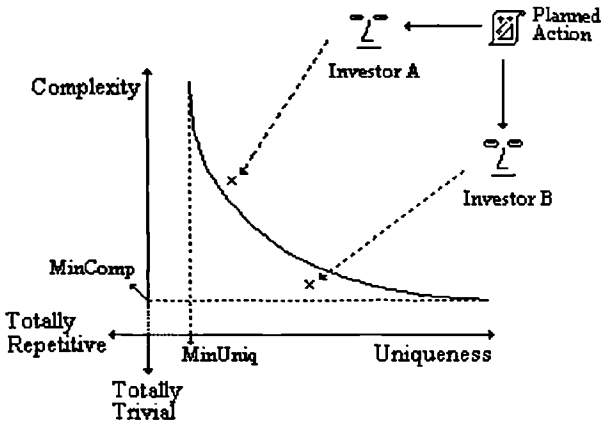


Figure G.2 – Projects within the space of complexity and uniqueness

Complexity and uniqueness bring about a project, but they are not sufficient conditions on their own. Another crucial characteristic of a project identified by both Nicholas (1990) and Turner (1990), is that projects are finite endeavours. This is a very objective characteristic, thus very easy to identify. However, it also has important managerial implications and brings about other project characteristics. For example, the project team, which is organised in a certain way, will be disbanded in the end of the project. The temporary nature of a project is also strongly linked to the fact that projects follow a life-cycle, as argued by Nicholas (1990). This is another important characteristic of a project. There are various

possible structures for a project life-cycle, but in general it will cover the phases of growth, maturity and death.

Also related to their temporary nature, is the fact that projects have well defined and finite objectives. A crucial characteristic of projects is that at least four of these objectives, cost, time, quality and requirements, are unbalanced, as pointed out by Turner (1990). This is, they compete one another. In order for a better performance to be achieved in a certain objective, one or more of the others will have to be sacrificed. This important characteristic of projects is illustrated in figure G.3. "Pressing" in one of the objectives means taking an action to improve the project in that dimension. As a secondary result, one or more of the others will be "pushed back" (meaning poorer performance).

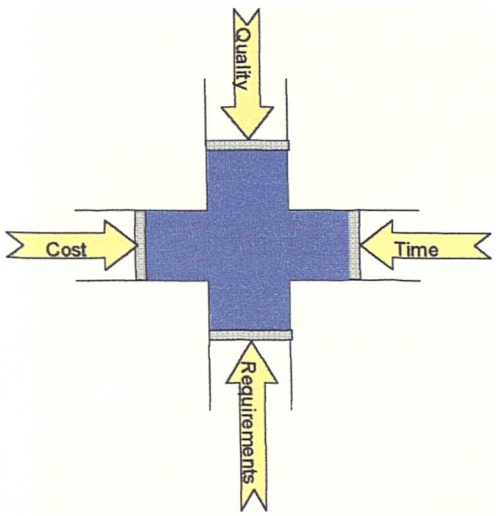


Figure G.3 – The four competing objectives of a project

This particular characteristic has a tremendous impact on the management process of a project. Managing a project consists in controlling its outcome towards the specified objectives. Once a deviation in one of the objectives occurs, the project manager will try to take actions in order to improve the project performance in that dimension. Because the objectives are unbalanced, this means that almost always as, and a secondary impact, the project will deviate from one of the other objectives (e.g. reducing the estimated completion time by recruiting more resources, or by reducing the quality of the product).

Uniqueness, complexity, finite and unbalanced objectives of cost, time quality and requirements are proposed here as the primary characteristics which make a human effort to be a project. It is important to note that the objectives of a project are related to the achievement of a desired change. As a result there are various other characteristics of projects which have important implications in terms of management. Among these there are two worth pointing out: (1) projects require temporary resources, and (2) for a project to be feasible it requires that work and resources are planned for and are organised. The temporary nature of the resources is critical because it implies that their allocation to the project needs to be planned according to the work requirements. This way, if changes to the planned work progress occur then the currently planned allocation of resources tends to become counter-productive and needs to be adjusted immediately. Responding quickly and effectively to the new changing demands is not easy and, in general, is costly. Regarding the need to plan and organise, in general, it is only possible to develop the project's product if work and resources are subjected to a minimum level of organisation. Not all work can be accomplished at once and not all resources can work at the same time. There are work precedence relationships that need to be respected and tasks need to be accomplished by groups of individuals working as teams. Without a minimum level of organisation, problems with work accomplishment would cause the project to collapse. The immediate impact of this characteristic is that a detailed work plan needs to be devised, monitored and readjusted as required.

A final characteristic of project which is worth mentioning is the fact that projects are often subjected to the interests of various stakeholders. In many cases, the Client to whom the product will be delivered is not the only stakeholder in the project. In the first place, the Client can be made up of various entities, persons or organisations. Secondly, within complex social and political environments some projects will have an impact on many other parties, than just the Client. For example, ecological organisations, unions, social organisations, economic lobbies, governments, competitors, partners and alike may all have an interest on the project outcome. Stakeholders can have different levels of importance and of power over the project. Often, stakeholders which are not considered as important in the light of the project objectives, but which are powerful (i.e. can affect the

project), must be given much attention. The main impact in terms of management is that it requires flexibility. The project manager must follow flexible plans capable of accommodating changes that stem from the various pressures and needs of the stakeholders.

Table G.3 summarises the main project characteristics and their impacts on the management process. Most of the difficulties inherent to managing a project result from these impacts: the uncertainty, the difficulty in anticipating outcomes, the tight deadlines, the trade-offs that need to be made, and so forth. In great part, this unstable scenario wherein projects stake place is what makes successful project management a major challenge. It has caused the project management discipline to emerge, moving away from the traditional functional-based management (Nicholas 1990).

Characteristic	**Impacts on management**
Unique	*Uncertainty* about the problems to be faced and about the outcome of decisions
Complex	*Difficulty* in anticipating impacts of actions and thereby in identifying effective solutions to problems
Finite	Imposes intermediate *deadlines* which get tighter as the project progresses
Life-cycle	Imposes a *sequence* on the type of work and different problems that need to be addressed over-time
Unbalanced objectives	Decisions need to be devised on a *trade-off* basis among the objectives
Temporary resources	Requires continuous *resource planning*
Requires organisation	Requires a *detailed plan*, to be monitored and updated as required
Various stakeholders	*Flexibility* to accommodate conflicting interests and consequent changes to the objectives

Table G.3 – Main project characteristics and their impacts on management

The project life-cycle

By their nature, projects are transient undertakings. The whole course of events taking place throughout a project changes over time. For example, the resources employed, the type of work being performed, the key managerial concerns, the sub-products being produced, and the priorities between cost, time and quality. In order to cope with this transient nature of projects, life-cycles are established (more or less explicitly), by those who undertake a project. The life-cycle followed by a project is in part a natural and inevitable phenomenon. At the same time, it is also the result of a management decision.

Management establishes a life-cycle with the aim of addressing important project requirements. This decision will have a major impact on the project outcome (e.g. the profile of resources required over-time; the deliverables to the Client). One of the most important issues addressed by a life-cycle are the technological requirements and constraints for developing the specific product. As a consequence, typical project life-cycles tend to be industry-specific. Nevertheless, within a same industry different life-cycles are often adopted. For example, in the software industry there are various software process models which lead to different types of project life-cycle. For example, the spiral model, the RAD model, and the OO model, among others (Pressman 1997). Different life-cycles within a same industry reflect the different circumstances to which the project needs to respond (e.g. the Client; the type of product; the degree of innovation).

Specific project life-cycles can be described in more or less detail. In general terms, the description of a life-cycle will comprise the following elements:

- a sequence of phases, each with well defined start and end points;
- a description of the type of work to be accomplished within each phase;
- the inputs required to start each phase (typically, intermediate sub-products from the previous phase);
- the outputs to be produced by each phase, to be delivered to the following phase and/or to the Client (e.g. at the end of the project).

The project life-cycle is often confused with the product development life-cycle. Despite the similarities, it is important to understand that they are different concepts. The product life-cycle addresses the phases required for the technical development of the product. This is only a part of the project, which is a broader event. For example, the detailed planning of the project is part of the project life-cycle and takes place prior to the product starts being developed. Furthermore, while technical product development is underway, there is a considerable amount of work being undertaken which does not have to do directly with the technical aspects of product development. Therefore, the product development life-cycle has a narrower scope. In terms of time-scale, it generally falls within one or more phases of the project life-cycle.

There is no project life-cycle universally agreed within the project management community. Various alternative descriptions can be found in the literature, with more or less differences. The PMI (1996) proposes a set of representative life-cycles for various industries: defence acquisition, construction, pharmaceuticals and software development. Turner (1993) proposes a life-cycle based on a management perspective, which comprises four main phases:

(1) *germination* – proposal and initiation. Focuses on defining the problem, assessing the project feasibility and taking a go / no go decision;
(2) *growth* – design and appraisal. Focuses on refining the solution through a more detailed system design. A baseline plan is developed;
(3) *maturity* – execution and control. Focuses on the progressive detailing of the plan, execution and control;
(4) *death* – finalisation and close-out. Focuses on the delivery and installation of the system, and all related close-out activities (e.g. audit and review).

This life-cycle is generic enough to accommodate most of the specific life-cycles followed in real projects.

Kerzner (1998) points out that while for the product development life-cycle there has been a partial agreement about a generic description, the same is not true for the project life-cycle. He argues that this is understandable because of the diversity and complexity of projects. Interestingly, for the product development life-

cycle, Kerzner (1998) proposes a life-cycle with the following phases: growth, maturity, deterioration and death; this is very similar to the project life-cycle proposed by Turner (1993), as described above. Kerzner (1998) argues that a generic definition for the project life-cycle can be based on the systems development life-cycle proposed by Cleland and King (1975). This includes the following phases: conceptual, definition, production, operational, and divestment.

Nicholas (1990) also proposes a project life-cycle based on the general systems development approach, which comprises the following main phases: conception, definition, acquisition and operation. This is very similar to the life-cycle proposed by Kerzner (1998), the only relevant difference being the final divestment phase, which Nicholas (1990) considers explicitly as part of the operation phase. These two authors describe the activities taking place within each phase of their proposed life-cycle in some detail. It is not the purpose of this research to review and analyse these descriptions in detail. These two systems' based life-cycles can also be easily mapped to Turner's description (1993).

For the purpose of this research it is not relevant to impose a specific project life-cycle. It is sufficient that the life-cycle adopted for a specific project can be clearly mapped to one of the three described above. The project life-cycle hereafter considered as a reference is the one proposed by Nicholas (1990):

(a) *conception* – this phase focuses on the problem. A clear definition of the problem and of its scope is developed. A preliminary system concept is proposed as a possible solution. Feasibility analysis takes place to support a go / no go decision. A request for proposal (RFP) is issued, proposals are evaluated and a contractor is selected;
(b) *definition* – this phase focuses on the solution. The system concept is refined into a preliminary system design. An initial detailed plan is developed for the project, specifying the organisation, work tasks, schedules, and resource allocation;
(c) *acquisition* – this phase focuses on the system development and installation. A detailed design is developed and production (or realization) takes place. At the end, the system is installed at the Client site, factory acceptance test (FAT) is carried out and the system's users are trained;

(d) *operation* – this phase focuses on the use of the system. The system has been deployed. The user operates the system with aim of solving the initial problem. The contractor may be more or less involved providing maintenance and evaluation services. The system is continuously evaluated until an improvement or replacement decision is taken. This eventually leads to a new project.

The research work herein developed addresses the project management needs throughout the whole life-cycle. However, more emphasis will be given to the phases of Definition and Acquisition. In particular, the development of a detailed initial plan, prior to the product starts being developed, and the process of implementing the plan and controlling progress, while the product is being developed, prior to final operation at the Client site.

The parties involved

An important project characteristic is the fact that it tends to involve various different parties, each with its own perspective and interests about the project outcome. It is important for project management to be aware of and consider all the relevant parties. While some of these may not be too obvious, perhaps because they do not have a direct involvement in the project work, they can sometimes exert a great influence over the project. Managing the potential and often subtle conflicts among the project parties is a key factor for project success.

Who are the key parties and which are the most critical potential conflicts, depends in great part on the specific project and environment. First, it is important to be aware of the potential relevant parties. However, these parties can be of many different types, from the Client to a government or a competitor. For this purpose, a generic list is desirable. The PMI (1996) proposes a set of main "project stakeholders":

- *project manager* – the individual responsible for managing the project, ensuring that the objectives are achieved at the project level;

- *customer* – the user of the project product. This can be an individual, an organisation or a larger set of people. It can contain various sub-groups or layers;
- *performing organisation* – the enterprise who carries out the project work;
- *sponsor* – the entity within the performing organisation who provides the financial resources necessary to carry out the project;
- *other stakeholders* – more or less directly involved and affected by the project.

Turner (1993) also analyses the importance of the various project parties. He considers two main parties: the owner and the contractor. The owner is the party who will use the product and specifies its requirements. It is also the party who provides the finance to undertake the project – this includes the sponsor and the customer in the PMI classification. The contractor is the party who carries out the project work and consumes the finance. The contractor organises the resources required to accomplish the project work and delivers the product – this includes the project manager and the performing organisation in the PMI classification. Turner (1993) further breaks down these two main parties as follows:

(A) owner:

- *sponsor* – provides the finance;
- *champion* – senior representative who convinces the sponsor about the importance and priority of the project;

(B) contractor

- *manager* – the person responsible for managing the project;
- *integrator* – the person responsible for ensuring that the various teams are able to work together.

In addition to these two parties, Turner (1993) further identifies other important entities:

- *users* – the ones who will use the product;
- *supporters* – the ones that provide goods essential to implement the project work (e.g. sub-contractors);
- *other stakeholders* – parties which are not directly involved in the project and who will not benefit directly from it. However, the project implementation and final outcome has an impact on their lives.

The users are the customer in the PMI classification. The supporters can be considered as "other stakeholders" in PMI classification.

Clearly, the parties involved in the project can be classified and grouped under various different criteria. One of the difficulties in carrying out this task is the fact that each party can perform various roles, while different roles are generally identified with different parties. For example, in an internal project the owner can also be the contractor. In other cases, the user and the owner are the same.

For the sake of simplicity, in this research the following generic classification will be hereafter assumed:

(1) *Contractor* – the organisation who plans and executes the project work and delivers the product. It includes:
   - *executive managers* – senior business managers of the parent organisation. For them, a particular project is part of the business and is only successful if the outcome is aligned with and contributes to the core business objectives. They have the power to cancel the project and prioritise various projects;
   - *project manager* – the person responsible for managing the project within the objectives specified at the project level;
   - *first-line managers* – managers directly responsible for assisting the project manager. They are generally responsible for a certain management function, like Quality Assurance, Human Resource Management, sub-contracting, Client manager, among others;
   - *technical staff* – the staff directly involved in developing the product (e.g. software engineers);
   - *administrative staff* – the staff that provides administrative support to the rest of the project;

(2) *Client* – the user and buyer of the product. Specifies the requirements, uses the product, oversees the project progress and provides the finance;

(3) Sub-contractors – entities to which the contractor outsources work or some form of deliverables. They provide deliverables to the contractor which are integrated or otherwise incorporated into the product;

(4) *Other stakeholders* – all other parties less directly involved in the project implementation and output product, but which have two characteristics: the project affects them somehow, and they have power to affect the course of the project (e.g. ecological organisations, unions, governments).

This proposed generic classification is not intended to be exhaustive. It focuses on those aspects which can affect project implementation, at the project management level. This is, effective project management should be able to establish effective communication channels and cope with the difficulties. The breakdown of the contractor gives emphasis to the parties internal to the project, which are the ones more directly under the influence of project management. One of the key strategies to prevent problems from conflicting interests is to achieve effective communication among all parties, a major difficulty in most projects (e.g. Rodrigues 1999). As it will be seen, improving communication within project control is one of the key issues addresses by the current here presented.

The project organisational issues

Projects are implemented by organisations. A project organisation comprises the various human resources which will accomplish the project work. These organisations are generally structured in a certain way. A chosen organisational structure groups individuals into various teams and establishes relationships of authority and responsibility among them.

Because of their complex and unique nature, projects require that an appropriate organisational structure is established, a key factor for project success. This structure must respond to the particular needs and aspects of each project. An organisation comprised of teams with incompatible individuals, ineffective team leaders, unclear lines of authority, or where responsibilities for the project work are vague, is unlikely to carry out the project successfully. In addition, project environments give few scope for managers to re-structure the organisation frequently and to recover from bad organisational decisions.

A range of factors must be taken into account for management to establish an organisational structure for the project. Over the years, some generic types of organisations have been established. There is a natural tendency to map these generic structures to the scenarios where they are likely to be more effective. These scenarios are characterised by some key factors, in particular the project complexity and the frequency with which the parent-organisation carries out projects (Nicholas 1990).

The PMI (1996) proposes two extremes for the range of possible organisational structures: the functional organisation and the "projectized" organisation. Functional structures are typical of non-project based organisations. These structures breakdown the organisation according to the "classical" functional approach to management, regardless of the specific project. In this scenario, the organisational structure already exists prior to the project being considered. The mission of the organisation is to respond to the company's core business activity. In general, when such organisation implements a project, only some of its members are allocated to the project work. With this type of organisational structure, establishing an effective project management system is more difficult, because the organisation is not tailored to answer the specific needs of the project. This is the approach typically followed by organisations with a weak project culture, or with little knowledge about the project management science. For these organisations, implementing projects is not their "way of life" nor is at the core of their business activity. The two main characteristics of a functional organisation are the fact that authority over the project staff rests with the functional managers, well above the project manager (who plays the role of a project co-ordinator); the other characteristic is that most of the staff (if not all), including the project manager, are not assigned full-time to the project.

"Projectized" organisations are at the other extreme. They are built on purpose to carry out the specific project and therefore they try to address all of its specific needs. In this scenario, the project manager has full authority over the project staff and everyone is practically allocated 100% to the project. As a consequence, most of the staff is "recruited" on purpose for the project. This type of organisational structure makes the implementation and practice of a real project management

system easier and more likely to be effective. In general, organisations decide to adopt this type of structure in order to undertake major complex projects, which are the core of their business activity. Their way of life is often to implement this type of projects.

Between these two extremes, matrix type of organisations are considered. In this type of organisation, the project team is made-up mainly of members of the existing parent organisation, which has itself a functional structure. This staff is generally allocated a considerable amount of time to the project (around 50%). An important issue is that they report to both the functional manager and the project manager. In a perfectly balanced matrix organisation, the project manager has a similar level of authority as the functional manager. This raises the possibility of difficult conflicts. Nevertheless, the project manager often belongs to a functional department of the parent organisation and hence reports to a functional manager. Matrix organisations are typically used when an organisation is involved in various projects at the same time. Crossing the existing functional structure of the parent organisation with all on-going projects, becomes an effective way to respond to their needs, while keeping a single organisational structure. Nevertheless, each project will have its own project manager with relative authority over the project team. Matrix organisations can be more or less shaped towards either a "projectized" or a functional organisation. Intermediate classifications like "strong matrix" and "weak matrix" are considered respectively (PMI 1996).

In practice, various types of organisational structures can exist within a parent organisation. Depending on the characteristics and importance of each project, a different type of organisational structure can be adopted. A small project can be implemented by the functional parent-organisation, while a major complex project may require the build-up of a "projectized" organisation.

Different types of organisational structures have different impacts on the project outcome. In particular, the organisational structure may restrict the implementation of a specialised project management system. While the "projectized" organisation would be the ideal answer to each project, practical restrictions need to be

considered, in particular all the overheads and fixed costs required – typically, these organisations are expensive to build and maintain.

Nicholas (1990) considers some other variety of organisations, based on the three main types mentioned above. These are as follows:

(1) *pure project organisation*:

- *pure project* – as "projectized";
- *project centre* – the creation of a "projectized" organisation as an "arm" within the functional parent-organisation. Most of the staff is "borrowed" from the parent-organisation, as required;
- *partial project* – the core of the project work is implemented by a "projectized" organisation, and the remaining is implemented by the parent organisation under its functional structure;

(2) *matrix*:

- *temporary matrix* – exists only temporarily while the projects are underway;
- *permanent* – tends to be the way in which the organisation works;

(3) *functional*:

- *single function task force or team* – the project takes place and is implemented only within a specific department;
- *multi-functional task force or team* – the project tram comprises elements from various departments.

Based on this classification, Nicholas (1990) proposes that the adequacy of the organisational structure depends on three main factors: complexity / size, frequency and duration. Table G.4 below shows how the organisational structure should be considered depending on these project characteristics.

Complexity and Size	*Frequency*			
	Infrequent		**Frequent**	
	Duration		*Duration*	
	Short	**Long**	**Short**	**Long**
High	NA	Partial project Project centre	NA	Pure project ("Projectized")
Medium	Multi-functional	Temporary matrix	Multi-functional	Permanent matrix
Low	Single functional	Single function	Single function	Single function

Table G.4 – The organisational structures as a function of the project's characteristics

Functional type of organisations are more appropriate for simpler projects, which are not part of the organisation's core business activity. Matrix organisations are appropriate when projects are considerably complex and involve longer time-frames. "Projectized" organisations are the answer to complex projects, in particular when these are the "way of life" of the company.

Turner (1993) proposes a classification similar to the one of the PMI (1996). However, this author relates the organisational structure with two crucial elements of the project management system: the OBS (organisational breakdown structure) and WBS (work breakdown structure) – these elements will be explained in detailed in the following sub-section. The type of organisational structure chosen, dictates the way in which these elements are developed. This is illustrated in table G.5.

Organisational structure	**OBS**	**WBS**
Functional	Independent from project	Tailored to fit existing functional OBS
Balanced matrix	Independent from project	Independent from existing functional OBS. Tailored for the project
Project ("Projectized")	Tailored to fit the WBS	Independent from existing OBS. Tailored for the project

Table G.5 – The impact of the organisational structure on the OBS and WBS

As expected, in the functional organisation a functionally oriented OBS already exists, specifying how the organisation operates their normal business. In this case, the project work is disassembled into sub-tasks with the primary purpose of fitting the OBS, with each task falling under full responsibility of a certain department. In a sense, the project is decomposed into normal "business" activities. On the other hand, in a "projectized" organisation the work is specified and decomposed according to the project characteristics. The aim is to ensure an easier understanding of the project scope and the management of the individual tasks (here, the breakdown is often product oriented, but it can also be client or geographically oriented if appropriate). The OBS is then developed to address the "work logic" of the project. Finally, in a matrix organisation both WBS and OBS are developed independently one another. The project tasks will often be performed by various functional departments, which in turn work in various projects.

While there is no universal rule that determines the most appropriate organisational structure for a project, it is important to be aware of the various options available. It is important to understand what motivates their uses and what are the impacts on the project management system. On the one hand, management must be flexible to adjust this structure to the specific circumstances of the project. On the other hand, the organisational structure must be clear enough so that unambiguous and non-conflicting relationships of authority are established within the project team.

The nature of projects and project management were just described. The following sub-section, provides a brief overview of the existing standard project management framework.

A review of the traditional project management framework

Overview

Since project management first emerged as a discipline in the late 1920s, with the first uses of Gantt charts (Nicholas 1990), the ever growing community of practitioners and researchers has been developing a well established project management framework. This framework comprises various elements, some

unique and specialised in project management, while others have been borrowed from other management and engineering disciplines. These elements include various generic processes, practices, procedures, tools and techniques.

Various national and international project management associations have been attempting to develop their own standard project management body of knowledge (PMBOK) (Wirth and Tryloff 1995). The first association to produce this type of work was the US based Project Management Institute (PMI) (Duncan 1995), which in the long term was intended to become a world-wide standard. The other major association to propose a similar type of work was the UK based Association for Project Management (APM) (Willis 1995). The APM is currently a national member of the International Project Management Association (IPMA), which is mainly based on Western European countries but also includes various national association world-wide, including the US. The way in which these and other associations describe a proposed standard project management framework differs mainly in terms of perspective. This leads to the proposal of different knowledge areas and how these are further divided into sub-elements (a comparison can be found in Wirth and Tryloff 1995). However, the underlying general project management process is common to all of them and so are the main tools and techniques employed. For the sake of a brief description, the PMI perspective will be assumed (PMI 1996).

By knowledge about project management it is here meant processes, practices, procedures, techniques and tools, or any other element which portrays a structured understanding about particular aspects of project management.

The PMI (1995) proposes that the project management body of knowledge includes three types of elements:

(1) *unique and specialised in project management* – these elements were developed "on purpose" to satisfy the requirements of the project management practice and are therefore unique to the project management science;
(2) *from the general management practice* – these elements "are borrowed" from the more general management practice and therefore are not unique to project management. They were developed to answer needs faced by organisations

on their normal business activities. For example, human resource management procedures and techniques, and quality management tools and techniques. These elements are also useful to the successful management of a project;

(3) *from application areas* – these elements were developed in specific areas of product development and are not common to all projects. They often belong to an engineering discipline. For example, software engineering provides an extensive set of processes, tools and techniques which are critical to the successful management of software projects.

The PMI (1996) further proposes that the body of knowledge is primarily comprised of elementary processes. These processes are described in terms of inputs, tools and techniques used within, and outputs delivered. Based on their inputs and outputs, they are linked to together as building blocks to implement the main activities of the project management process: initiating, planning, executing, controlling and closing.

This process-based perspective is briefly described in the following sub-section. While major organisations like the PMI and IPMA have been trying to develop a common description through their own PMBOK, it is not the purpose of this research to reproduce these views in detail. Perhaps because project management is essentially a practical discipline, the many descriptions available in the literature are based on personal perspectives, according to their authors' own experience. For the sake of reference and simplicity the PMI's perspective is presented (PMI 1996).

The author's own view of the project management process is further presented in appendix E, in more detail. Some of the ideas may not be found in the literature described in the same way, where a linear and more static view is often adopted for pedagogical purposes. The main purpose of the description presented in appendix E is to provide a dynamic framework of the project management process, wherein continuous iteration, refinements, rework and interactions play a major role (as they do in the real world). Later in this research, this dynamic framework will allow for an easier understanding of the use and integration of System Dynamics models.

The project management process

The project management process is characterised by a logical flow of managerial actions, which aims at keeping the project outcome within the objectives. There is no universal and generic description of this process adopted by all practitioners and accepted as valid. While the descriptions available in the literature have many elements in common, most of them are not formal and are based on their author's own views and experience. The most standard descriptions are the ones proposed by project management associations. It is not the purpose of this research to analyse the differences and discuss their validity. For the sake of reference and simplicity, the PMI framework is here described. The author's own view of the project management process is described in more detail in appendix E, which provides an important perspective for the purpose of the present research.

The PMI (1996) proposes a view of the project management process in which various elementary sub-processes take place within the two main processes of management and engineering. These elementary sub-processes are grouped according to five main types of activities: initiating, planning, executing, controlling and closing, which are inter-related as show in figure G.4.

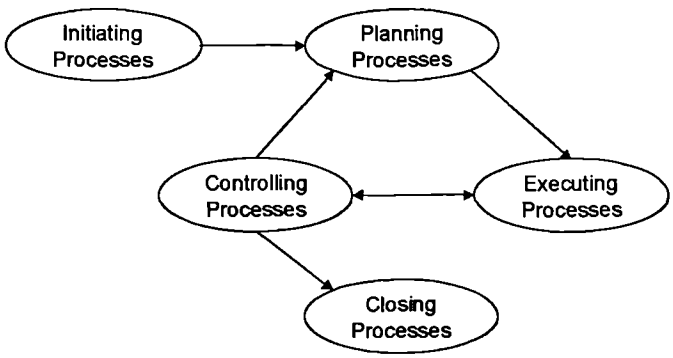


Figure G.4 – Main project activities (process groups) and their interactions (PMI 1996)

This view is similar to the one presented in figure E.1 (see appendix E), except that the initiating and closing processes are considered explicitly. These would correspond to the first implementation of the planning function and the last implementation of the monitoring function respectively. The controlling process

corresponds to the monitoring function (it is assumed in this research that control includes both monitoring followed by re-planning).

This set of activities (or process groups, as referred to by the PMI), take place throughout the whole project life-cycle. Therefore, they take place within each project phase and also interact across these phases, as shown in figure G.5 below.

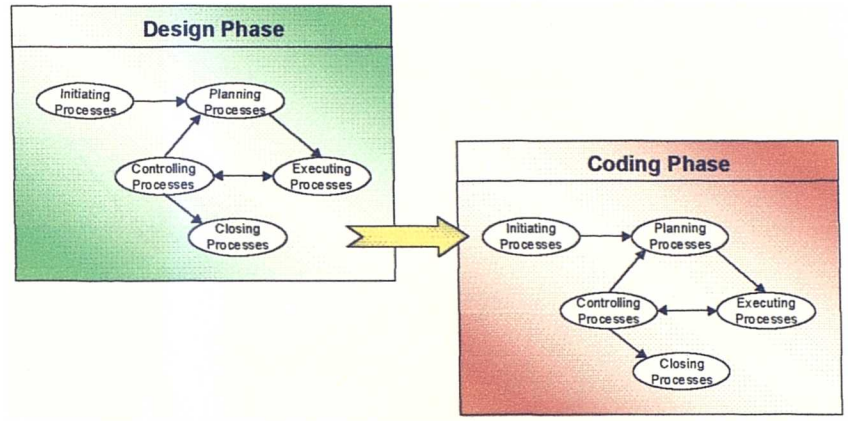


Figure G.5 – Interactions between elementary processes across phases

The PMI considers a total of 37 elementary processes. Each takes place within a specific project activity. There are initiating processes, planning processes, executing processes, controlling processes and closing processes. Within each activity, some elementary processes are considered as "core" processes, while others are considered as "facilitating" processes. The elementary processes belong to only one type of activity but like the activities they can take place in the various project phases. Within the activities, the elementary processes are linked through inputs and outputs (as suggested by figure G.4). The PMI describes these processes in terms of inputs required, tools and techniques used, and outputs produced. For this purpose, they are also grouped into nine main knowledge areas as shown in figure G.6 below.

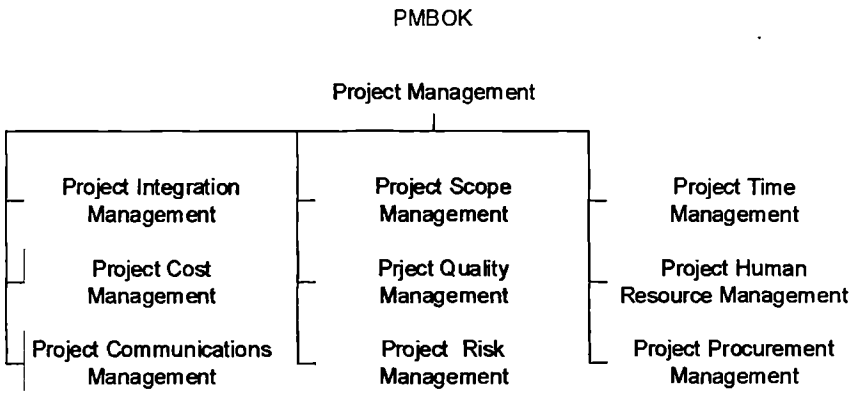


Figure G.6 – The nine knowledge areas of the PMI project management body of knowledge (PMBOK)

The elementary processes proposed by the PMI framework (PMI 1996) are not described here in detail. For each process, the PMBOK describes the inputs required to start the process, the tools and techniques applied and the outputs produced. Inputs and outputs are the basis for the interaction among the processes within each activity, across activities and across project phases. The tools and techniques considered include those which are specialised to project management and those which are borrowed from other general management areas and specific application areas. For example, the use of Pareto diagrams is considered in the processes related to quality control. This technique was borrowed form the quality management area and was not invented on purpose for project management. On the other hand, PERT/CPM networks were invented for project management purposes and hence constitute a specialised technique. They are employed in the processes related to work scheduling.

Procedures, tools and techniques

Overview

The project management process constitutes a framework wherein various sub-processes take place and where various procedures, tools and techniques are employed. The most common ones are now described.

For the sake of simplicity, no formal differentiation between procedures, techniques and tools is considered. Most of the tools result from modelling the techniques into

software applications which tend to automate their implementation, making them easier to use, quicker and more reliable. Procedures are often implied in the proper implementation of the techniques, or may consist in less structured techniques.

The PMBOK (PMI 1996) identifies a specific set of tools and techniques to be employed within each elementary process. Overall, this leads to an extensive list. The specific set of techniques here considered refer to the most common ones, which are well established in those organisations which practice project management. These are also the ones present in most of the project management literature. Some authors propose slightly different approaches to implementing certain techniques. Different techniques are also proposed by different authors. Where these differences are relevant a brief discussion will be presented.

Basic project management techniques and tools

The scope of the present research focuses on the improvement of the traditional project management framework, by considering the use of System Dynamics simulation models together with this set of well established techniques. It is therefore important to develop a basic understand about what they are, and how they are used. The brief description here presented is intended to provide an overview. A more detailed description can be found in appendix F.

Turner (1993) suggests that the basic project management techniques address the five main project objectives he identifies: cost, time, scope, quality and organisation. However, the techniques that emerged in the early days of project management were primarily aimed at addressing those issues of concern which were more tangible and readily quantifiable: resources (who), schedules (when), cost (how much) and work scope (what).

Various techniques were developed to address these issues at different phases in the project life-cycle. However, most of them are first applied at a certain stage in the development of the initial detailed project plan, during the definition phase (see life-cycle definition proposed). They are then updated continuously throughout the

project life-cycle. An important characteristic of these techniques is that they are supposed to be applied in an integrated manner. As a consequence, the information they handle must be consistent one another.

The techniques here described are presented by order of logical appearance in the process of initial project planning. This order is not intended to be rigid, nor to represent an established standard among practitioners and researchers. It is based on the author's view and on the fact that in each technique requires or benefits most from the application of its predecessors. It is important to note that the planning process is iterative and thus the techniques are reapplied or revised several times until the final plan is established.

The basic techniques are as follows:

(1) product breakdown structure (PBS) – this is a simple technique aimed at decomposing the project product into elementary sub-components. The PBS consists of a progressive breakdown process, where the product is decomposed into various levels, until the final sub-components are considered so simple and functionally self-contained, that further decomposition is not perceived useful. This specification is often confused with the project work breakdown structure (WBS). However, there is a fundamental difference between the product components and the work required to developed them: the scope of the project work is wider than the product specification. For example, some project work is not aimed at developing directly the product (e.g. administrative and management work), and the development of some product sub-components requires different type of work;

(2) milestones and deliverables chart – this is a very simple technique which is valuable in many ways throughout the project, but is often disregarded. A milestones and deliverables chart specifies the planned dates for these two elements throughout the whole project. Milestones are major events considered as landmarks of project progress, like the completion of the design phase. Deliverables are specific sub-products which are delivered to the Client for various purposes. A deliverable can be a prototype, design documentation, an intermediate product release or even the final product. Deliverables are often associated with milestones;

(3) *risk register* – the risk register is a technique similar to the milestones and deliverables chart. The risk register is the central element of the whole risk management process. It is aimed at identifying, analysing, monitoring, controlling and mitigating the relevant project risks. The risk register consists of a table listing the currently identified risks, ranked by "seriousness";

(4) *front-end estimating techniques* – front-end estimating techniques are aimed at providing estimates for the project cost, schedules and resources required, prior to the development of a detailed work plan. These techniques are appropriate for a top-down planning approach, discussed in appendix E (see figure E.3). These techniques do not required detailed information about the project, which in general is not available in the early stages of the project;

(5) *work breakdown structure (WBS)* – the work breakdown structure (WBS) is central to the whole process of project planning and control. The WBS specifies all the work that needs to be performed, so that the project objectives are achieved. Whatever needs to be incorporated into the product, or needs to be accomplished to satisfy the Client's expectations, must be translated into project work and thereby specified in the WBS. The project scope and product functionality are therefore directly related to the WBS;

(6) *organisation breakdown structure (OBS)* – the organisation breakdown structure (OBS) is also an important technique used in the project planning process. The OBS specifies the structure of the organisation which will accomplish the project work. Similarly to the PBS and WBS, the OBS is a hierarchical structure specified in the form of a tree, where individuals at the top manage the individuals below. The OBS can have a great influence on the project outcome because it addresses important organisational issues of human nature within the project. In particular, the OBS establishes relationships of authority between individuals, and forms teams of individuals which will have to work together;

(7) *responsibility matrix (WBS x OBS)* – the OBS specifies the responsibilities among the project team members – i.e. who is responsible for whom. Another important type of responsibility is the relationship between the team members and the project work – i.e. who is responsible for what, and how. This type of responsibilities is specified using a technique called the project responsibility matrix. This technique consists in developing a matrix by crossing the WBS

against the OBS. Each cell of the matrix corresponds to a couple (team member, work task), in which the type of responsibility is specified;

(8) *Gantt charts and PERT/CPM networks* – once the project work, the project team, and the responsibilities are specified, the final step into developing an operational plan is the allocation of the resources to the project tasks and the scheduling of these tasks. In order to accomplish this, there are two main techniques commonly used, which are specialised for project management purposes: Gantt charts and PERT/CPM networks (also referred to as critical path logical networks). These techniques are the basis of the whole project planning function (see figure E.1 in appendix E). There is extensive literature available on this topic, in particular regarding the use of PERT/CPM based models. Complex mathematical enhancements have been proposed over-time;

(9) *"earned value" and other control metrics / indices* – project control is based on the continuous monitoring of the project status, identification of deviations and implementation of re-planning actions. The techniques described above are used to support the development of re-planning actions. There are also techniques developed for project management purposes, which are used to support the monitoring of the project status and identification of deviations. These techniques are based on the measurement of the project status and of deviations. This implies the collection of data and further production of metrics and indices. The overall technique of project control, which specifies these metrics and indices is commonly referred to as Earned Value Management (EVM).

The techniques described above are the more commonly used in project management systems, and most of them have been developed within the project management discipline – i.e. they are specific to this field. There are many other techniques that can assist the project manager. The PMBOK (PMI 1996) provides an extensive list. Most of these tools have been developed in other fields, like Total Quality Management, but can help in handling project management issues. This highlights the inter-disciplinary nature of the project management discipline, already discussed in this section. The following sub-section briefly describes some of these techniques and tools.

Other techniques and tools

The PMBOK (PMI 1996) identifies a specific set of tools and techniques for each of the 37 elementary processes, within the project management framework. Many of these other tools and techniques are not specialised to project management and are "borrowed" from other sciences or areas of management. Other techniques are specialised to certain types of projects or industries. Finally, some other techniques are often highly sophisticated and too new so that their practical usefulness has not been accepted as a standard within the project management community. It is important to note that no single technique itself ensures the implementation of the project management process, nor does it address all the problems of project management. Additional techniques to the ones described above generally address specific problems and needs of the project management process, in a certain area of management activity.

It is not the purpose of this research to develop an exhaustive identification and review of these techniques. Many of the newer techniques address the area of risk management. This is probably because within increasingly changing environments, projects became exposed to more risks and the impacts are of greater magnitude. For the sake of reference some important techniques are briefly described:

- *decision trees* – this technique is aimed at assessing scenarios characterised by the occurrence of various risks and possible management responses. The tree results from branching the possible risk occurrences and managerial responses. Probabilities are associated to each risk occurrence as well as monetary values (say benefits). Decisions are selected based on the maximum expected monetary value (EMV) achieved. In the end, a plan of selected risk responses and their associated EMV is identified;
- *stakeholder analysis* – this technique is aimed at identifying the various stakeholders involved in the project, their interests and how the project affects them, and from here identify possible risks. Stakeholders are characterised mainly by their importance to the project success and by their influence over the project outcome. Both synergistic and conflicting relationships among them are

also identified. Appropriate ways of involving the stakeholders in the project are devised. This technique is particularly useful in large projects where there are many stakeholders affected by the project outcome (many indirectly);

- *checklists* – checklists are a very simple but useful technique which primary aim is to prevent important issues to be missed out by management. Checklists often compile lessons learned from past experiences and are updated from project to project;
- *Delphi method* – the purpose of the Delphi technique is to help a group of persons to reach a consensual opinion (e.g. a decision to be taken, a cost estimate), in a way that none feels intimidated by the others hierarchical position, dominant personality, or by other forms of power. It is based on a process wherein everyone receives compiled feedback, without knowing specifically who has produced what opinion;
- *nominal group technique* – this technique has a similar purpose to the Delphi method, but the process is different. It is based on opened discussion of opinions and secret voting;
- *influence diagrams* – as described in appendix H, influence diagrams are the qualitative side of the System Dynamics approach. Reference to the use of this technique is becoming popular in project management text books (Turner 1993), in particular regarding project risk management (Chapman 1997). Their aim is to diagnose complex situations where "everything affects everything", identify likely outcomes and devise possible solutions.

In this section, a review of the traditional project management framework was described. This description focused on the project management process and on the core set of procedures, tools and techniques employed within. The project implementation process was considered as a dual process of management and engineering work. The latter refers to the product development activities and the former refers to project control which includes the two main managerial functions of monitoring and re-planning. The set of techniques described in this section are based on the WBS, PERT/CPM logical networks and the EVM concepts.

The following sub-sections briefly discuss the "other side of the coin", of the project management "state-of-art": most projects fail to achieve their targets. Why is that so? What developments have been attempted to address the causes? Have these succeeded? Is there anything missing?

Current scenario: "management by projects" versus "project failure"

The increasing rate of change and the complexity of the new technologies and markets impose the need for quick and effective responses. As a consequence many organisations started adopting "management by projects" as a general management approach: "To achieve their corporate strategy, organisations must respond quickly to changing circumstances... (and) To respond to this pressure many organisations are adopting management by projects as a new general management." (Turner 1993). As already discussed in this chapter, projects are the natural vehicle to implement, manage and react to change. In the last decade, project success therefore became a primary factor for the survival and prosperity of most organisations. At the same time, projects have become increasingly more complex.

Unfortunately, project failure has been a major problem: "Many projects appear as failures... (and) are often completed later or over budget, do not perform in the way expected, involve severe strain in participating institutions or are cancelled prior to their completion, after the expenditure of considerable sums of money." (Morris and Hough 1987). Overall, over-runs of 40% to 200% are common, while other projects are cancelled before completion and after considerable expenditure (Morris and Hough 1987). In the software industry, a fairly recent MIT-PA survey showed that more than half of development projects fail to meet their targets: over-expenditures range from an average of 40% in commercial developments to an average of 210% in the defence industry, while schedule overruns range from 90% to 360%, respectively (Cooper and Mullen 1993). A more recent project management global survey currently underway by PA Consulting (Cooper 1999), indicates an average 35% over-runs across many various industries. The situation in the software industry is widely accepted, with the familiar "software crisis" still persisting (Pressman 1997). Many other authors appear to assume that project failure is a

fact (e.g. Davidson and Huot 1991, Turner 1993). These and other discussions about project failure agree in general that the nature of project failure is of strategic, human nature. For example, Williams (1997) argues that as modern projects became complex, strategic human issues became a crucial factor of success and the traditional approach does not address them properly.

While "managing by projects" will continue to be the preferred way for companies to implement and react to change (e.g. the pre- and post-Y2K projects, continuous business-IT alignment), the overall social welfare will depend on whether project management performance is improved in the future. Sophisticated software based tools that support project management are being launched in the market. Major technical breakthroughs are being achieved to increase product development and support increasing productivity levels. Will this solve the problem? Along with these favourable factors, within an increasingly competitive environment the scope of most projects is becoming more ambitious and complex. Projects themselves, as social and technical systems, became more complex and harder to manage. The complexity of the problems may therefore outweigh the power of technical breakthroughs. Most likely, the main causes for project failure are not being addressed properly or are being ignored by the traditional approach. It is therefore fundamental to identify these causes, verify and improve the traditional project management approach, so that they are addressed. This may imply changes to the general process logic of the approach and the introduction of new techniques and tools.

The following sub-section briefly discusses the nature of project failure and how the traditional approach fails to address the causes.

The nature of project failure and the traditional approach

Overview

Project failure can be blamed on many factors. Uncontrollable external forces are often cited but the real causes may well be internal: a defective project management system, with ineffective organisational practices and procedures

(Nicholas 1990). Good project management should be able to cope with many of the adverse external influences and thereby ensure a successful completion, despite the environment. However, while during the last decade considerable attention has been devoted to improve project management practices, such as the development of project risk management, dramatic failures still occur. It is therefore critical to address the question of what are the causes for project failure.

Projects can fail due various reasons. Problems in projects seldom result from single isolated events. Instead, they are generally caused by different interacting factors. This makes it more difficult to identify the causes. Problems can be of different nature, as they emerge in different areas of the project. Without intending to be exhaustive, the following areas and examples can be considered:

- *strategic context of the project* – the project is not properly aligned with the business objectives of the parent organisation. The project may also be competing with other higher-priority parallel projects, within the parent organisation;
- *poor management* – this may include: inadequate estimation of costs, resources required and schedules; lack of basic planning and control system, poor Client management, poor sub-contract management, poor scope specification and control, inadequate resources, among many other factors;
- *technical problems* – unsuccessful R&D and/or innovation, wrong choice of technical platforms for product development, lack of required technical know-how;
- *organisational issues of human nature* – hierarchical conflicts within the project team, "political" factors, project objectives conflicting with "personal agendas".

Project management is a fairly new management discipline, often unknown to many practising project managers. This can also be related to the causes of project failure: the above examples are more likely to occur in those organisations which do not have a proper project management "know-how".

While the causes of project failure are not always of management nature, it is the author's opinion that most of them can be prevented or attenuated if a proper project management system is in place. In many industries, it has been recognised

that the management arena offers the greatest opportunities for improvements on project performance (e.g. Boehm 1983). With a "standard" project management process, well equipped with various tools and techniques, as described in the previous sub-section, it could be expected that most problems would be eliminated. However, even organisations that implement such process experience major failures. Often, many of the core traditional techniques such as PERT/CPM are abandoned half-way through the project (Morris and Hough 1987). So, what is really causing projects to fail? Is the traditional project management approach missing something critical?

There are not many extensive studies available in the literature which explore this question in detail. However, there appears to be a strong evidence that most causes of project failure are to be found at the strategic level, and relate to human factors (Morris and Hough 1987). The main problem of the traditional approach appears to be the lack of a strategic perspective, and an undue focus on the operational issues (Turner 1993).

Strategic issues, human factors and the traditional approach

Morris and Hough (1987) undertook a survey which suggests that the main causes of project failure are to be found in areas such as the political/social environment, legal agreements and human factors. The majority of the factors relate to strategic issues of project management and are not addressed explicitly by the traditional project management techniques.

At the same time, with the increasing complexity of projects and their key role within the organisations' businesses, strategic project management has become crucial issue to project success. Despite this growing importance of strategic management, project managers have a reputation as excellent "fire-fighters", more "... interested in the here-and-now of next steps rather than strategic questions of definition, which were generally seen as someone else's responsibility..." (Morris 1994). Turner (1993) also notes project managers' common emphasis on short term planning, identifying the need for a model for the strategic management of projects. This relative lack of emphasis on strategy in project management is also

reflected in the literature: while the concept of strategy has been examined exhaustively in the context of other management areas, e.g. corporate strategy (Andrews 1980) and operations' strategy (Anderson et all 1989), there has been relatively little explicit analysis of project management.

However, translating definitions of strategic management to a project context, the characteristics of a project management strategy would be:

- the individual scheduling, budgeting and resource allocation decisions should have some pervasive logical pattern;
- strategic decisions have a widespread effect (e.g. on numerous activities);
- a project's strategy should define its position relative to its environment recognising the critical constraints;
- the strategy should ensure that the project contributes to the organisation's long term objectives.

Turner (1993) distinguishes three levels of project management reflecting these characteristics:

- *level 1* – the interaction of the project with the rest of the business; do the project's objectives contribute to the business's objectives?
- *level 2* – the individual project's strategy; this may be centred on the systems design providing the basis for determining the major targets (e.g. milestones), and the appropriate allocation of responsibilities;
- *level 3* – the tactical plan, specifying the means of achieving the project's targets, typically via the activity schedule.

In level 1 management is primarily concerned with the project's compatibility with the organisation's objectives (project selection / portfolio management). At this strategic level, managers are concerned about issues beyond the individual success of a project: as an example, within a strategy of market diversification, a project resulting in high overspend and overrun might still have contributed to the long term organisation's success. Level 2 also refers to strategic management but now focused on an individual project (or set of projects being implemented in parallel). In this research, the term "strategic project management" will be used to correspond to level 2 of Turner's classification above.

Within the traditional approach, the core techniques offered to project managers are designed for use at the tactical level (level 3). Operational issues are more readily analysed and are natural candidates for the discrete models incorporated in the traditional techniques. The operational decisions typically assume well defined objectives and constraints, which provide the boundaries for the decomposition of the project into a set of well specified tasks, resource availability and costs. Given this detailed specification, a simple discrete analysis can deliver a precise output providing a comforting timetable of activities and the associated cash-flows. However, identifying the appropriate objectives and constraints, a major element of the strategic analysis of the project (level 2), requires a different approach. A proper strategic analysis demands a more flexible tool which can model a variety of complex and not so readily quantifiable issues. However, contrasting with the proliferation of analytical techniques that assist the detailed planning and operations, there seems to be little analytical aid on these strategic higher level issues (Cooper 1980). The lack of a strategic analysis as the basis of project management has been cited as a major reason for the failure of many projects (Morris and Hough 1987).

It appears that project managers have been using informal mental models, based on their own experience and vision of reality, to support strategic decision-making in project management. Having made the key strategic decisions, the traditional techniques are deployed to support the detailed operational planning, but the crucial mistakes may already have been made. This suggests that poor, informal strategic judgement may be the root cause of many project failures. This problem has been reinforced by an apparent reluctance of organisations to learn effectively from these failures. The transfer of lessons from the past into the future can offer a crucial competitive advantage (Senge 1990), but in practice this process is seldom implemented to its full extent; often it is constrained by cultural and political factors (Abdel-Hamid and Madnick 1990).

Human factors have a critical influence on the strategic issues of project management. At the strategic level, their influence is of greater magnitude and their subjectivity is more difficult to manage. The process of estimating tasks'

duration in a project network analysis provides a good example. The estimated duration of project tasks are based on the assumption that the staff employed will work at a certain productivity level. On making this estimation, the project manager naturally considers subjective factors, like workforce motivation, schedule pressure, workforce experience, and possible errors. However, if in practice this informal analysis fails, most of the effort employed in the development of the work schedule plan may be wasted. A good, experienced project manager may well make adequate allowance for all these factors but the traditional techniques do not encourage their consideration by any explicit analysis.

Another good example of the disruptive influence of human factors relates to project monitoring: project control is based on human perceptions of the project status. In the real world errors tend to remain unperceived. As a consequence, the real progress often differs from the perceived progress. This illusion of project progress may be exacerbated by political factors, which encourage a trend to overlook errors in the early development stages of projects (Abdel-Hamid and Madnick 1990). Detailed plans based on these misleading perceptions can result in ineffective or even counterproductive efforts. Eventually the problems will have to be confronted and considerable effort is then expended in correcting errors. Despite much activity and expenditure, time passes with little change in the apparent progress with the project remaining at the nearly, or 90%, completion level; this phenomenon is usually referred to as the "90% syndrome" (Abdel-Hamid 1988; Cooper 1993); its persistent occurrence highlights poor organisational learning. Cooper (1993, 1999) has been arguing about the importance of the strategic management, with special emphasis on managing rework, something the traditional approach has been failing to address.

Two main reasons can be identified for traditional techniques not to consider explicitly most of these human factors (so called "soft factors"), at the strategic level: (1) their intrinsic subjectivity seriously restrains an appropriate quantification, particularly at the level of detail assumed by the traditional techniques; (2) their local impact on the individual elements of the project system is perceived of minor relevance and hence empirical assumptions can easily be made; their long-term impacts result from less visible compounding effects that ripple throughout the

project life-cycle. These same arguments support the idea that an appropriate analysis requires a strategic perspective. As an example, the use of schedule pressure to ensure high staff productivity might look a simple issue when the analysis is focused on a single individual task: while the small team might have been able to meet the tight schedule, the long-term effects of staff exhaustion and low work quality are not visible, yet. To cope with these effects the manager needs a more holistic view of the problem and hence a strategic perspective: how should schedule pressure be used throughout the project life-cycle, in order to provide a beneficial outcome? Reinforcing this idea, the explicit definition and possible quantification of many other human factors like staff attrition, training and communication overheads, or management willingness to change workforce, also demands such an aggregated view. While other soft factors might take place at lower levels of detail, the assessment of their impacts on project performance still demands a strategic perspective to which traditional techniques are not aimed.

A project is a man-made goal-oriented open system and as such it tends to be unpredictable and unstable. The complexity of projects and of their environment has increased the disruptive effect of subjective human factors at the strategic management level. Personal judgement based on past experience is no longer sufficient to cope with this problem. There appears to be a need to understand better the strategic issues of project management and to learn effectively from past failures; this can only be achieved through a more formal systemic analysis.

The route cause for project failure is often bad project management (Nicholas 1990). The traditional approach has placed an undue focus on the operational issues (Turner 1993). Bad project management therefore comes from the strategic arena. The need for a new model capable of addressing the systemic and "softer" issues in this area has been widely recognised (Morris and Hough 1987, Davidson and Huot 1991, Cooper 1993, Williams 1997).

Further developments

Overview

Despite its usefulness, the limitations of the traditional approach have been recognised. Further developments have been undertaken and others are underway, in an attempt to overcome these limitations. It is not the purpose of this research to investigate and present all these developments exhaustively. Obviously, they are numerous and can be found, for example, in the proceedings of the more important project management conferences, like the ones organised by the IPMA and the PMI. In this section, some of the more significant developments which are relevant for the purpose of this research are briefly discussed.

In concept, improvements to the traditional project management framework can be developed in two main areas: the underlying process logic of the approach, and the techniques and tools employed within. Improvements to the process logic generally imply changes to the way in which managerial actions are combined (e.g. sequencing, adding new actions or removing existing ones). Improvements to the techniques and tools may consist in improving existing ones or creating and introducing new ones. This may also imply changes to the process logic to consider the use of the new techniques and their integration with the existing ones.

There are some weaknesses in the traditional process logic, some of which have been recognised by both researchers and practitioners: (1) an undue focus on operational issues, (2) a reactive perspective, failing to give the required emphasis to a more pro-active control, and (3) the lack of a structured risk management process as an essential element of proactive control. In order to overcome these weaknesses, some developments have been undertaken. First, there has been an increased concern with the strategic issues (e.g. Turner 1993, Davidson and Huot 1991; Morris and Hough 1987). Structured project risk management frameworks have been proposed (Chapman 1997, Simon et al 1997, Wideman 1992), some of which will be briefly discussed. Computer based simulation tools are being used more frequently to support a more pro-active approach to the management process

(e.g. the Monte Carlo technique described above, and the System Dynamics models (Cooper 1980)). Interestingly, these developments are strongly inter-related: risks are complex events which demand a strategic approach to control, and computer simulation models provide a "test-bed" to analyse these risks and devise mitigating actions. The most recent development in the process area of project management is the Critical Chain Project Management (CCPM) approach (Goldratt 1996). The CCPM approach proposes a new way of developing project plans (still based on the logical network, like PERT/CPM), new working practices and gives special emphasis to a more pro-active risk management.

Regarding the techniques and tools, many different types of developments have been undertaken. In part, this resulted from the availability of computer power to support complex models and their analytical requirements. Many of these developments focused on improving the basic PERT/CPM network model to include a wider range of factors and project conditions. Very often, these developments resulted in complex analytical models, few of which have moved successfully into practice. Two main reasons can be identified for this: (1) the modes typically require an extensive set of input data, most of which is not available, and (2) their underlying logic tends to be complex, difficult to understand by project managers and thus difficult to validate.

For the purpose of this research, the relevant developments and trends are identified with the purpose of verifying whether they are addressing the major causes of project failure.

A good review of latest developments can be found in Williams (1997), where various emerging techniques aimed at coping with complex projects are described by their authors. This work presents developments in the following areas: modelling techniques, corporate structures, management techniques and programme management. The motivation for this work was the acknowledged insufficiency, or even inadequacy, of the traditional project management approach to cope with complex projects (Williams 1997). Regarding modelling techniques, the interesting conclusions from this work were that PERT/CPM based models are inadequate to cope with complexity because they do not incorporate management

actions, do not address uncertainty adequately, and do not capture human "soft" factors and "systemic" effects. It is important to note that both PERT and CPM were developed in the late 1920s to cope with large-scale complex projects (Nicholas 1990). If PERT/CPM has been used in practice since then, it is probably because it addresses some important issues correctly and thus helps project managers. One of its greatest merits is its simplicity and accessibility to practising managers. It therefore appears appropriate to conclude that, as it stands, the PERT/CPM technique cannot cope with complexity on its own. It needs to be improved and complemented by other techniques. Some advances to the PERT/CPM basic model are presented in Williams (1997): stochastic controlled networks (the GAAN model), and diffusion activity networks (the DiAN model). A risk management framework is also presented as well as the application of System Dynamics. There are other important extensions to the PERT/CPM approach like the PDM and GERT methods.

In addition to the PERT/CPM extensions there has been attempts to apply other types of network modelling techniques to represent projects, like Petri nets and object oriented modelling (OO). Another area where considerable effort has been underway to improve existing techniques and develop new ones, is front-end estimating. The aim is to develop accurate estimates of the effort, time and resources required, early in the project, prior to the development of a work plan. Finally, as already mentioned, a recent new approach to project planning and control has been deserving considerable attention: Critical Chain Project Management (CCPM).

Except for the System Dynamics approach, which is the subject of a more detailed study in appendix H, these developments are now briefly described separately.

PERT/CPM based models

Various extensions have been proposed to the basic PERT/CPM model aimed at addressing the various limitations in representing certain aspects a project's reality. Most of them try to address the uncertainties inherent in a project, capturing the various routes that a project may follow. These models are generally referred to as

"stochastic network models" (Golenko-Ginzburg 1997). However, the first extension to the PERT/CPM basic model was the PDM method which introduced various types of relationships among the project tasks. The PDM is described first, followed by the stochastic network methods.

Precedence Diagramming Method (PDM)

The PDM model is based on the Activity-On-the-Node (AON) representation of the project logical network. Basically, it considers three additional types of dependencies and considers "lags" in all dependencies. The three extra dependencies are: "Start-to-Start" (e.g. activity B can only start after activity A has started), "Finish-to-Finish" (e.g. activity B can only finish after activity A is finished), and "Start-to-Finish" (e.g. activity B can only finish after activity A has started). The use of "lags" means that the restrictions imposed by the dependencies can be "delayed". For example, a "Start-to-Start" dependency from A to B with a lag of 2 weeks means that B can only start 2 weeks after A has started. Negative lags may also be considered. For example, a "Finish-to-Start" dependency with a lag of –3 weeks means that B can only start 3 weeks before A is expected to be completed.

The extra dependencies and the lags in the PDM model provide more flexibility to represent certain aspects of project's reality, which in the basic PERT/CPM model would have to be simplified or ignored. The disadvantage of PDM is that the interpretation of the network is more difficult. In particular, it is difficult to anticipate the impacts of changes in the tasks' schedules (e.g. shortening the duration of a critical task does not always lead to a shortening of the whole project duration).

The basic PERT/CPM model can be considered as a simplified instance of PDM. Most project management software tools currently in the market implement the PDM model.

While PDM provides more flexibility to represent the project, this model is still focused on the technical operational issues of work scheduling. It does not consider new features to address the systemic causes of project failure discussed above.

Stochastic network models

Stochastic network models incorporate special features to address the uncertain nature of projects. Various models have been proposed over-time and there are several reviews in the literature (e.g. Golenko-Ginzburg 1997, Elmaghraby 1995). The more well-known method which appears to have gained the support of practitioners is the GERT method (Nicholas 1990), which is considered in the PMBOK (PMI 1996). Further developments led to more elaborated and mathematically complex models, like the DiAN model (Elmaghraby 1997) and the GAAN model (Golenko-Ginzburg 1997). Methods to cope with resource constrains in stochastic networks have also been developed (Bowers 1996). To illustrate the nature of stochastic network methods, the GERT model is briefly described.

Graphical Evaluation and Review Technique (GERT)

The GERT model addresses various issues regarding the non-deterministic nature of the project, including both the network logic and the tasks' duration. For example, it considers that:

- for a task to start, it may not need all of its predecessors to be completed;
- loops can occur in the network, where activities can be revisited;
- mutually exclusive alternative paths can be considered, with associated probabilities (i.e. the project can follow different courses of work depending on uncertain events associated with the outcome of certain tasks).

The GERT model considers conditional dependencies, uncertain tasks' duration and uncertain outcomes of tasks. Both tasks' duration and the network logic are therefore treated in a probabilistic manner. As a consequence, some tasks may not be performed, others may only be performed partially and others may be performed more than once (i.e. loops can occur). Because of its probabilistic nature, a GERT network is simulated in a Monte Carlo fashion. Occurrences of the project are sampled and the results are recorded in frequency histograms.

The implementation of the GERT model requires the support of an appropriate software tool. It provides an extended set of options to represent the reality of a project. For example, in many projects the work needs to be re-done several times. This can be captured in the GERT model because it considers loops.

The GERT model is still focused on the operational aspects of work scheduling. It requires that various probabilities are estimated as input parameters. The logical network itself is not as easy to interpret as in PERT/CPM because it encapsulates alternative instances of the project, as opposed to the PERT/CPM network which provides a single occurrence. The GERT network represents more a work flow process rather than a work plan with scheduled tasks. Because of the probabilistic branches, the GERT model requires that different possible outcomes are anticipated for the project. In practice, this may not be easy to specify. Perhaps because of these reasons, the GERT model is not commonly used.

The Generalised Alternative Activity Network model (GAAN)

The GAAN model was proposed by Golenko-Ginzburg (1997) as an extension to its previous CAAN model (Controlled Alternative Activity Network model, Golenko-Ginzburg 1988). The concept of "generalised activity network" (GAN) is described by Elmaghraby et al (1995) as a relaxation of the basic PERT/CPM model, to allow for non-deterministic paths in the network. The GERT model described above is an example of GAN. However, the GAAN model is much more sophisticated and complex than the simpler GAN concept.

The GAAN model builds upon the CAAN model which considers deterministic and random outcomes from nodes in the network, as well as decision-making nodes. Complex algorithms are considered to produce optimal management decisions in these nodes. These decisions select one sub-network from a possible set (called joint-variants), to represent the future project work. The aim is that when a decision needs to be made, the project will always follow the optimal path. This type of model thereby captures part of the management process. However, management decisions are considered in a discrete manner at specific moments in time. The decisions taken consist in selecting a predefined network of tasks for the project

future. Golenko-Ginzburg (1997) further argues that the CAAN model dos not address projects with "non fully-divisible" networks (i.e. cannot be subdivided into non-intersecting fragments). The GAAN model is developed with the purpose of handling this limitation of the CAAN model.

As presented by the author (Golenko-Ginzburg 1997), the GAAN model appears as a rather complex mathematical approach totally out of reach to most practising project managers. A likely cause for this complexity is the attempt to capture the managerial decision-making process at the detailed operational level. It requires that alternative paths for the project network at the decision nodes are identified by the project manager in great detail. This is clearly a major obstacle to the practical implementation of the model. Without the support of an expert highly educated in the approach, or without the use of a powerful software tool that implements the method, it is unlikely that practising project managers can use this model in real life. When faced with this question, Golenko-Ginzburg argued that this model not only identifies delays as the PERT/CPM model does, but it also helps to select better corrective actions; he further argued that this had been used in practise in small projects (a network with 40-50 tasks) (Williams 1997). In the author's opinion, this argument appears vague and self-directed. Stronger evidence is required that the model has been used successfully in practise and that it is accessible to managers.

The GAAN model is characterised by a complex mathematical approach. It stands at the detailed operational level of the PERT/CPM model. Its main contribution is to capture explicitly some aspects of decision-making in project control. As a consequence, it may help to identify optimal reactive decisions to problems at the operational level. However, the overall focus is still on the detailed project work and all the "hard" quantifiable factors.

The Diffusion Activity Network Model (DiAN)

The DiAN model was proposed by Elmaghraby (1997) and is aimed at addressing the uncertainties associated with completing tasks on time, due to dynamic changes to their remaining work contents. This approach clearly tries to address a most relevant problem in project planning and control: the scope of most tasks is

often very difficult to estimate and measure. Under pressure, the desire to finish on time leads to optimistic estimates. As the work is underway, the remaining work contents of a task may change for various reasons and it will not necessarily follow a steady decrease down to zero, within the planned schedule.

The DiAN model focuses on the dynamic and uncertain nature of the remaining work contents (rwc) in the project tasks . It considers a diffusion process to model their uncertain progress and suggests "reflective barriers" to limit the growth of rwc. This model is stochastic and is implemented via Monte Carlo simulation. For each task in the network, the model requires two inputs: an estimate of the mean of the duration, and the changes in the variance over-time. The second input is probably difficult to obtain from project managers. When faced with this question Elmaghraby argued that a set of intuitive questions may provide the required information (Williams 1997). In terms of results, when compared with the use of Poisson and Uniform distributions via Monte Carlo, the DiAN model appears to give higher probability to early completions and a smoother cumulative probability function which also extends to late completions.

The DiAN model tries to address a most relevant issue in project control: uncertainty in scope estimating and scope growth. However, it does this at the detailed operational level by modelling individually each task's uncertainty. The main contribution is an enhanced way of modelling this uncertainty by focusing on the variations of the tasks' remaining work contents. The result is a fairly complex mathematical model, which most likely is not accessible to the practising project manager. The consequent limitations are similar to the ones discussed for the GAAN model. More practical evidence of successful applications is required. Specialised computer software is probably required to pose friendly questions to managers and to "hide" the mathematical complexities.

Resource constrained network models

The basic PERT/CPM model schedules the project tasks according to their precedence relationships. The start date of a tasks depends on the completion date of its predecessors. However, in reality resources availability is often a crucial

factor that restrains the start date of tasks. This is particularly true when resources are scarce and in smaller numbers than required by the "ideal plan". Very often, all the predecessors of a task may have been completed, but the task may not be able to start because it needs to wait for the required resources to be available. Generally, other tasks with no precedence relationship in the PERT/CPM network need to be completed so that these resources become available. In this scenario, awaiting tasks will often compete for the resources being made available and thus priorities need to be considered. Different policies for prioritising the resources will lead to different project results. This scenario is referred to in the literature as "resource constrained networks" and it has been the subject of extensive study (Gemmill and Edwards 1999). Resource constraints are particular important in stochastic networks, where the tasks' duration is uncertain and thereby resource requirements may also vary considerably. Various methods have been developed to consider explicitly the impact resource constrains in PERT/CPM network planning.

A brief review of resource constrained network methods is presented in Gemmill and Edwards (1999). Weist (1964) first noted that the critical sequence of activities in a project should consider both technological dependencies and the dependencies implied by the sharing of scarce resources. Woodworth and Shanahan (1988) further implemented the concept and Bowers (1995) further proposed a simplified algorithm. These developments consider deterministic networks. Bowers (1996) further examined the problem in stochastic networks and proposed the concept of "criticality probability" as a measure of a task's overall criticality to the project, and which he compares with an alternative measure previously developed by Williams (1995), also for stochastic networks. Further studies to improve these methods using "look-ahead" techniques have been developed (Gemmill and Edwards 1999).

Overall, the methods developed to cope with resource constrains in both deterministic and stochastic networks are important and very useful. This is because they address a very real and critical factor of project performance. Nowadays, most projects need to be implemented with scarce resources. By identifying a measure of criticality for the tasks which accounts for this factor, these

methods provide the project manager with valuable information for work planning and control. However, because the heuristics are not simple and are laborious to implement manually, the practical usefulness of these methods depends on the availability of specialised software tools. Like PERT and network Monte Carlo simulation, these methods stand at the operational level. They also model the tasks' uncertainties independently and focus on the outcome of those uncertainties rather than on the causes. The systemic and soft nature of uncertainty is not addressed.

Other network based approaches

There has been many other developments of network based models, most of which aimed at coping with uncertainty. Generally, these developments take the form of complex mathematical models based on specific techniques.

For example, there have been attempts to develop models based on Petri nets. A Petri net is an abstract model used to describe and analyse information and control flow in asynchronous concurrent systems. They have been used to model projects in order to handle with time-independent issues. In these developments, the Petri net modelling approach has often been combined with other modelling approaches like PERT (Lee and Murata 1994), or its has been extended to incorporate the specific issues of projects (e.g. the WBS) (Liu and Horowitz 1989; Lee et al 1994).

Object models have also been used to represent projects as a network of inter-related elements. For example, Brandl and Worley (1993) developed an object state model of a software project, which included the following elements: management tasks, development tasks, people, systems, artifacts, versions, assemblies, products, builds and releases. This object oriented state model was then implemented as a software tool and was used to help controlling a project.

Many other advanced mathematical models have been developed over-time around the concept of the project logical network. Tavares (1999) provides a good overview of some of these developments in five main areas: modelling and

structural analysis, simulation and stochastic risk analysis, resource scheduling, project assessment and evaluation, and synthetic support to decision making.

These advanced models are all based in complex mathematical approaches and thus require the support of specialised software tools to become accessible to the practising project manager. These tools should have a user-friendly interface and must "speak the language" of practising managers. Most of these complex models are not used in practice because they lack this support. Another serious obstacle is the level of expertise required to develop a valid model, and the large amount s of input data often required. In other cases, as these complex models try to cover a wide range of issues at the detailed network level, they also tend to impose a large number of restrictions and thus their domain of application in the real world becomes narrow. While some of these models can be useful to project manager they are lacking practical testing.

Overall, most network models stand at the detailed operational level of the PERT/CPM model. They capture a wider range of factors and project conditions thus delivering a greater flexibility to represent the project reality accurately. However, the systemic causes of human and social nature, which in reality interact with the work scheduling and resource allocation issues, are still not being addressed. These models are keeping a focus on the project work. This is important but not sufficient. In order to address the systemic causes and human factors, a more strategic and holistic perspective is required.

Front-end estimating

Front-end estimating is aimed at providing high-level estimates for a project regarding the effort required, schedules and resource, prior to investing effort in developing a detailed plan. For many reasons, front-end estimates are important. They are often used at the bidding stage of a project by the contractor to develop a proposal, and by the Client to decide whether it is worth moving ahead with requests for proposal. Front-end estimates are also often used as the basis of a top-down planning process, where the high-level estimates are decomposed down the WBS.

Since the techniques used for front-end estimating do not look at the details of the project to produce an estimate (i.e. bottom-up), they are generally based on knowledge taken from past similar projects and from management experience. The role of these techniques is to structure and explore this knowledge so that accurate estimates ca be derived. The estimating process is often referred to as "knowledge-based estimation" and most of the techniques are based on empirical regression analysis. A large database of past projects is developed over-time. To estimate a new project, the most similar projects are identified in the database. Regression analysis is carried out to identify which factors or project characteristics correlate most with the project outcome. Based on the factors and characteristics of the new project, estimates are produced based on the regression curves. This technique emerged successfully in the software industry (Boehm 1981) and is nowadays becoming the dominant approach to front-end estimating (Putnam and Myers 1999). Various software tools are now available in the market and have been the basis of progressive refinement and improvement (Jones 1998). This estimating technique is also used in other industries and it has been the subject of continued research. For example, there are efforts underway to improve the process through the use of neural networks (Sequeira 1999).

Front-end estimating techniques are important to successful project management because they encourage the project to be initially planned with realistic estimates. Optimistic estimates typically lead to schedule pressure and quality problems. Pessimistic estimates lead to the prevalence of Parkinson's law, with unnecessary over-expenditures. Studies suggest that the initial estimates can have a great impact on the project outcome (Abdel-Hamid and Madnick 1990). Finding a stable estimate that provides the best outcome is a difficult task.

While poor front-end estimating can be an important cause of failure, once the project is started there are many other factors that need to be handled carefully. These are not addressed by front-end estimating techniques.

Critical Chain Project Management (CCPM)

Over the last few years a new approach to project planning and control called "Critical Chain Project Management" (CCPM) has been developed, based on the theory of constraints (TOC) (Goldratt 1997, Goldratt 1999). CCPM has been the subject of much debate and controversy among researchers and practitioners (e.g. Duncan 1999, *PMNetwork* April 1999, Pinto 1999, Cabanis-Brewin 1999, Rodrigues 1999). A detailed description of this new approach can be found in Goldratt (1999). Zultner (1999) and Patrick (1999) provide a good overview of the key principles.

One of the basic motivations of CCPM is the poor estimating of the individuals tasks' duration in a PERT/CPM network. It is argued that technical developers always pad their estimates, asking for more time than what would really be necessary. This tendency is due to a conservative attitude to protect them against uncertainty. It is also argued that top-managers are aware of this general trend and therefore have themselves the tendency to compress the schedules to remove the extra safety-time. Zultner (1999) describes these two opposing forces as creating a vicious circle: compressed estimates lead to actual delays; these delays lead to longer conservative estimates in the next project, which in turn motivate top-management to cut the extra safety further. CCPM is based on the premise that, generally, staff will ask for a conservative schedule so that they feel they will have a 90% chance of succeeding (Zultner 1999). This is, in PERT/CPM safety extra-time is considered in each individual task of the network. CCPM proposes the opposite approach: the extra safety time should be removed from the individual tasks and should be added to the end of the project, creating a project-wide protection buffer. Goldratt suggests that the duration of each individual task should be compressed down to a 50% probability of success. Based on TOC mathematics, Goldratt suggests that the whole compressed project, with the protection buffer at the end, will require a shorter duration in order to have a 90% probability of being complete on time (Zultner 1999), than the original PERT/CPM plan – an overall 15% to 25% reduction is claimed (Zultner 1999). Once all tasks will now have only a 50% chance of being complete on time, many of them will complete late. In that case, the required extra time is taken from the project buffer

and is added to the late task. Likewise, when a task is completed earlier, the extra saved in the task is added to the buffer. According to Zultner (1999), the use of the project buffer in this way provides an excellent framework for risk management: whenever time is removed or added to the buffer these events are recorded as well as their justification. The size of he buffer throughout the project provides an excellent indicator of the project overall risk of being late, regardless of what is happening in each individual task. In this way, a specific task considered as critical may last twice as planned and the overall project may still be in good shape. This prevents over-reactive management actions.

Compressing the tasks' duration creating and managing a project buffer are not the only new features proposed by CCPM. The others include:

- *the critical chain* – this is identified as the longest sequence of tasks linked not only by precedence relationships but also by resource constraints (i.e. the resource constrained critical path, a concept already analysed by Weist (1964), Woodworth and Shanahan (1988) and Bowers (1995));
- *feeding buffers* – other non-critical sequence of tasks will eventually link to the critical chain. In order to protect the critical chain of being late from delays in non-critical chains, a local buffer is created for each non-critical chain, called "feeding buffer". This buffer is managed for the respective non-critical chain in the same way as the project buffer;
- *no multi-tasking* – this principle suggests that resources working on critical tasks will not carry out any other parallel work in other tasks. This prevents distractions and ensures that the critical work is accomplished with maximum concentration and focus. This is particularly important since the schedules are aggressive (50% probability of completion on time);
- *no fixed dates* – the tasks are not planned ahead to be completed in specific fixed dates. Instead, the work is carried out as fast as possible within the compressed durations. The star and finishing of the tasks is continuously re-planed in a dynamic manner depending on actual progress of its predecessors
- *resource alerts* – because the schedules are dynamic, resources are asked to provide an advance warning of when they will complete their current task. The resources planned to work on the following task in the chain receive this "alert" and will get ready to be available to start the work.

These are the main principles underlying the CCPM approach. There are clearly differences to the traditional PERT/CPM approach. However, not all is new. First the concept underlying the critical chain is old (Weist 1964) and has been the subject of much study on resource constrained networks (e.g. Bowers 1995, Gemmill and Edwards 1999) – it is somewhat surprising that some authors are claiming the concept as new under CCPM (e.g. Uyttewaal 1999). Secondly, CCPM resembles in many aspects a flexible and dynamic implementation of PERT/CPM where the initial plan is considered as a "living object", continuously revised based on updates of actual results and forecasts. This is probably why Duncan (1999), the author of the guide to the PMI's PMBOK (1996), argues that while CCPM brings about some good ideas, these are not innovative. While some authors describe the approach with much excitement, claiming to be a promising success (e.g. Zultner 1999, Leach 1999, Patrick 1999, Rizzo 1999, Uyttewaal 1999), others advise caution. For example, Duncan (1999) argues that, according to CCPM, around 90% of the past projects planned using traditional PERT/CPM should have been completed on time (because individual tasks were planned for this degree of success), but they have not. Pinto (1999) also draws the attention for some practical constraints on the "theory of constraints" (TOC) when applied to project management:

- the difficulty in attaining fully dedicated resources and prevent multi-tasking;
- compressing the schedules by half may be eliminating essential "learning curve" time;
- the TOC assumes a highly motivated team willing to work within highly compressed schedules. In reality, motivation and team cohesion is often not a given.

In a brief response to Zultner's article (Zultner 1999), Rodrigues (1999) also argues that CCPM needs to be addressed with caution and raises two critical issues: (1) the diagnosis of many past projects indicates that excessive schedule compression is often the main cause of failure due to the various "knock-on" effects of schedule pressure and consequent low work quality (e.g. Cooper and Mullen 1993), and (2) CCPM assumes a non-changing critical chain identified at the beginning of the project. Dramatic changes in the project may lead to changes in

the actual critical chain. CCPM does not appear to provide a solution to this scenario, which will certainly require a heavy rework of the overall project plan.

The CCPM approach is based on valid concerns and proposes interesting alternatives to cope with the uncertainty associated with the project work. The focus is still on operational planning, at the same level of detail of PERT/CPM. The alternative principles proposed by CCPM are aimed at addressing important human factors, like the disruptive effects of multi-tasking, and the Parkinson's law which tends to prevent early finishes. It introduces the project buffer as an interesting element of risk management. The use of project buffer in this way assumes a more aggregated perspective of uncertainty, which is claimed to be more effectively handled at this level than at the operational level of individual tasks. These principles are valuable and will probably bear useful in the future. However, the essence of CCPM is still on operational network planning and control, where problems are identified and solutions are devised. Aggregate human factors and strategic issues of managerial nature are not explicitly addressed.

Project risk management

Project risk management has been the focus of much attention in the last few years. Williams (1993, 1998) has been carrying out a classified bibliography research in this field, identifying the more relevant developments.

Project risk management can be seen as one of the most proactive aspects of project management. Risk management looks at those events that may threaten the project, and which occurrence is uncertain and out of managerial control. A risk management process works like a "window" to the outside, from where unpredictable disturbances can be foreseen to a certain degree of confidence. Given the increasing rate of change of the business environments wherein projects take place, and the increasing complexity of projects, risks and their management became a crucial factor for project success.

Like project management, risk management evolved in two main dimensions: the process and the techniques and tools. Chapman (1997) argues that a good risk

management process is vital and should work as a framework to employ techniques and tools. The risk management process should also be closely integrated within the project management process. Chapman (1997) proposes a formal structured process called "Project Risk Analysis and Management" (PRAM), also described by Hilson and Newland (1997) and which is adopted by the APM (the UK chapter of the IPMA). This process comprises eight main stages: define, focus, identify, structure, ownership, estimate, evaluate, plan and manage. This process gives a strong emphasis to the analysis of the risks and the context within which they take place, before mitigating actions are devised. A good understanding of the risks and of their impacts on the project are essential. The PMI (1996) also proposes a risk management process fully integrated within the project management process, comprising four main stages: identification, quantification, response development and response control. This process is also described in Wideman (1992). The PMI risk management process gives special emphasis to the quantification of the risks and to the control of the responses.

Various tools and techniques have been developed over time to support the risk management process. Some of these have been described in a previous section and include: PERT analysis, Monte Carlo simulation, decision-trees, stake holder analysis and influence diagrams. According to the reviews undertaken by Williams (1993, 1995, 1998), there are two main inter-related areas of continuous development: risk networks and simulation. The term "risk networks" is used to describe sophisticated models where the logical PERT/CPM like network is enhanced to incorporate important aspects of risks. Examples of these developments include the GERT model, the GAAN model (Golenko-Ginzburg 1997) and the DiAN model (Elmaghraby 1999) previously described. The first form of risk network used in project management was the PERT model. The more commonly used technique nowadays is the basic Monte Carlo network simulation. Techniques based on risk networks incorporate risk analysis as part of planning and control, using a common logical network. As already discussed, sophisticated network models tend to proliferate in the research arena, but unless they are presented in a user-friendly and accessible fashion to project managers, they will never be tested and therefore fail become of practical use. Simulation is often used in risk networks in the form of Monte Carlo, which consists in sampling

simulation. Williams (1995) comments that simulation is becoming the main generally used tool, and later that it has become established to analyse risk networks (Williams 1998). System Dynamics simulation also started to be applied for risk management purposes (Williams 1995). System Dynamics is based on continuous process simulation and thus it is clearly different from Monte Carlo type of simulation. Later, Williams (1998) describes this technique as becoming increasingly important for the analysis of the cumulative and systemic effects of complex risks. This approach and its application to project management is discussed in great detail in appendix H. A preliminary review of some applications can be found in Rodrigues (1994).

There has been an increasing emphasis on establishing a well structured risk management process, integrated within the project management process, and commonly shared among researchers and practitioners. Because risk management is part of pro-active control, most of the techniques used are based on a PERT/CPM network, which is the core of planning and control, and on Monte Carlo simulation. The result are complex risk networks, which struggle to get acceptance from practitioners. A major limitation of risk networks is that they stand at the PERT/CPM operational level. At this level they cannot address the "softer" and higher level strategic issues, where the main causes of project failure can be found.

System Dynamics simulation is being increasingly used for risk analysis purposes. This modelling technique focuses on systemic issues and assumes a more strategic view of risks. It is not based on the network logic of the project at the operational level. System Dynamics is fairly new to the project management arena and unknown by most practitioners. A reflection of this is its reference in the PMBOK (PMI 1996) as an "activity-sequencing" tool. This shows a narrow view and limited understanding of its potential applicability. As it will be seen, System Dynamics has a much wider scope of application, and thus it can provide support to many other aspects of project management.

Conclusions: what is missing?

The project management discipline has developed a well established and comprehensive body of knowledge. It considers the project management process as a control mechanism wherein a large collection of techniques and tools are employed. Project control is achieved primarily through reactive monitoring and re-planning actions: actual results are monitored, deviations against the targets are identified and corrective actions are generated. The overall approach is based on a top-down decomposition and analysis of the project, followed by the bottom-up aggregation of results. The project is decomposed into many elementary simple tasks. Their results are aggregated to form the overall project outcome. If these tasks are managed effectively and completed on target, the whole project will also be implemented successfully. The WBS, OBS, responsibility matrix, PERT/CPM networks and earned value (EVM), are the main techniques employed to implement the project management process. The traditional approach delivers a logical view where the high-level outcome is imposed by the results achieved in the detailed tasks at the bottom level of the project. This portrays a classical analytical perspective, where the micro-events are studied in detail to derive the outcome at the macro level.

This perspective has motivated a focus on the operational issues of projects. The tools and techniques based on the WBS and logical network cope effectively with problems at this level. This traditional approach has some important merits. First, it delivers a detailed work plan which can be readily used to direct the work in the field. It also monitors progress at this level, allowing management to analyse both performance and deviations in great detail and identify the sources and persons responsible. Ultimately, it provides a robust framework to implement control at the basic project level. While this is not sufficient to ensure control of the whole project, it is an essential requirement. The success and usefulness of the network based techniques at this level has motivated extensive research to develop more complete and flexible models. Even new approaches like CCPM, which take a different perspective of planning issues, are based on the project operational network.

Experience has shown that operational control is not sufficient to cope with emerging complexities in modern projects. As already pointed out, an undue focus on the operational issues has prevented the project management discipline to tackle problems of different nature, which appear to be the cause of most failures. As previously discussed, these causes relate to systemic issues of human nature, which take place mainly at the strategic level of projects. A simplistic analogy would be to argue that it is not just the mechanical aspects of the car that matters; it is essential that driver gets the right perceptions of the problem and takes the right direction. The need for a complementary systemic analysis has been previously identified (Rodrigues and Bowers 1996b), in particular for risk management purposes (Williams 1998). Attempts to employ the traditional techniques in order to cope with these systemic problems are counter-productive and are likely to fail. These techniques take a narrow and discrete view of the project and were not designed to address these systemic issues. For example, they do not quantify human factors, they do not capture the continuous interaction between technical development and managerial decision-making, they do not consider the dynamics of rework generation, and they do not consider the impacts of managerial policies and initial project estimates. All these elements are examples of systemic issues which have a crucial impact on the project outcome.

A systemic view is required, focusing on the various dynamic interactions among the project elements, where the whole becomes much more than just the linear sum of the parts.

The important role of project management in modern life has highlighted some deficiencies of the traditional approach and the need for an alternative. Traditional techniques encourage a narrow, operational view of the project, concentrating on the detailed planning.. Several studies (e.g. Davidson and Huot 1991; Morris and Hough 1987) have identified the need for a more strategic approach. As it will be seen in the following sections of this chapter, Systems Dynamics modelling appears to offer this strategic alternative, assuming a holistic view of the organisation, with an emphasis on the behavioural aspects of projects and their relation with managerial strategies. There has been a number of academic and practical applications of System Dynamics to project management. The remainder

of this chapter addresses the need for a better understanding of the nature, differences, similarities, and purposes of traditional and System Dynamics approaches. If System Dynamics models are to play a core role in the future developments of project management, it is important to understand their distinctive contribution to the current body of knowledge and their place in a future methodology.

Appendix H – A methodological overview of System Dynamics

Overview

This appendix presents a methodological overview of System Dynamics. There is no well established and widely accepted modelling process. Views of different authors are considered, compared and discussed. For future reference a generic process is here proposed.

The overview here presented includes a brief historical background, explaining why and how the methodology emerged in the early 1960s, and a discussion of the term "System Dynamics" regarding both concept and scope, since the term is also used in other fields. This section then follows to describe the modelling process underlying the approach. This includes the discussion of how this has evolved since the early days up to present, and the different methodological and practical perspectives advocated by different authors. As it will be seen, there are currently some unsolved critical issues likely to have a great impact on the future of the methodology. Perhaps the most important one, the problem of model validation is discussed first separately, in more detail. This discussion is expanded to the more general context of validation in the field of Operational Research. The other important critical issues of System Dynamics are then discussed in some detail. This includes the problem of the endogenous perspective, continuity and aggregation among others.

Historical background

Proceeding from previous work initiated at M.I.T, in the late 50s (see Forrester 1958, reprinted in Roberts 1978), Professor Jay Forrester published in 1961 a book entitled "Industrial Dynamics" (Forrester 1961). The contents of this work would become in the following decades the subject of much controversy within the research community, as well as the inspiration for many dedicated efforts to pursue his cause. Nowadays, some fundamental problems that the methodology has to face remain unanswered. While these form a rich source for further improvement through an on-going continuous research in the field, it can be asserted some

confidence that Forrester's work has now its own place within the theory and practice of Management Science.

In his book, Forrester proposed a new computer-based modelling methodology and with it an underlying paradigm of thinking about managerial problems which, at that time, he summarised as follows: "... the investigation of the information-feedback character of industrial systems and the use of models for the design of improved organisational form and guiding policy." Initially, his work focused on analysing large industrial systems, and hence the methodology was termed as *Industrial Dynamics*. Further academic and practical developments would shift the focus to many other types of social systems, and this name soon gave way to the more general term *System Dynamics*.

As the motivation for Industrial Dynamics, Forrester identified the need for a solid scientific basis for the effective management of large industrial systems. At that time, this need was emphasised by the many observed failures in the design and management of this type of systems. While the search for such a scientific basis was not a novelty, the underlying motivation of the dominant modelling approaches at that time was essentially of mathematical nature, rather than managerial, hence focusing on optimum solutions. According to Forrester, this misleading objective was imposing unrealistic simplifications and so the resulting mathematical models were proving ineffective in practice.

The critical issue in the management of social system was the failure to translate past experiences into a common frame of reference, so that lessons learned could be transferred in time and space, and thereby be used by other managers in other situations. This failure to understand that the many observed problems were often produced by a same underlying system, encouraged a focus on the individual parts of management systems at the lower management levels, where automation of the processes is easier. This way, the resulting operational models would fail to capture the *holistic* nature of a systems' behaviour and hence to deliver effective solutions.

The core argument presented by Forrester to justify a radical change in the scientific approach to the management of industrial systems was that, as these systems were growing larger and more complex, the knowledge of their parts taken separately was not sufficient. The interconnections and interactions between the components of a system would prove more important than the separate components themselves. If Management Science were to be useful, it would have to evolve effective methods to analyse these key interactions among all the important components of a company as well as the interactions with its external environment. Furthermore, these methods would have to speak the language of the practising manager, dealing with both the pertinent information that is available and with the intangibles where these are important (Forrester 1961).

With *Industrial Dynamics*, Forrester proposed a new modelling paradigm as an attempt to deliver such useful models, which were to be used at the higher management levels. These simulation models were primarily characterised by dealing with the time-varying interactions between the individual parts of an industrial system, and by incorporating explicitly the human decision-making processes. These features would allow the models to assess the performance of management policies. The strategy of Forrester's approach was to use the power of digital computers to implement these complex models. At the same time, the idea of computer-assisted policy analysis would become the focus of further independent research, like the work developed by Bossel (1977). At that time, the study of the human decision-making processes in social systems would also become the focus of much attention (e.g. Eden and Harris 1975).

It is not the purpose of this research to present an exhaustive description of Forrester's modelling methodology, as this can be found in several books and other publications (e.g. Forrester 1961, Goodman 1974, Coyle 1977, Richardson and Pugh 1981, and more recently Wolstenholme 1990, and Coyle 1996). However, for the purpose of it is important to outline the underlying process of the methodology, and to clarify the wide nature of the concept *System Dynamics*.

System Dynamics: a discussion of concept and scope

The term "System Dynamics" is not unique to the field developed by Forrester. In a book entitled "Introduction to System Dynamics" (Shearer 1967), the author proposes "... a unified engineering treatment of mechanical, electrical, fluid, and thermal dynamic systems", and argues that "...System Dynamics interacts with...[and] is important in many fields of engineering and in scientific, economic, and business activity." On the other hand, in one of the most important publications after "Industrial Dynamics", Roberts (1978) provides "...an overview of past and continuous applications of system dynamics philosophy and methodology to managerial issues...". He defines System Dynamics as "...the application of feedback control systems principles and techniques to managerial, organisational, and socio-economic problems." The common use of the term "System Dynamics" reflects the fact that both studies focus on systems that exhibit *dynamic* behaviour. Both apply the same general principles of systems theory, systems analysis, and control theory, but the type of systems targeted is clearly different.

A system is a collection of parts which act together, in a co-ordinated way and for a certain purpose (Churchman 1968). Any system is embedded within a surrounding environment with which it interacts, and which affects its status. A *dynamic* system is one that changes its status over time. This continuous change is called system *behaviour*. Systems can be classified according to various perspectives, like their complexity, the way in which their components are interrelated, and how they interact with the environment. Boulding (1956) proposes a taxonomy for classifying systems into a hierarchy of levels of growing complexity:

(1) static structures (e.g. a map);
(2) simple dynamic systems (often referred to as "clockworks");
(3) control mechanisms, cybernetic systems, or self-regulated systems. These exhibit a goal seeking behaviour, but with no self-changing of goals;
(4) open systems, self-maintaining, or self-reproductive systems (e.g. a cell);
(5) genetic-societal systems. These a have life-cycle genetically programmed;
(6) animal systems, exhibiting both self-awareness and instinctive goal seeking behaviour;

(7) human systems, which exhibit self-consciousness (awareness of being aware), goal formulation, reflection and planning;
(8) social systems, exhibiting characteristics of human organisations: values, roles, culture, and other forms of human interaction.

The first book from Shearer (1967) is about "Engineering Control Theory" and targets systems mainly at level (3) in the above classification. These systems are some times complex, but their internal structure is well understood, since it has been fully designed by human mind. Their study is usually aimed at achieving an optimal structural design in respect to how the system reacts to exogenous shocks (called system inputs). In building a model, the engineer seeks being able to predict the system output as a reaction to the stimulus of certain inputs, as well as to understand how the system structure can be re-design in order to achieve an "optimum" performance (Shearer et al 1967).

The ideal of Forrester's breakthrough in Industrial Dynamics was to apply these same principles to the understanding and re-design of social systems. In this line of thought, a recent definition of System Dynamics (certainly not intended to be comprehensive) has been proposed by Coyle (1996): "...the application of the attitude of mind of a control engineer to the improvement of dynamic behaviour in managed systems." The aim of System Dynamics is therefore: "...to achieve in socio-economic systems the standards of controllability and dynamic behaviour which are common place in engineering systems" (Coyle 1996). Roberts (1978) also identifies this transition in the application of the information-feedback principles of engineering control theory, from simple mechanical systems to more complex electronic systems, and finally to social systems. Forrester (1961) also proposes the study of engineering systems and models as the source of inspiration for Industrial Dynamics: "The manager deals with the components of his organisation just as the engineer does with the components of his air plane..." – for a more detailed discussion on this topic the reader may refer to Richardson (1991).

Forrester's approach to System Dynamics targets systems at level (8) in the classification above. Like the engineering physical systems at level (3), social systems are self-regulated and exhibit goal-seeking behaviour. However, the

presence of the human organisational element generates novel properties like self-consciousness, goal formulation, self-change, planning, while incorporating various forms of human interaction, like cultural values. These new properties have major impacts on various aspects of the modelling approach proposed by Forrester.

Given the mathematical complexity of Forrester's models, analytical analyses would prove unfeasible. These models were only viable through the use of high-speed digital computers, where they could be easily translated into simulation models. The same type of developments also progressed in the field of control engineering, with digital computers facilitating the design, development and implementation of reliable simulation models for complex systems (Seborg et all 1989).

As the discipline of *Industrial Dynamics* gained enthusiasm, further developments emerged and the methodology was soon termed *System Dynamics* (Roberts 1978). At the same time, this term continues to be used in the engineering fields related with process control (e.g. see Ogata 1993). In this area of literature, terms like "improve understanding of the processes", "train personnel", and "design control laws and strategies" (e.g. see Seborg 1989), can be found frequently. All of these concepts and terms are also common in Forrester's related literature. Undoubtedly, the similarities will prevail: as Shearer (1967) discusses, System Dynamics interacts with many other fields, of which Management Science is just one. The common root of the different applications rests on the principles of the feedback-based Control Theory. Regarding Forrester's work however, it is the author's opinion that the difference in the type of systems being targeted gave birth to a new discipline. In this way, and without disregarding the meaning and application of the term in other fields, Forrester's discipline will be hereafter referred to as *System Dynamics*.

The System Dynamics modelling process

Overview

This sub-section is intended to provide an outline of the generic process of the System Dynamics approach. Forrester's initial description of the methodology is

first summarised. Further methodological developments proposed by other authors are then discussed. An overall generic structure for the SD process is proposed as comprising five main stages, which are presented and discussed separately. Finally, some of the different perspectives about the System Dynamics process, as assumed by different authors, are discussed.

The process proposed by Forrester

As presented by Forrester in Industrial Dynamics, the SD modelling process should evolve towards the development and use of quantitative simulation models. These models are used to support the structural re-design of management systems, in particular of their control policies, towards improved performance. Used in this way as management "laboratories", these models have the potential to sustain accelerated managerial learning.

Forrester's general method comprised several stages, starting with problem analysis, following onto formal model development, and finally to model application through repeated experimentation. Forrester proposed the following steps:

1. *the goals* – a model must address an important goal. In this step, the questions to be answered by the model are clarified. The model is set for a purpose;
2. *the description of the situation* – development of an unambiguous verbal description of the factors that bear on the questions to be answered, and their interrelationships. This is where intuition and insight have their greatest opportunity;
3. *the mathematical model* – the verbal system description is converted into a formal mathematical form, which allows for experimentation. This is a simulation model containing the mechanisms of the interactions that have been visualised between the parts of the system in the verbal description;
4. *simulation* – the model takes the place of the real system and simulates its operation under specific circumstances, with a match to real life. This simulation consists of tracing the system's time evolution;
5. *interpretation* – the results from the experimentation (i.e. the system behaviour) are interpreted. These often contradict managers' expectations. This analysis

may uncover defects in the system description implemented in the model and highlights counter-intuitive aspects of the system behaviour;

6. *system revision* – the goal of the modelling exercise is usually to improve the performance of the system, which exhibits problematic behaviour. The first model is usually developed to represent the system as it "has been". The next step in the search for improved performance is to test alternative system structures. The simulation model is revised to reflect these changes, just as like they would be implemented in the real system;
7. *repeated experimentation* – at each step in this sequence, the prior steps may need to be revisited. Each simulation results "teaches" additional questions, until the difficulties have been reduced to a point where the new resulting system design can be translated into the real system.

Like any other modelling methodology, the process comprises three main phases: (1) the problem is identified and described, (2) a model is developed with the purpose of analysing the problem, and (3) the model is used as a "tool" to help designing a satisfactory solution for the problem. The SD process is described by Forrester as being iterative, as opposed to a linear progression: at any stage it can cycle back to previous steps, thus feeding-back improved understanding and knowledge about the system and the problem. The emphasis is on interpreting the simulation results and revising the simulation model, in the search for better structures and policies. This emphasis on iteration and model revision highlights that the process is not intended to consist of a pure sequence of "model development" followed by "model use". A perfect model is never achieved first time, and changing the model is an essential issue to analyse the problem and to identify solutions.

Further developments to the System Dynamics process

Forrester proposed a general method for the SD process, but over the years the need for methodological improvements has been recognised (Coyle 1973). Further advances have been made in an attempt to achieve a more formal and clear definition of the SD process (Forrester 1968, Goodman 1974, Coyle 1977, Roberts 1978, Richardson 1981, Wolstenholme 1982, Richmond 1990, Wolstenholme

1990, Coyle 1996). Much effort has also been directed towards formalising various particular aspects of this process (e.g. Burns 1977, 1979; Forrester and Senge 1980, Randers 1980, Wolstenholme and Coyle 1983, Wolstenholme 1994, Richardson 1995, Barlas 1996, Coyle 1996, Lane and Smart 1996).

Wolstenholme (1982, 1990) proposed a clear split of the methodology into two phases: (1) "qualitative analysis", through system description, and (2) "quantified analysis", through simulation techniques. In respect to the original approach, this development was aimed at enhancing the role of the qualitative phase of system description. According to Wolstenholme, this phase should be seen as an useful method of system analysis on its own right, and as an aid to compatibility of System Dynamics with other methods of system enquiry (see for example Eden 1994). At the same time, and partially as a consequence, the qualitative phase of system description shifted from the use of Forrester's initial notation (sources, sinks, valves, and other symbols), to the use of "signed digraphs" (Wolstenholme 1982). This alternative notation has its origins in the discipline of Control Theory, and is more commonly known in the social sciences as *Influence Diagrams* (IDs) (see Morecroft 1980 for a review of diagramming tools, and also Richardson 1991 for a discussion of the origins).

Over the years other several authors have proposed other descriptions for the SD process. Table 2.1, compares some of these descriptions. There is a main sequence of five steps which is recognised by all authors (left column), so is the overall iterative nature of the process. These five steps are as follows:

(1) problem definition and system conceptualisation;
(2) development of an ID;
(3) use of the ID;
(4) development of a quantitative simulation model;
(5) use of the simulation model.

Richmond (1990) describes the modelling process using Stella/iThink. This author does not consider the use of IDs for system description, replacing this type of description with high-level maps of the simulation model and with descriptions of several reference modes of system behaviour (i.e. graphs over-time). Richardson

(1981) also proposes the identification of the reference mode of behaviour for the problem, as the starting point of the modelling process. The development of an ID consistent with this reference mode is proposed as the appropriate step forward. This ID is then the basis for the development of the simulation model, using Forrester's notation (Richardson 1981). Like Wolstenholme (1990), Coyle (1996) presents the qualitative phase of System Dynamics as an independent process of system analysis. Both of these authors propose an intermediate stage of "qualitative analysis" using IDs, wherein alternative system structures and policies are assessed based on an informal inference of the system behaviour. They stress that if a satisfactory solution is eventually found at this stage, there will be no need to proceed to the quantitative phase of simulation modelling. As it will be discussed later, this approach raises a potential conflict of opinion between those authors that consider the qualitative phase as a method on its own, advocating that IDs must be used and a solution can be found at the qualitative stage (Wolstenholme 1990, 1999; Coyle 1996, 1999), and other authors (e.g. Sterman 1994; Richardson 1981, 1996) who consider influence diagramming as an important and useful phase, but yet an non-mandatory requirement for quantitative simulation modelling. The split of the process into two major phases of "qualitative influence diagramming" and "quantitative simulation modelling" is not fully agreed within the SD community, nor is the mandatory use of IDs and the imperative need for the process to move towards quantitative simulation. Recently, there has been recent strong evidence that the qualitative phase of System Dynamics can be a method of its own (Coyle 1999)

In this research, the two phases of qualitative and quantitative analysis will be considered explicitly in the description of the SD process. It will be considered that IDs are the most appropriate precedent for simulation modelling and hence should be used in the qualitative phase (which may also find valuable support in other techniques like cognitive mapping; e.g. Eden 1994, Ackerman et al 1991). It will also be considered that the complete SD process should include the quantitative phase, through the development and use of simulation models.

Four main characteristics of the SD process are present in all descriptions of table 2.1:

(1) a logical sequence of steps, starting with problem definition and system conceptualisation, moving through system analysis and leading to the identification of a satisfactory solution;

(2) an overall process of two main phases wherein a more informal qualitative analysis is followed by a more rigorous quantitative analysis based on simulation modelling;

(3) a continuous re-iteration of the process, with the feedback of knowledge contributing to improved understanding. The identification of a satisfactory solution almost always requires more than one iteration and hence the process can be considered as iterative by nature;

(4) two major distinctive outcomes result from this process: (i) improved understanding about the feedback nature of the problem and of the system's "working laws", and (ii) the "solution" to the problem, defined by a set of structural and policy changes, and the expected improvement in the system performance.

Figure H.1 depicts the overall structure of this iterative process, highlighting the two phases of qualitative and quantitative System Dynamics, as well as the two main outcomes. Each of the phases is divided into the two sub-phases of "model development" and "model use". The two main phases of qualitative influence diagramming and of quantitative simulation modelling can be seen as continuous activities which are carried out throughout several iterations, with the ID and the simulation model being continuously revised. In an ideal scenario, the two types of analysis will interact in a complementary manner, contributing to the continuous improvement of the modeller's understanding of the problem and of the models themselves.

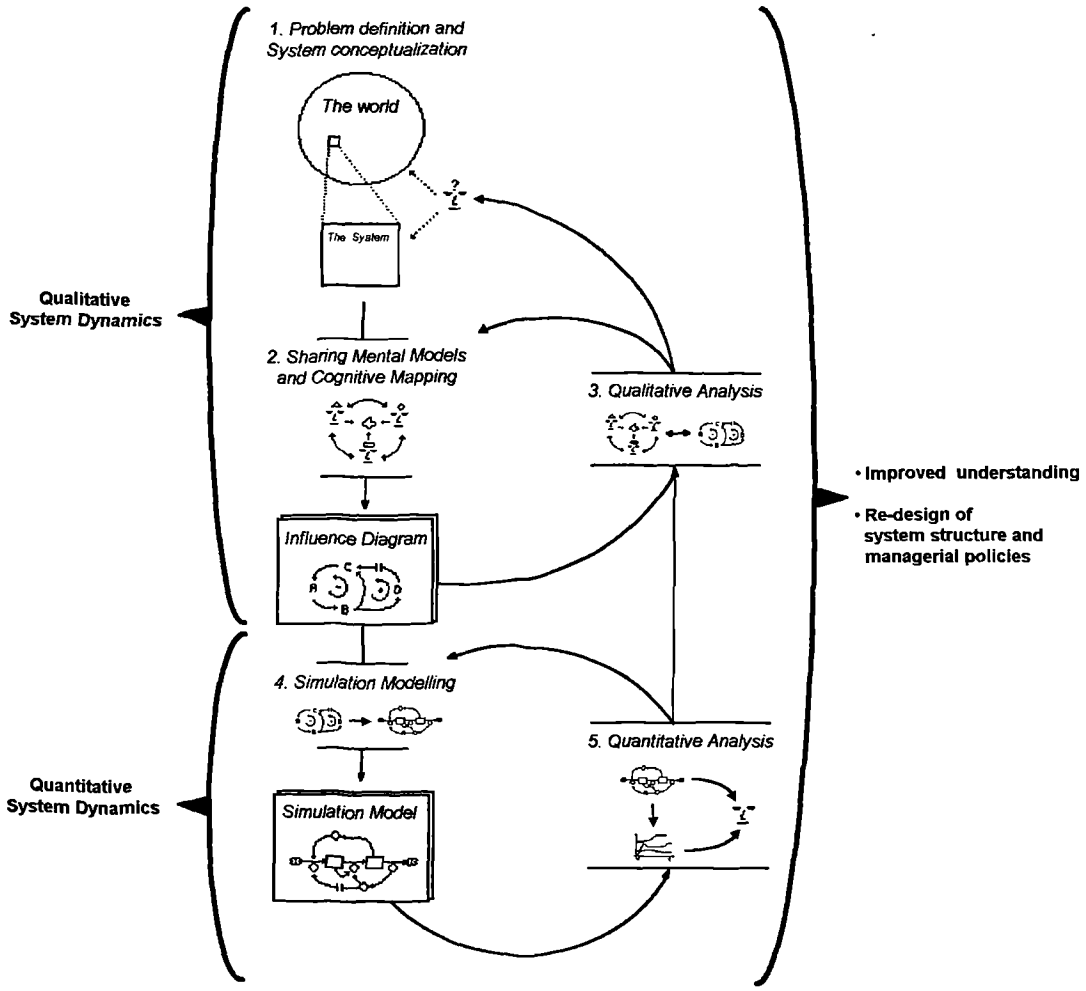


Figure H.1 – A generic view of the System Dynamics process

Having summarised the overall SD process, the following sub-sections describe separately each of the five main individual steps, and introduce the basic SD notation to be used throughout this research. The last two sub-sections provide a brief discussion about the use of different modelling elements throughout the SD process and about the different perspectives about the process advocated by different authors within the SD community.

a) problem identification and system definition

A System Dynamics study is generally motivated by the need to improve the current performance of a system. It is therefore recognised that there is an undesired scenario (i.e. the problem) which needs to be eliminated. However, not all problems are appropriate for the application of the SD method. The problem must be of *dynamic* nature and caused by *endogenous* forces within the system. The dynamic nature of a problem implies the presence of an undesired mode of system behaviour over time (e.g. a continuous loss of market share). This undesired behaviour should be primarily caused by the internal interactions within the system. A SD study is not appropriate to analyse problems which are more characterised by discrete events and which are caused by uncontrollable random forces external to the system.

Defining a problem dynamically consists of identifying which are the system characteristics of concern and their undesired patterns of behaviour. In other words, "what" is going wrong and "how"? A list of the relevant characteristics of the system is therefore first developed, and their (undesired) current behaviour is drawn in a graph over time, within a specified time horizon. This set of patterns is referred to as the "reference mode" of behaviour for the problem. It describes the actual system dynamic evolution which needs to be changed towards an improved performance.

A good example is the typical problem of schedule slippage in a software development project. This can be described dynamically by the way in which some of the relevant project characteristics evolve over time. An experienced manager will be able to identify these characteristics and describe their dynamic pattern of change. An initial verbal description would include comments about how unexpected errors emerged half way through the project, requiring extra an effort above the planned budget. The typical management reaction would be to try keeping the original schedule, transferring pressure to the staff. As progress is still slow, extra staff is ten hired in order to increase the daily man-power available. As time moves towards the original deadline, and the work is still behind the original schedule, managers are forced to re-negotiate schedule extensions, while still

putting pressure on the staff and hiring more people. From this type of description, some relevant project characteristics can be identified as follows: planned schedule, staff in the project, errors detected, among others. Figure H.2 below describes this problem dynamically, showing how these characteristics are likely to evolve within the problematic scenario of persistent schedule and budget overrun. Like any conceptual model, the graphs serve as maps to debate the problem. Further relevant characteristics are identified and described dynamically, until a consensual reference mode of behaviour for the problem is achieved.

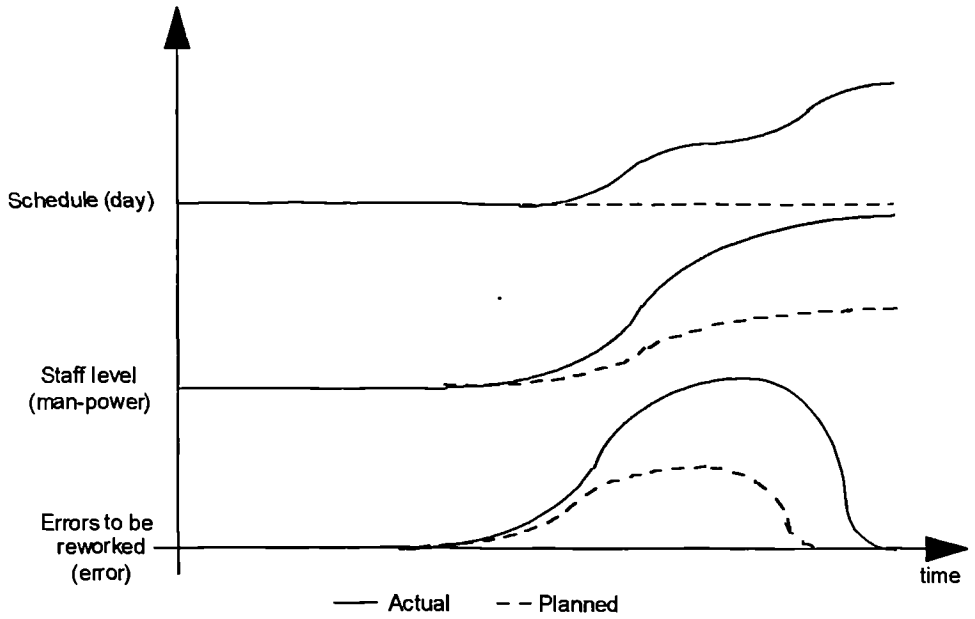


Figure H.2 – Some dynamic characteristics of a software project

Defining a problem dynamically raises questions about the relationships between the several behaviour patterns identified. For example: (i) more staff was needed because unexpected errors were detected; (ii) exerting pressure over the staff may have increased error generation; (iii) hiring more staff in the later stages decreased productivity, because of to training and communication overheads. This type of discussion leads to the identification and introduction of new factors in the study. The key premise of SD is that the problematic behaviour is generated by the underlying system structure. At this stage, the aim is to define the scope of this system, identifying which factors must be considered.

Defining the system scope involves several decisions. In the first place, boundaries must be drawn to define which sub-domain of the real world will be considered as the system under study. Secondly, not all the elements within this boundary will be considered explicitly and some will need to be aggregated to an appropriate level of detail. While these decisions will often be subjective they must be always justified, even at this preliminary stage.

At the end of this qualitative stage in the SD process, the following information should have been structured:

1. *problem identification*
    - a list of the relevant system characteristics: from which the problem can be identified, described, and analysed;
    - the reference mode of the system behaviour for the problem: how the above characteristics evolve over time in the problematic scenario, and within the time horizon established to analyse the problem;
2. *system definition*
    - system boundaries: a list of factors that must be considered as internal to the system, and their perceived relevance to the problem; a list of all the identified relevant factors which were excluded from the study, followed by the justifying assumptions;
    - a preliminary discussion of the likely level of aggregation required for the factors identified within the system boundaries

This initial attempt to clarify what the problem is, and to define the scope of system, is common to any systemic approach to problem analysis (Churchman 1968, Checkland 1991), as well as to the OR methodology (Keys 1991, Rosenhead 1989, White 1985). Formulating the problem outlines the objectives and purpose of the modelling process. One of Forrester's novelties is that all the relevant factors must be considered explicitly in this process, *regardless* of their subjective or intangible nature. The particular SD perspective is the dynamic view of the problem and its endogenous nature: the cause for the system behaviour is primarily a consequence of its internal structure. This focus on the internal view is intended to prevent the problematic behaviour to be explained by means of exogenous factors (Forrester 1961).

Like in any modelling exercise, the next stage is to convert this verbal description of the system and problem into a more formal model. This model will provide a more clear representation of the system and will be used as the basis to devise and test possible solutions. In the SD process, this model initially consists of a qualitative ID which identifies the system's factors and their cause-effect relationships. The resultant feedback structure should be able to explicate the observed dynamic problem. This ID will then be translated into a quantitative simulation model.

b) development and use of influence diagrams

The two following steps in the proposed generic SD process are the development of an ID model and its practical use for problem analysis and identification of solutions. The basic principles and notation of IDs will be first introduced. This is followed by a description of the general methods available to support the development of Ids. Finally the use of IDs is discussed.

Basic principles, notation and terminology of Influence Diagrams

The term "influence diagram" (ID) has already been referred to in this chapter. Other alternative terms are some times used to refer this type of diagrams, like "causal loop diagrams" (Goodman 1974; Richardson 1981, 1991), or "signed digraphs" (Wolstenholme 1982, Burns et al 1979, Burns 1977). When Forrester introduced Industrial Dynamics he proposed the use of "level/rate diagrams" (also called "pipe diagrams"). Although some authors argue that IDs and "level/rate diagrams" are two alternatives for the same purpose (Wolstenholme 1990, Coyle 1996), it is argued in this work that in many cases they might not be exactly the same, as it will be discussed.

The representation of feedback structures using IDs follows a specific but simple notation. There are some variations followed by different authors. The one presented is intended to be simple and as generally accepted as possible.

The primary purpose of an ID is to represent the system in the form of a dynamic feedback structure, capable of explaining the occurrence of the observed problem. This structure consists of a set of elements linked through cause-effect relationships which originate closed loops. What is a cause-effect relationships? A causal relationships means that a certain element of the system affects the other. An element can be a physical component or an abstract concept about the system. As an example of a cause-effect relationships is: the higher the "number of inexperienced staff" (cause) the lower the "overall productivity" (effect) of the project team. A causal relationship can represent two types of effects which are referred to as "positive" or "negative". Since the elements of a dynamic system change over time, a negative effect occurs when a change in the element which is the cause has the opposite effect in the element that suffers the effect. In the example above, the more the inexperienced staff the less the productivity. On other hand, a positive effect takes place when the changes in the causal relationships follow the same direction. For example, the higher the productivity the higher the progress rate. Causal relationships are represented in IDs through the use of arrows, which point from the element "cause" to the element "effect". The arrow has sign "+" or "-", which indicates the type of effect (some times the letters "s" for "same", and "o" for "opposite" are used instead, to represent positive or negative effects respectively). The relationships for the examples mentioned above would be represented as follows:

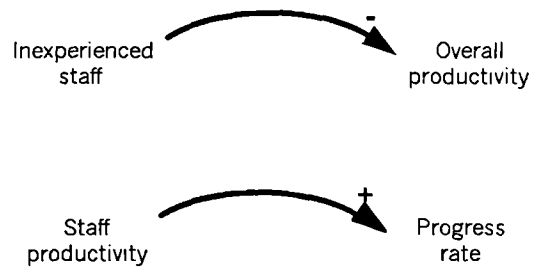


Figure H.3 – Representation of positive and negative cause-effect relationships in IDs

Dashed lines are some times used to represent abstract relationships, which represent information flows, whereas full lines are used for physical relationships. For the sake of simplicity, this distinction will not be made. If dashed lines are used their meaning will be clarified.

It is important to note that the concepts of "positive" and "negative" causal relationships should not be confused with the idea of "desired" or "undesired" effects. For example, the "staff experience level" has a negative effect on "error generation" in a software project. This is a desired effect. The concepts simply refer to type of change, either in the same direction (positive) or in the opposite direction (negative).

Several causal relationships tend to originate a closed chain, which is called a *feedback loop*. Throughout a feedback loop a change in one element will propagate throughout the chain, eventually affecting itself. For example, "inexperienced staff" in a project reduces "productivity". Low productivity motivates managers to "hire more staff". But new staff is usually inexperienced, thereby increasing the number of "inexperienced staff" in the project (the original element . Like causal relationships, feedback loops can be "positive" or "negative". A positive loop occurs when an initial change reinforces itself by propagation throughout the chain, just as in the above example. A negative loops occurs in the opposite situation, where the initial change propagates effects throughout the chain which eventually counter itself. For example, as staff productivity increases and the work is accomplished at a higher progress rate, the staff starts feeling less pressured to work hard. This "relaxation" brings down the initial increase in productivity. Positive loops are also referred to as "reinforcing loops", "degenerative loops", "vicious circles" or "virtuous circles". Negative loops are may also be called "balancing loops" or "control loops". According to the notation here proposed, feedback loops are represented in a ID by an internal arrow which is intended to identify the set of causal relationships that create the loop. This arrow has an indication of the type of loop in the middle. The sign "R+" will be used for positive loops, and "B-" for negative loops. The above examples can be represented as follows:

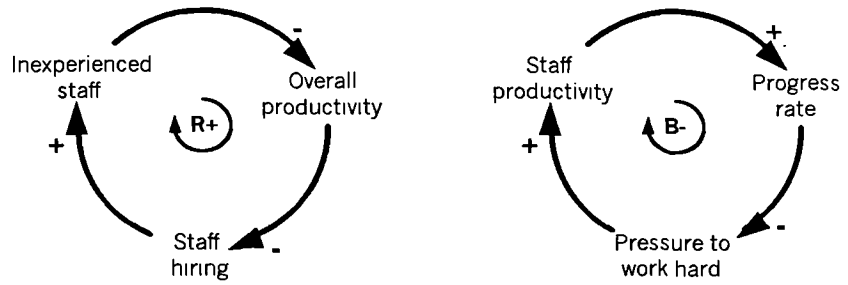


Figure H.4 – Representation of positive and negative feedback loops in IDs

Again the concepts of "positive" and "negative" feedback loops should not be confused with the idea of "desired" or "undesired" loops. Positive loops often represent undesired "snow-ball" effects. On the other hand, negative loops are the essence of managerial control in managed systems. In fact, they often represent the managerial control mechanism, which the SD study is aimed at improve. The generic management control loop can be represented as follows:

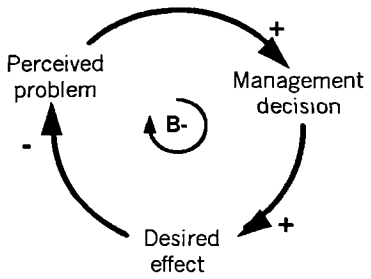


Figure H.5 – Generic negative loop of managerial control

As a problem is perceived, corrective actions are generated through management decision-making. These actions are aimed at producing a certain effect on the system, which will eventually reduce or eliminate the initial problem.

Feedback loops will often incorporate a large amount of causal relationships. In this case the type of loop is not obvious. One way of identifying the type of loop is to multiply the signs of all the relationships in the loop, with the resulting sign indicating the type of loop.

Causal relationships identify effects that the elements of a system exert one another. These effects are seldom instantaneous: in reality, some time elapses until they take place. As Forrester (1961) notes: "Time delays arise in every stage

of system activity – in decisions, in transportation, in averaging data, and in inventories and stocks of all kinds." The concept of *delay* is critical in System Dynamics: problems observed in social systems often occur because of the delays present within the control mechanisms. Two different types of delays are considered in System Dynamics: physical delays and information delays. A physical delay has to do with the flow of a physical entity within the system. For example, the time it takes for newly hired staff to go through training and become available in the project. An information delay has to do with how information about the system state is transformed to generate human perceptions, decisions and human behaviour. For example, changes in the staff productivity take some time to be fully perceived by management. These delays can be more or less continuous (or "smoothed"), or even discrete, depending on the type of real world effect being modelled.

Delays are often represented in IDs through the symbol "D", with the name of the delay in subscript. Delays are present in many cause-effect interactions between elements of a system. The explicit representation of delays in IDs is sometimes omitted, as it is assumed to be implicit in the relationship. In this work an explicit representation will be used whenever there is a particular interest in highlighting the presence of the delay (e.g. the delay is of significant magnitude).

An example of information delays in a software project are when staff perceive that progress is behind schedule and start feeling pressured to work harder. The cause-effect relationship between the elements "perceived work progress" and "schedule pressure" is not instantaneous: progress takes time to be assessed by the staff, and "schedule pressure" also takes some time to build-up. Furthermore, the impact of this pressure on the work rate is also progressive, as staff take their time to find ways of increasing their productivity. This can be represented in a ID as follows:

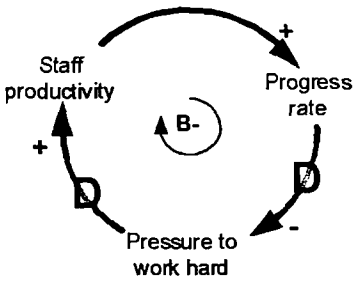


Figure H.6 – Information delays represented in an ID

Delays are extremely important in System Dynamics, as they are in the real world. As problems occur, it takes some time for managers to perceive their occurrence through changes in the system state. It takes further time for them to analyse this information and take a reactive decision. The effects from these decisions (often not the expected ones), also take some time to produce an impact on the system state. Coyle (1996) proposes the following ID to highlight the importance of delays in managerial feedback:

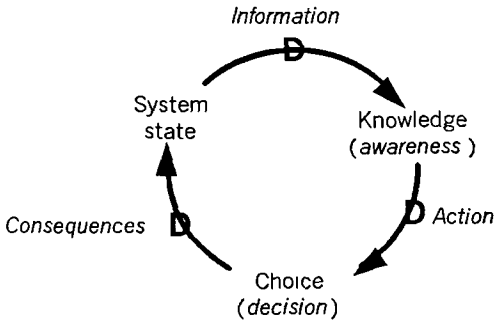


Figure H.7 – The importance of delays in managerial feedback

Finally, it is important to note that the use of IDs in the form of "word and arrow", as described above, is not shared among all authors. Some suggest the immediate use of "level/rate" type of diagrams, as a direct means to describe the feedback structure. However, this type of diagrams is more formal then the "word and arrow" IDs as they impose the need to classify the variables into certain categories (e.g. level, rate). The "level/rate" notation was the one initially proposed by Forrester (1961) and is closer to the quantification level of a simulation model. In fact, in most SD simulation tools this notation is used to represent them model and is directly translated into the equations. In the author's opinion this level of formality restrains the modeller's ability to develop an initial high-level qualitative image of the system feedback structure, which is more naturally perceived by "words and

arrows" forming feedback loops (with no need to decide what is a level and what is a rate). As it will be shown, "level/rate" diagrams are better at identifying the physical process flows within the system, but are weaker at representing feedback loops. Nevertheless, this is an issue opened to discussion and is not the purpose of this research to propose a final judgement. In this research, the use of IDs as described above will be considered as *preceding* the use of "level/rate" diagrams for simulation modelling (which will generally be more ids-aggregated)

Developing an Influence Diagram

Developing an ID is a process that requires some caution, as these diagrams may quickly grow too complex, becoming confusing and difficult to read. The scope of an ID and the level of detail to be considered must both be well balanced, so that the diagram is simple enough to provide useful insights. If a certain aspect of the system requires a high level of detail, then the scope of the ID should be restricted to the specific sub-domain of interest within the system. On the other hand, less detail usually represents the need to achieve a wider view of the problem, and hence a wider scope should be considered. In order to overcome the conflict between scope and detail, and the resultant complexity, several IDs can be developed for the same study, as suggested by Coyle (1996). Each individual ID should incorporate a limited number of feedback loops. This can be achieved either by adopting a high level perspective, by looking at a particular part of the system, or by looking at a particular set of feedback effects at a time.

Given the dynamic description of the problem develop in the previous step in the SD process, the modeller now wants to develop the appropriate ID that captures this description. There is unfortunately no formal method that can ensure the development of the appropriate ID (at least commonly adopted within the SD community). However, structured approaches can be used in order to bring some discipline to this process and thereby promote validity. Coyle (1996) reviews some of the methods available in the literature, and Wolstenholme (1990, 1994) also reviews two of these methods. Overall, these are as follows:

1. the list extension method (Coyle 1977, 1996): this is based around the idea of starting with a small list of factors, identify the direct interrelationships, and gradually extending the list "to the right", until the feedback loops emerge;
2. the entity/state/transition method (Coyle 1996): this gives emphasis to identifying first the entities in the system and their life-cycles. The life-cycle of an entity is defined by a sequence of states through which it flows continuously. The method then follows to developing the information processes that dictate the transition rates between these states;
3. the feedback loop approach (Wolstenholme 1990, 1994): this method evolves by identifying the key feedback loops individually, and then linking them together. These feedback loops are then refined into more detail, with intermediate variables being introduced in the relationships, until they can be eventually classified as levels or rates;
4. the common modules method or modular approach (Coyle 1996; Wolstenholme 1990, 1994; Wolstenholme and Coyle 1983): this is based on the use of generic standard modules that can be linked together to represent the system feedback structure. The system should be decomposed until the fitness to the modules is recognised. This method is appropriate to be used in conjunction with (2).

None of these methods is the best or more appropriate. In practice the modeller often uses all of them, although in an informal manner.

Use of Influence Diagrams throughout the SD process

Influence diagrams are useful during and after the development process. When developed with a group of managers, the IDs can provide a useful forum for the debate of ideas and sharing of mental models (a crucial discipline for learning organisations (Senge 1990)). In this modelling process managers are "forced" to represent explicitly their personal "beliefs" and views about the system structure. In this way, the consequent need to share an agreed vision enables mental models to be improved. Figure H.8 provides a very simple example of how developing IDs enables this learning process: in the face of schedule slippage in a project, one manager believes that the problem can be solved by hiring more staff. Another

manager has a different perspective: bringing more staff can make things even worse because new staff needs training; this will create overheads and the work rate is more likely to decrease. By sharing mental models it is possible to identify explicitly these different mental models and achieve a common improved understanding (and model). A shared mental model will consider both effects (i.e. feedback loops), as shown in the final ID represented in figure H.8.

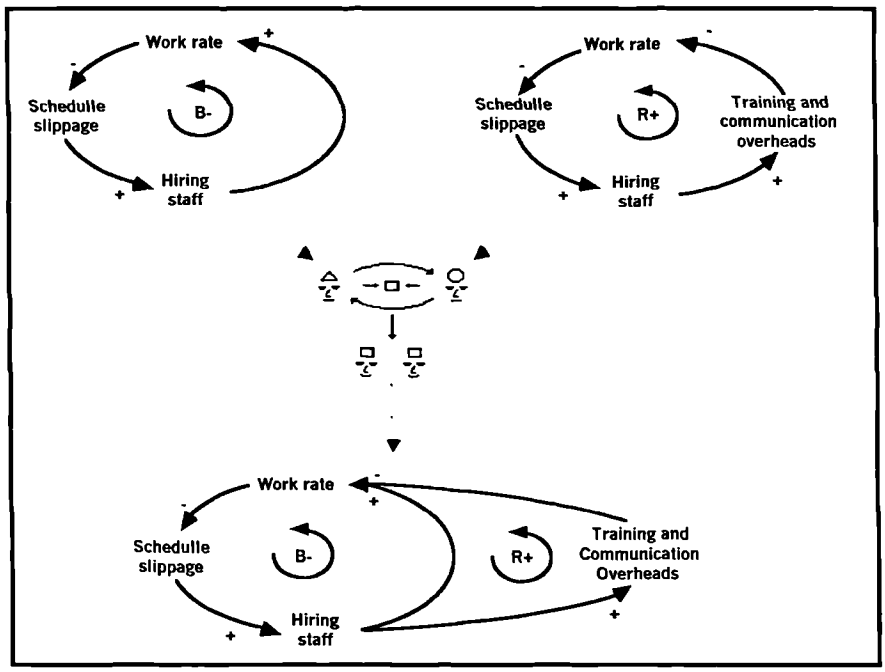


Figure H.8 – Sharing mental models while developing influence diagrams

After a final shared feedback structure is developed, the influence diagram is used to analyse the feedback-based causes of the observed problem. This qualitative analysis consists of two main steps:

(1) *relate the feedback structure with the observed behaviour* – this includes identifying the main feedback loops, recognising their impact over the observed behaviour and their likely dominance over time;

(2) *infer how possible changes in the system structure will affect behaviour* – the basis of this analysis is that the system behaviour results from the several feedback loops dominating the course of the events at different periods in time. The search for solutions is focused on strengthening existing desired loops or creating new ones, and on weakening or eliminating the undesired loops.

The degree of confidence with which the system feedback structure, as described in an ID, can be used to infer about the system behaviour is the subject of much disagreement (see the contrast of opinions between Richardson 1996, Wolstenholme 1990, and Coyle 1999). Changing the strength of the feedback loops (and thereby their dominance over the system behaviour), and eliminating and creating new ones, generates new feedback structures and therefore unknown scenarios for system behaviour. The ideal of the qualitative phase is to infer about the *general* trends of the system behaviour within these scenarios. In simple diagrams that represent scenarios with which the modeller is familiarised, this inference may appear to be reliable. However, experience shows that when these same scenarios are tested using quantitative simulation models, counter-intuitive effects often emerge. The "mathematical" complexity of the feedback structure of a social system is often overwhelming, far beyond analytical reach, and hence it may hold some surprises. Familiarity with the problem, modelling experience, and model complexity are important factors that restrain the reliability of qualitative inference.

Nevertheless, qualitative analysis does bring light to why problems occur: undesired positive loops (so called "vicious circles") are often the responsible for problematic behaviour. By analysing how the effects from undesired loops can be eliminated, and how the dominance of "beneficial loops" can be strengthened, an ID provides a useful tool to devise successful solutions.

At the end of this stage of qualitative analysis in the SD process, the feedback structure of the system is defined and the key feedback loops identified. The problem is diagnosed and possible solutions are identified. The SD process now moves into the quantitative phase of simulation modelling. Quantitative simulation models hold the promise of providing a much more reliable and rigorous inference of how changes in the feedback structure of a complex system will affect its behaviour.

c) development and use of simulation models

The following two steps in the generic SD process proposed refer to the quantitative phase of simulation modelling. The notation and basic the principles of continuous simulation in System Dynamics are fist introduced. This is then followed by a brief discussion about the practical use of SD simulation models.

Basic principles and notation of simulation in System Dynamics

System Dynamics models differ in some aspects from other more traditional types of simulation models (for a review of computer simulation see Pidd 1984). There are currently a few specialist software packages that support the development of System Dynamics models, like Stella/iThink, Vensim, Powersim, Dynamo, or Cosmic (Coyle 1996). One of the distinctive characteristics of System Dynamics models is that they quantify subjective elements of human nature in a system (so called *soft* factors), regardless of their intangible nature (e.g. "staff motivation" or "experience level"). As will be discussed later, this feature raises some important issues in terms of validation.

The relationship between the feedback structure captured within a SD simulation model and the dynamic behaviour it produces can be visualised as feedback loops "spinning" continuously over time (e.g. A affects B, then B affects C, then C affects A, and so on). Figure H.9 illustrates this concept by showing how a simple reinforcing loop generates behaviour patterns: as more staff is hired into the project, the number of inexperienced staff increases and the overall productivity decreases, leading in turn to even more hiring.

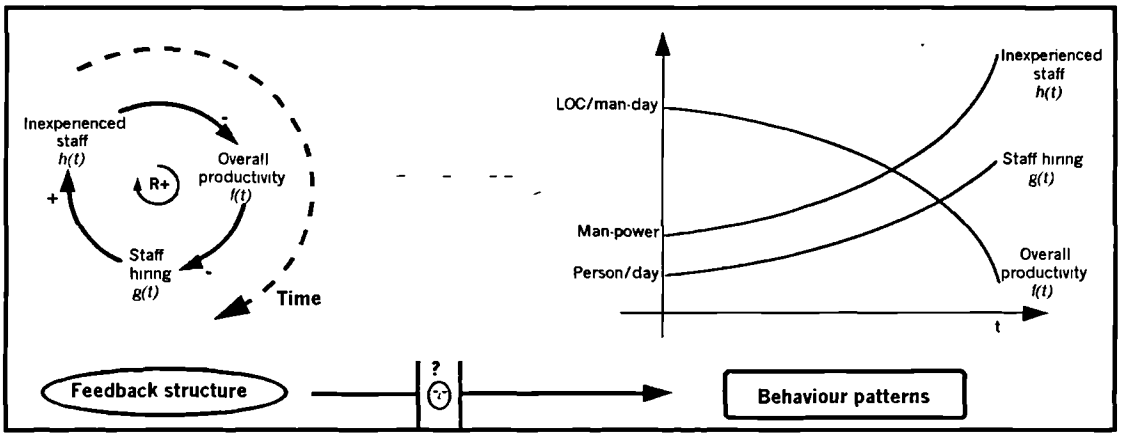


Figure H.9 – The feedback structure within the model generates the behaviour patterns

The ideal of a SD simulation model is to implement this process quantitatively, thereby establishing a formal relationship between the feedback structure and the dynamic behaviour of the system. Without simulation this relationship can only be established intuitively, as also suggested in figure H.9. The core principle of a SD simulation model is that, given the system feedback structure, as represented in the ID, it is possible to derive with mathematical rigour the resultant system behaviour.

The first requirement of a SD simulation model is therefore to capture the feedback structure represented in the IDs. The system elements in the ID need to be translated into variables in the simulation model and their causal interrelationships into mathematical equations (which define the variables). Conceptually, a simulation model can be considered as a complex system of equations. However, simulation models are certainly more than this. Most of the specialist software packages associate a "level/rate" type of diagram to the model (as initially proposed by Forrester (1961)), which is used as the basis to define the equations. In this research it will be considered a SD simulation model includes both this "level/rate" diagram which identifies the system elements and relationships, and the mathematical equations which quantify the relationships.

The "level/rate" diagrams used in simulation models are developed using a small set of elementary "building blocks": levels, flow rates, information links, and auxiliaries. Except for information links, each building block is itself a variable. The

underlying principle is that systems can be seen as being composed by entities which flow continuously throughout several states. As an example, the diagram in figure H.10 represents the entity "staff", flowing throughout a project. This entity enters the project system from the outside world (represented by a cloud), it then flows within the project through the states "staff in training" and "staff in the project", and finally leaves the project again to the outside world (another cloud).

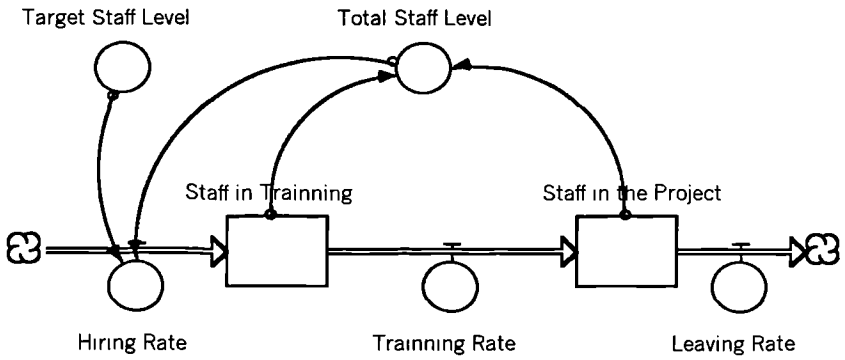


Figure H.10 – Example of a "level/rate" diagram considered in a SD simulation model

The squared boxes represent "level" type of variables, which can be seen as the accumulation of a certain type of entity in a certain state of its life-cycle. The flows with a round valve attached represent the "flow rate" type of variables, and can be seen as the mechanism responsible for transferring entities from one state to another, or between a state and the outside word. The round circles with no flow attached represent "auxiliary" type of variables, which contain information about the system. Finally, the arrows linking variables represent "information flows", which identify how variables affect one another. This process can also be seen as water (instead of staff) flowing throughout several tanks, wherein it accumulates for some time. This water-flow is controlled by the valves in the pipes (i.e. the flow rates). The information links are used to implement the feedback control mechanism: information about the system state, which is represented by the levels and auxiliaries, is used to generate control decisions (also in the form of information). These decisions are an input to the flow rates which will change to affect the system state.

The concept of "system state" is important in SD. At any moment in time, this is determined by the values of all the "level" type of variables in the model. The flow

rates, on the other hand, dictate how this state changes over a period of time. These variables will often represent management decisions, and depend on the system state. The closed loop between the system state and the rate of change implements the feedback mechanism of managerial control present in managed social systems. The auxiliary variables in the model can be used for different purposes, but they are primarily designed to represent information about the system state which is derived from the levels. This information is implicit in the levels and the auxiliaries make it explicit, thereby clarifying the logic of the feedback control mechanisms. Auxiliaries can also be used to represent external factors which affect the system rate of change but do not depend on the system state. In this case, they do not participate in the feedback loops and are therefore referred to as "exogenous" variables. Finally, auxiliaries can also be used to represent a particular property of the system, which affects the way it changes over-time. In this case they will be hereafter referred to as "intrinsic" factors. Like the exogenous variables, these factors are either constant or a function of time. Figure H.11 provides a generic representation of how a "level/rate" diagram represents a feedback control mechanism based on these concepts.

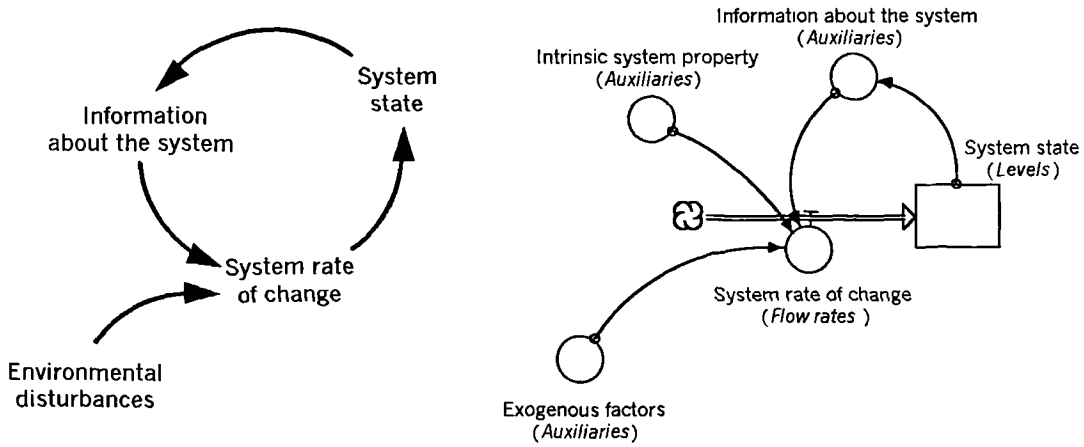


Figure H.11 – "Level/rate" diagram representing a generic feedback control mechanism

A "level/rate" diagram specifies which are the variables in the model and their type. It also identifies which variables affect which. The next step in the modelling process is to quantify these causal relationships through the use of equations. There are some restrictions to which variables can affect which, and how. These

restrictions have to do with some underlying principles of the "level/rate" representation, which are summarised as follows:

(1) levels can only be affected directly by rates. As they represent accumulations they depend only on their input and output flow rates;

(2) rates can only affect directly levels. As they represent the flow of an entity moving from one state to another, or between states and the outside world, their role is to accumulate and deplete entities from levels;

(3) rates can only depend directly on levels and auxiliaries. As they represent how the system changes over a period of time, they will depend only on the system state and on external factors;

(4) auxiliaries can only depend directly on levels and on other auxiliaries (as far as this does not originate a closed loop of auxiliaries). This is because they represent information about the system state, in a certain moment in time, implicit and derived from the levels. A rate does not represent a static characteristic of the system state;

(5) time is the only independent variable in the model (i.e. it changes but it does not depend on any other variable), and can influence directly auxiliaries and rates. Levels are also influenced by this variable, but only indirectly through the flow-rates.

It should be noted that some SD software packages violate some of the above restrictions. For example, Stella/iThink package allows a rate to affect directly an auxiliary or another rates. However, in these cases a delay should be implicit, like in the smoothing of information. An intermediate level is therefore also implicit. In general, although in practice these principles can be violated, it is here suggested that it is a good modelling practice to preserve their meaning in mind thereby avoiding cumbersome structures.

The ideal of SD simulation is to implement continuous processes. In practice, this is impossible to implement in digital computers. The simulation is implemented instead on a discrete "time-step" basis according to which time progresses in discrete steps. This time-step is constant and is defined by a constant called DT. The smaller the DT the more the simulation approaches continuity. In this way, it is

considered that for every interval of time DT, the rates are constant. The equations for the levels are automatically defined as follows:

$$Level(t) = Level(t - DT) + Rate([t - DT, t]) \times DT$$

This "time-step" approach to continuous simulation has further technical impacts. In particular, it restricts the definition of information and material delays in the model. To counter this problem, the value of DT should be set as small as possible but this also has a serious undesired impact: the amount of time required to run the simulations. Furthermore, changing the value of DT may affect the results produced by them model. It is not the purpose of this research to discuss this issue in detail, and the interested reader may refer to the explanations in Forrester (1961), Richardson (1981), Wolstenholme (1990) or in Coyle (1996). In order to prevent problems with the value of DT, the most commonly regarded "rule of thumb" has been proposed by Forrester (1961) as follows: "...the value of DT should always be less than half of the time delay of the highest order delay in the model, divided by its order. If there are several delays with the same highest order, then choose the on with the shortest time delay".

The simulation in the SD model runs continuously over time (i.e. in a time-step basis). All the variables in the model are continuously updated as they affect one another, in closed loops. This process of deriving the value of the variables from their inputs follows a logic sequence: the levels are first affected by the rates and new information is generated about the system state. The new system state will affect the rates for the following DT and the process repeats. In a certain moment in time *t*, the following computational steps are implemented:

(1) update the levels from the rates according to the following equation:
$$Level(t) = Level(t - DT) + Rate([t - DT, t]) \times DT$$
(2) update the auxiliaries from the levels, and eventually from the independent variable *time*. Auxiliaries that depend on other auxiliaries are updated only after all their predecessors have already been updated;
(3) update the rates for the next time interval [*t, t+DT*], from the auxiliaries and eventually directly from the levels;

(4) advance the *time* variable to *t+DT* and repeat the process, until the time horizon established for the simulation has been reached.

Practical use of SD simulation models

The modelling process forces the analyst to assume an even more rigorous view of the system than in the ID. This is because in the simulation model the variables must be first classified, and further quantified into a measurement scale. Relationships must be translated into mathematical equations, and dimensional consistency must be ensured. Several simulations are usually run before the final model is complete. As the results produced by the model are not as expected, assumptions must be revised. This conceptual revision includes not only the quantification of the causal relationships, but also the feedback structure captured in the model as translated from the ID.

The simulation model provides a more rigorous description of the system and of the description of its behaviour. When the model is run, behaviour patterns are plotted in graphs over-time. Unlike in the qualitative analysis using the ID, this behaviour is not influenced by the modeller's personal expectations and often proves counter-intuitive.

Figure H.12 provides an example of the typical quantitative output produced by a system dynamics project model. In this example, the simulation runs from 0 to 276 days. Four patterns of project b behaviour are shown: productivity, number of staff working, estimated completion (days), and work rate. Each of these variables has its own scale on the vertical axis. The project was originally scheduled to take 100 days but the initial estimate of completion (curve 3) is 298 days and consequently more staff (curve 2) is hired, in an attempt to reduce the schedule slippage. However, the individual productivity (curve 1) falls as efforts are diverted into training and communications overheads. The overall work rate (curve 4) increases, but only to a limit beyond which the various disruptive factors dominate. At this point in time, the estimated completion is 220 days and management reacts by recruiting even more staff. However, the training overheads and other disruptive factors cause both the individual and total productivity to decline. Clearly, keep

hiring more staff is not the to the answer to this project's problems.

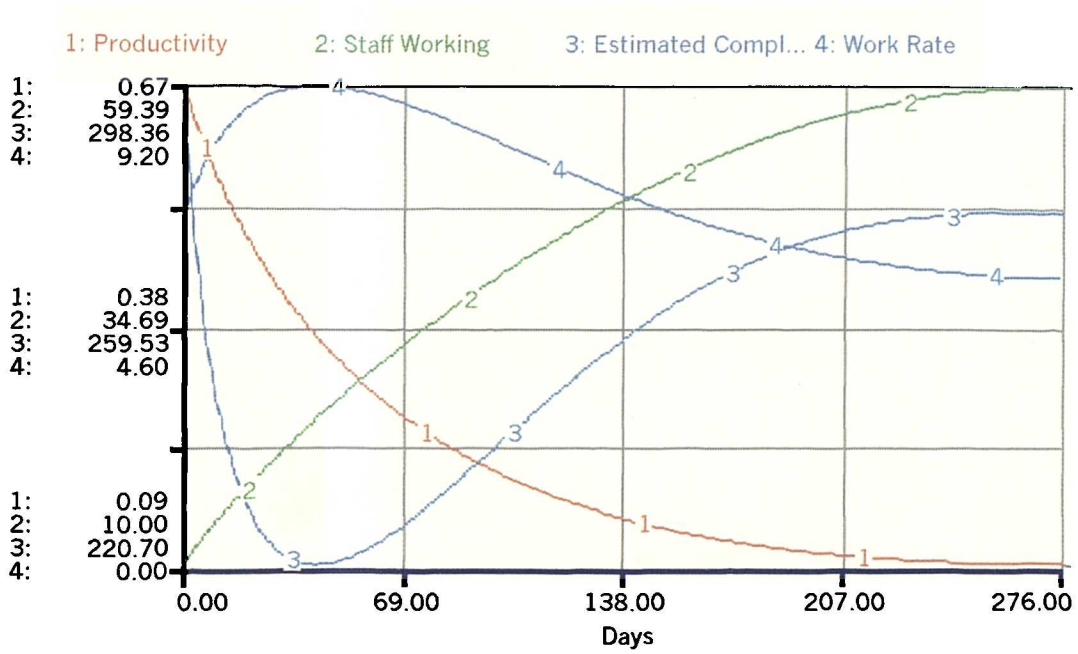


Figure H.12 – Example of graphical representation of a project's behaviour produced by a simple System Dynamics simulation model

Managers can use SD simulation models to test how changes in the system structure affect the outcome. In this way, the model plays the role of a management laboratory, where alternative scenarios can be assessed quickly and through reliable "what-if" analysis.

d) the modelling stages and the different types of models

As just described, the SD process evolves throughout several stages of system description. Throughout these stages the level of formality increases, reflecting gains in knowledge and a more consistent and deeper understanding about the problem and the system. This evolution towards increased formality is reflected in the use of different concepts and modelling elements of system description, as shown in table H.1.

Modelling stage	Elements of system description
Problem identification	Verbal description. Behaviour patterns
Causal mapping (e.g. using GCope)	Main concepts and relationships
Influence Diagram	System characteristics, signed arrows, delays, feedback loops
Level/rate diagram (simulation model)	Entities, variables: stock levels, rates, auxiliaries; relationships: entity process-flows, information links
Quantified simulation model	the above, plus mathematical equations

Table H.1 – The modelling stages and the formality of system description

Increasing the level of formality brings more discipline and rigour to the knowledge incorporated in the model. However this can be at the expense of some clarity and model flexibility. The SD process should be regarded as iterative, where *all* models are important and hence should be used and updated throughout the whole process.

e) different perspectives of the SD process

As discussed before, there is no commonly agreed definition of how a System Dynamics study should be implemented. Over the years, there has been a number of applications in different fields wherein the SD process was implemented in different ways. In some cases, these applications were carried out under very different methodological perspectives. It is important for this research to clarify some of these issues.

The "stability" of the SD model

A first issue has to do with the stability of the SD model developed. The SD process, as described in this section, can be considered as implying a two-phase perspective of "tool development" followed by "tool application". This perspective suggests the idea of a "finalised" model emerging from the "tool development" phase, and thereafter being kept unchanged while being used as a "solution-finding tool". While this is not completely true in more "classic" modelling

approaches, it is even less the case in System Dynamics where the search for better solutions implies testing structural changes in the model, as originally stressed by Forrester (1961). Furthermore, while the model is being used and new insights are gained, the need for structural changes and improvements emerges. The continuous change of the model structure throughout the modelling process is a characteristic of the SD approach, very much emphasised by its iterative nature.

Qualitative versus Quantitative System Dynamics

Another important issue has to do with the relationship between the qualitative and the quantitative sides of System Dynamics. This is recognised as a crucial issue that still remains unclear (Richardson 1996). When Forrester introduced Industrial Dynamics, the modelling process was presented as evolving towards the development and use of a simulation model. Later, the emergence of diagramming tools to conceptualise the structure of feedback systems (Morecroft 1980) has led to the use of influence diagrams, and with this the emergence of the qualitative phase in the SD process. Some authors then started advocating the use of IDs as a necessary requirement to develop simulation models (Wolstenholme and Coyle 1983, Wolstenholme 1990, Coyle 1996). The same authors further proposed that this qualitative phase would have its own role within the methodology, having great potential to provide insights and even to deliver solutions (Coyle 1996, 1999). Further developments suggested a combined use of IDs with other qualitative techniques, like cognitive mapping (see Eden 1994, Ackerman et al 1991). These developments reflected a growing interest in the use of qualitative techniques, without recurring to simulation modelling. However, this idea, that solutions can be found at the qualitative level, contrasts with on-going arguments from other authors that reliable inference about policy implications cannot be achieved without simulation (Richardson 1991, Sterman 1994). It is the author's opinion that perhaps the most promising way ahead is towards improving the techniques on both sides and further integrate them formally.

SD notation: "word-and-arrow" versus "level / rate" diagrams

The emergence of the qualitative phase of System Dynamics has also led to some disagreements regarding the notation used to represent the feedback structure of a System Dynamics model. The two alternative representations are "word-and-arrow" influence diagrams, as already described in this chapter, and the "level/rate" diagrams also presented in this chapter as part of a quantitative simulation model. The initial notation proposed by Forrester (1961) was the "level/rate" type of diagram, where the primary aim of the SD process was the development and use of a quantitative simulation model. However, soon the "word-and-arrow" type of notation emerged as being preferred to represent and identify feedback loops and structures, prior to the use of "level/rate" diagram, which afterwards would be used as the basis for quantification (e.g. Roberts 1974, Richardson 1981). Later Wolstenholme (1990) suggested that both type of representations would be identical (referring to "level/rate" diagrams as "pipe diagrams"). Eventually, the preference for "word-and-arrow" diagrams has encouraged some authors to use these diagrams at detailed levels of formality (Wolstenholme and Coyle 1983), where they represent directly the structure of the quantified simulation model (e.g. Coyle 1996 – see "Cosmic" software tool accompanying the book). On the other hand, other authors preferred the use of "level/rate" diagrams only, even at the qualitative level (e.g. Richmond 1990). The preference for "level/rate" diagrams was strongly based on the importance of policies and their direct identification with "rates" in a "level/rate" diagram. This focus on model development centred around policies, was further explored by Morecroft (1982, 1984).

The shared view within the SD community appears to be a flexible approach wherein both types of representations are complementary one another. Each can be used with more or less emphasis, depending on the problem, on the audience and on the purpose of the model. Beyond this apparent compromise, it is the authors' opinion that each representations has its own strengths and weaknesses, and hence they should be used in combination. The "word-and-arrow" IDs are less formal because they do not impose any classification of the "words" (i.e. variables) present in the diagram. On the other hand, "level/rate" diagrams impose such classification: a variable is either a level (i.e. accumulation), a rate (i.e. policy or

system change), an auxiliary (i.e. information about the system, exogenous factors), or an information link (i.e. a causal effect). "Word-and-arrow" diagrams can be used to link aggregated concepts, according to more subjective and possibly more aggregated influences. Therefore they look more appropriate to identify individual feedback loops and more general feedback structures, at the higher strategic level of aggregation. On the other hand, the formality of "level/rate" diagrams suggests a final level of detail ready for quantification. This diagram is therefore more appropriate for a direct translation into a simulation model. Another strength of the "level/rate" notation is its appropriateness to represent the life-cycle of the system's entities, which is not so clearly represented in a "word-and-arrow" diagram. The feedback structure of a managed system involves the presence of material or abstract entities which have their own "life-cycle" of states through which they flow. These life-cycles are an important element of a system, which can be a good starting point for model conceptualisation (Wolstenholme 1990). Again, this entity / life-cycle perspective is in general more detailed than the high-level view assumed in the "word-and-arrow" IDs.

Based on these arguments, in this research "word-and-arrow" influence diagrams will be used in the qualitative phase of System Dynamics. "Level/rate" diagrams will be used in the quantitative phase, as an integrative part of the simulation model itself.

"Modelling to learn" versus "modelling to find a solution"

One of the novelties claimed by System Dynamics was its "openness" to managers (i.e. white-box modelling) and its role in organisational learning. This has been advocated as a "modern view" of modelling, at the core of the methodology (Morecroft and Sterman 1994). On the other hand, practical applications show that the more classic "solution-finding" route has been followed with success (e.g. Cooper 1980, PMMS 1993, Williams et al 1995). Perhaps this reflects that there is a wide domain of application, where different routes can be followed (Lane 1995). The key difference between these two routes is the main *purpose* of the modelling process: "modelling to learn" or "modelling to find a solution". This issue has various important implications in the modelling process: how close should

managers be involved? Should models be simple or complex? is there a clear distinction between model development and model use?

"Modelling to learn" refers to applications where the modelling process is aimed at creating a learning environment. Here, the involvement of managers as "model builders" is essential (Morecroft 1994). The models are likely to be simple, since the main source of learning is the iterative process through which managers develop, run and readjust the models. On the other hand, "solution-finding" applications usually demand a more rigorous correspondence of the model to the real world, hence restricting simplifications. Because reality is complex, the result is often a more complex model. In this scenario, the "model development" phase is usually undertaken by a team of experts. Managers are partially involved just to provide the required input. A "finalised" model is then used test alternative solutions and the most satisfactory one is selected and proposed to managers. Practical applications indicate that both routes can be implemented with success. They can both take advantage the distinctive benefits of System Dynamics. Ultimately, it is the practical success of the various types of application which dictates their appropriateness.

Although the System Dynamics methodology has been applied extensively in practice, there are still some critical issues that remain be solved. They have been the subject of much disagreement and discussion in the past and are currently the focus of on-going research. The following sub-sections discuss some of these issues. Particular attention is given to the subject of model validation, which is critical for the purpose of the research here presented.

Model validation

Overview

In a previous sub-section the System Dynamics process was presented in a structured manner. The methodology holds the promise of delivering a new modern view of systemic modelling of social systems. However, there are also some unsolved critical issues which are likely to have a major impact on the future

of the methodology. Criticisms regarding both conceptual and practical aspects have been following the emergence and evolution of the methodology. This sub-section discusses perhaps the most critical issue: model validation.

Validation is simultaneously such a critical and passionate issue in the modelling of social systems, that I would like to start by stating those principles which underlay most of the meaning of the research proposed in this work. Throughout this section, I will try to develop a brief but consistent rationale to support these principles, as well as regarding all related issues. The proposed principles are as follows:

- what is "to predict"? In a model of a social system, to predict is to anticipate a future scenario implied in the human knowledge and "beliefs" incorporated in the particular model.
- what does it mean to say that a prediction is "accurate"? A prediction produced by the model is accurate if this anticipated future scenario has a good chance to be achieved;
- when is a prediction "correct"? A prediction will proof correct if the expectations are fulfilled;
- what is a "valid model"? A valid model is not the one that represents reality as it is, delivering a true image of the inevitable future. Instead, a valid model is the one that describes the system in accordance with the modellers' mental models, while delivering consistent and achievable images of the future;
- what is a "useful model"? In real situations, there is usually a range of possible prediction which are both consistent and achievable. The useful model is the one that helps the analyst to select and plan better achievable futures.

The usefulness of the research here proposed rests on these principles. The following sub-sections provide a discussion and rationale.

The early criticism to validity in System Dynamics

It is important to note that the problem of *model validation* is not unique to the System Dynamics field. Apparently, the concept is not fully understood and agreed within the wider field of Operational Research (see Landry and Oral 1993 for a

discussion). In System Dynamics, there has been heated debate on this issue since the early days of the methodology (for a more detailed discussion see Barlas 1990; for particular examples see also Ansoff and Slevin 1968, Forrester 1968, Nordhaus 1973, Forrester et al 1974, Forrester and Senge 1980, and Zelner 1980).

One of the most ferocious attacks to System Dynamics came from Berlinsky (1976), who comments about the novelty of SD models in the following way (pp 45): "...the apparatus that is actually developed is nothing more than the traditional method of handling changes through time, by means of differential equations. The principles of systems [as proposed by Forrester] that were to hold universally turn out to involve nothing more than a clumsy application of the calculus." This author conceives a SD model as an attempt to achieve a "true" mathematical formulation of a system's dynamic behaviour over time. It follows that his criticisms are directed to the inappropriateness and the difficulty in validating the many equations used in a model like this – in particular, in the models presented in the *World Dynamics* and *Limits to Growth* (Berlinsky 1976, pp 75). It can be argued that there is certainly much more in System Dynamics than the goal of attaining a "mathematical formulation" for a system, as it generally happens in the natural sciences. Any attempt to structure knowledge involves the exercising and sharing of mental models, which in turn leads to improved understanding. Furthermore, and more important, quantitative modelling in social systems must be approached under a different perspective than in the engineering and in the natural sciences fields. As it will be discussed, in the social sciences the concept of "validity" of a model requires a shift in perspective from the path of "realism" to the path of "constructivism". In turn, this further implies a different perspective regarding the concepts of "prediction" and "accuracy" (often misused to distrust the validity of models in social sciences).

The problem of validity in models of social systems

Conceptually, a "perfectly valid" model would be no more than an exact replica of the real system being modelled. However, not only this is not possible as the aim of a model is to provide a simplified view of reality, retaining only what is relevant for the specific problem. Furthermore, a model usually targets a specific sub-

domain of the real world, and hence it tends to divorce a system from its environment. From here, problems regarding validity emerge: how to judge what is relevant and what is negligible in the real world? How to ensure that a system "handicapped" from what appears to be negligible, will behave in the same way as in the real world? The search for satisfying answers to these questions leads inevitably to some "concessions": the system, as represented in the model, will not be required to behave exactly in the same way as in the real world; instead, an approximate behaviour is acceptable, and only regarding those aspects of concern to the problem.

Isolating a system from its environment is always a problematic task: in the real world everything affects everything. While the modeller should focus only on the relevant interactions, the truth is that our understanding of how complex social systems work is poor. Flood (1987) asserts that the concept of complexity emerges in great part from this lack of knowledge about the system, which restricts our ability to judge what elements are relevant and how they interact within the system. Checkland (1981) argues that there are other difficulties beyond the problem of system complexity: in human systems its elements are aware of, and are affected by, their own awareness about the system. For example, if a model which is used for planning and control in a project affects the life of the staff, how will that same people counter-react to the use of the model itself? Since the act of observation may have a great impact on the system, this results in the theoretical impossibility of "predicting" the future. In theory, the problem could only be countered by a model capable of incorporating within its own structure the process by which its own use as a predictive tool would affect the system structure – something hardly conceivable in social systems, for both practical and theoretical reasons.

The solution: a shift from "realism" to "constructivism"

Roy (1991) argues that a satisfactory answer to these and other conceptual problems requires a shift in perspective from the path of "realism" to the path of "constructivism". The first considers that a system is independent from human observations and can be isolated from its environment. On the other hand,

constructivism considers explicitly that, given the above restrictions, while a model must not be taken as a true final representation of reality, it can incorporate valuable knowledge in a consistent manner. This knowledge can be "exercised" by testing hypothesis with the model. This analysis can produce valuable conclusions and recommend well defined solutions, which otherwise could have not been achieved (i.e. without the "imperfect" model). The "gain" from "constructivism" is that the use of "imperfect" knowledge within the model is acceptable. The concession is that the results produced, in particular numerical results, can no longer be seen as accurate predictions (unlike with a model of a "well-known" and "observer-independent" system).

According to this perspective, a model of a social system always represents a "biased" view of the reality. The predictive results produced are therefore implied by a relative view. Since in social systems humans tend to adjust their behaviour according to anticipated expectations, a model's prediction is better considered as an "achievable target", rather than as an inevitable event independent from the prediction. Therefore, *to* predict *is to anticipate a future scenario implied in the human knowledge and "beliefs" incorporated in the particular model. A prediction produced by the model is* accurate *if this anticipated future scenario has a good chance to be achieved; and the prediction will proof* correct *if the expectations are fulfilled.* A valid model is *not* the one that represents reality as it is, delivering a true image of the inevitable future. Instead, *a* valid *model is the one that describes the system in accordance with the modellers' mental models, while delivering consistent and achievable images of the future.* In real situations, there is usually a range of possible consistent and achievable predictions. *The* useful *model is the one that helps the analyst to select and plan better achievable futures.*

Model validity defined in this way depends on two elements:

- the rigour with which the consistency between the formal model structure and the modellers' mental model can be verified one another;
- how *useful* the model proofs to be in practice by producing "good predictions" (i.e. good achievable results).

Clearly, validation cannot be achieved by means of a single and objective formal procedure. Instead, model validation will rest on a progressive process of building *confidence.*

Proposed frameworks for validity in System Dynamics

In the System Dynamics field, the issue of model validation has always been approached around the idea of model *usefulness* in fitting a purpose: "The validity of a model should be judged by its suitability for a particular purpose... validity as an abstract concept, divorced from a purpose, has no useful meaning." (Forrester 1961). Richardson (1981) emphasises the two important aspects of validation in SD, which he terms as "suitability" and "consistency". Suitability means that a model must be able to address the plausible alternatives that ensure improved behaviour. Consistency means that the mechanisms represented in the model must correspond, as close as possible, to how the modeller perceives the mechanisms in the real system. Furthermore, the model must not only help to identify good solutions, but it must also be able to *explain* the predictions. Richardson (1981) further argues that the ultimate test of model validity would therefore consist in waiting to see whether, once implemented, the policies recommended by the model would produce the predicted results, *and* whether that happened for the same reasons as explained by the model (i.e. the predicted results for the predicted reasons). Not only such test is difficult to implement, as it would require a prohibitive amount of time, of effort, and of risks, so that the specific model could be tested several times.

In order to cope with the problem of validation, Forrester and Senge (1980) proposed a well defined set of confidence tests. They argue that "...there is no single test that serves to validate a SD model. Rather, confidence in a SD model accumulates gradually, as the model passes more tests, and as new points of correspondence between the model and the empirical reality are identified." Richardson (1996) argues that this work is still the most comprehensive statement on model validation in System Dynamics. Homer (1983) further advanced the idea of "partial-model testing", which applies to circumstances where dis-aggregated information is not available to estimate parameters, or select formulations. Barlas

(1985, 1989, 1996) has dedicated a considerable amount of effort to the quest of validation. He provides an interesting discussion about the philosophical roots in some depth (Barlas and Carpenter 1990). This author has also recently suggested that while validation in its essence is a continuous and gradual process throughout the SD process, there are benefits in having a well defined phase of "formal validation", wherein confidence tests are carried out more intensively in a well structured manner (Barlas 1996). Lane (1995) also proposes extensions to the basic framework of model validation, based on confidence tests. In turn, Peterson and Eberlein (1994) argue that in practice confidence tests are usually not implemented as extensively as desired, and are not properly documented. These authors developed a facility to define and include tests with the model and automatically executing them.

The importance of model legitimisation

Recently, Landry et al (1996) have argued about the importance of the differences, and relationship, between the distinctive concepts of model "validation" and model "legitimisation" in the field of OR. The authors argue that although the two concepts overlap, they are different: validation relates to how confidently the model can be taken as a representation of reality; on the other hand, legitimisation relates to the model being accepted within the persons with the organisation where it will be used. The fundamental difference is that the first refers concept to a scientific code, while the second refers to a social code. While validation should be considered as exhaustively as necessary for legitimisation, this is not a sufficient condition. On the other hand, legitimisation must not jeopardise validation: the modeller must not sacrifice, in any circumstance, the model scientific correctness for the sake of acceptability. These authors conclude that model legitimisation in OR is very important for the success of any modelling approach, which has unfortunately been so often overlooked. The importance of model legitimisation is a crucial to the success of System Dynamics, in particular because of the audience of a SD model (i.e. top executives) and because this audience often seeks good predictions from the model.

Further critical issues

Overview

An interesting discussion about the future of System Dynamics was recently presented by Richardson (1996). This author identifies what he believes to be the most crucial current problems for the future of System Dynamics:

- the need for tools to aid understanding model behaviour;
- accumulating wise modelling practise;
- advancing practise to further levels of knowledge and skill;
- accumulating results achieved in particular types of systems;
- making models accessible to a wider audience;
- qualitative mapping and formal quantitative modelling: when to use?
- the need to widen the range of people that can potentially understand SD;
- the need for well defined and robust procedures to ensure confidence and validation of SD models within the various types of application.

The author discusses each of these topics individually and concludes that other problems of same importance are likely to be beyond this list. Hence much needs to be done in the future. A more recent review of the current trends in System Dynamics undertaken by the same author (Richardson 1999) reveals that while some progress has been made in some of these areas, the need for further deeper developments still remains.

As a modelling technique, the methodology itself has important characteristics which affect its scope of application. Some of these issue are here discussed separately.

In first place, some criticisms emerged and are still being debated in respect to the actual capability of SD to address the many facets of complexity in social systems: a brief discussion presented by Dash and Murthy (1994) suggests that "...it is clear that the range of complexity explicitly addressed by SD is rather narrow", and that it gives privilege to "computational complexity" and "non-linearities". Other criticisms

include the inadequacy of differential equations for the description of cognitive processes in decision-making, and to other aspects of systems behaviour (Bossel 1977). It is argued that the simplistic modelling of the decision making processes as a "pre-programmed stimulus-response-transformation", and the excessive "endogenisation" of uncertainty regarding future scenarios, leads to a kind of determinism which does not exist in reality.

While these criticisms are debatable, it is a fact that some of the characteristics of SD restrict the scope of its application, and have implications for model validation. These characteristics include: aggregation, continuity, endogenous perspective, quantitative simulation, incorporation of human factors, and incorporation of decision-making processes, among others. These are some of the SD features that must be handled with special attention.

The endogenous perspective: a strength or a limitation?

The endogenous perspective of System Dynamics may imply that the system, as represented in the model, has little or no interaction with the environment: the core principle of the SD approach is that the system behaviour is primarily generated by its internal feedback structure. For this, the methodology has been criticised to assume a "closed view" of the system and hence inappropriate for social systems, since their behaviour depends considerably on their continuous interaction with the surrounding environment (see Eden and Harris 1975, and Richardson 1991 for a discussion). This problem reinforces the need to ensure that the specific problem being addressed is essentially of endogenous nature, and hence a "closed view" of the system is acceptable. Richardson (1991) also argues that if the interactions of a certain part of the environment are really important, then these must be included within the closed boundaries of the model.

A continuous perspective in a discrete world?

The continuous perspective of the approach also raises some problems: viewing the system as a set of continuous of material and information flows is, in some ways, contrary to the idea that many important events and changes in social

systems are of discrete nature. In fact, most people, including managers, tend to see the world as discrete. The use of averages and the aggregation of system entities and processes into flows, may therefore be an obstacle to the user in understanding the model. While there are ways of incorporating discrete events in SD models (e.g. see Coyle 1996), the underlying perspective is still continuous. Continuity does demands a shift in perspective that must be well understood by both modeller and user. In order to become comfortable with the continuous perspective, a strategic and longer-term view of the events in the real world is required. Experience with SD modelling and managerial maturity help.

Aggregation: overlooking fundamental "seeds" of behaviour?

Aggregation may also be a problem. Aggregation is related with continuity: both impose a high level perspective of the system. It is often argued that in social systems, major changes can emerge suddenly from within underlying sub-structures at the bottom of the system. These low-level processes responsible for these "internal shocks" are usually overlooked by the level of aggregation adopted in a SD model. Given the practical inadequacy of SD in capturing these low level processes, the alternative is to act on the real system in order to keep these shocks as under control as possible. In other words, efforts must be made so that the real system becomes an appropriate candidate for the aggregated perspective of SD.

Quantifying the unquantifiable?

As previously mentioned, a SD model incorporates and quantifies human factors. The methodology makes a point in capturing explicitly this type of subjective factors, arguing that they have a fundamental impact on the system behaviour. However, this requires valid scales for measurement as well as the availability of reliable procedures of data collection. It also requires a valid quantification of the interactions that this type of factors have with the rest of the system (e.g. within a project, how to measure "schedule pressure" and how to quantify its impacts on "productivity" and "error generation"?). This is perhaps one of the most critical problems when a model is used with the purpose of producing accurate estimates. Some factors and relationships are simply not measurable, and hence validation

cannot be achieved through a direct match with reality. Expert judgement from those who have experience with the system is crucial. This judgement should be further refined by exercising with model the recreation of historical scenarios. The use of "partial-model testing" techniques may also be helpful regarding this matter (Homer 1983; see also Graham 1977, Hamilton 1977, and Peterson 1977).

It should be noted however that the quantification of subjective factors and effects is an inherent difficulty of any technique which attempts to quantify this type of "soft" factors. One of Forrester's strongest arguments is that an approximation to these factors is far less damaging than simply omitting them.

Modelling decision-making: simulating managers minds?

Another ambitious aim of System Dynamics models is to incorporate and reproduce the decision-making processes, which in the real world generate managerial decisions. Simulating how people take decisions in real-life can be expected to be extremely difficult. Bossel (1977) criticises the SD approach because, he argues, it considers these processes in an overly simplistic manner, which he calls a "pre-programmed stimulus-reaction-transformation". It should be noted however, that the adequacy of a SD model to reproduce managerial decision-making is explicitly restricted to a limited set of simple and general decision-rules. The underlying assumption is that the lack of managerial understanding is not in each of the individual decisions, but on how, altogether, these decision should be combined over-time to ensure a better managerial control. If complex decisions which are well beyond reach of mathematical formulation are relevant to the system, then this problem can be approached through "flight simulation", where the model interacts periodically with the players (i.e. managers), leaving them the responsibility of generating these decisions. Another alternative is to develop more complex decision-making structures within the SD model, which will probably incorporate a composite of various simpler decision-rules.

A static model to represent a dynamic system?

Finally, another problem is the static perspective of the system structure of a SD model, during the simulation. In reality, the feedback structure of a real system is likely to be affected by the system behaviour and will therefore it may change over time. As some relationships become loose and others emerge, new feedback loops are created and others disappear. While some of these changes are beyond managerial direct control, in many cases they will reflect transformations within the decision-making processes (in particular changes in the information which is used as inputs to the decision rules). In order to cope with this problem, structural parameters can be used to model policy changes (see Coyle 1996). However, this technique forces the modeller to anticipate all the alternative policies, and thereby the emergence of new feedback structures. For those structural changes which are difficult to anticipate, either because they are too complex or because they are beyond management control, it will be difficult to consider them in the model in this way. It is therefore clear that the system targeted should preferably exhibit a fairly stable feedback structure within the time horizon of the analysis. It is also desirable that, if relevant structural changes occur, these can be anticipated with confidence within the time horizon of the analysis.

All the issues discussed above are critical to the success of a SD modelling application, in particular model validation. However, it should also be noted that the difficulties identified steam from the ambitious aim of the methodology. Any other approach attempting to address the same type of problems within complex social systems, is just as likely to face these same difficulties. In other words, these limitations are not a characteristic of the SD methodology but rather a result of its ambitious aim.

References

Abdel-Hamid, T. K., and Madnick, S. E. (1983). "The Dynamics of Software Project Scheduling." Communications of the ACM, 26(5), 340-346.

Abdel-Hamid, T. K., and Morecroft, J. D. W. (1983) "A Generic System Dynamics Model of Software Project Management." 1983 International System Dynamics Conference, Manor College, Chestnut Hill, MA.

Abdel-Hamid, T. K. (1984). "The Dynamics of Software Development Project Management: An Integrative System Dynamics Perspective," Ph.D. Thesis, Massachusetts Institute of Technology.

Abdel-Hamid, T. (1988). "Understanding the 90% Syndrome in Software Project Management: a Simulation Case Study." The Journal of Systems and Software, 8, 319-330.

Abdel-Hamid, T. K. (1989). "The Dynamics of Software Project Staffing: A System Dynamics Based Simulation Approach." IEEE Transactions on Software Engineering, February, 109-119.

Abdel-Hamid, T., and Madnick, S. (1990). "The Elusive Silver Lining: How We Fail to Learn from Software Development Failures." Sloan Management Review, Fall, 39-48.

Abdel-Hamid, T. (1990). "On the Utility of Historical Project Statistics for Cost and Schedule Estimation: Results from a Simulation-based Case Study." Journal of Systems Software, 13, 71-82.

Abdel-Hamid, T., and Madnick, S. E. (1991). Software Project Dynamics: an integrated approach, Prentice-Hall, New Jersey.

Abdel-Hamid, T. (1992). "Investigating the Impacts of Managerial Turnover / Succession on Software Project Performance." Journal of Management Information Systems, 9(2), 127-144.

Abdel-Hamid, T. (1993). "A Multiproject Perspective of Single-Project Dynamics." Journal of Systems Software, 22, 151-165.

Ackerman, F., Eden, C., and Williams, T. (1997). "A persuasive approach to delay and disruption – using "mixed methods"." Interfaces, 27(2), 48-65.

Alonso, W. (1968). "Predicting best with imperfect data." Journal of the American Institute of Planners, 34(4), 248-255.

Anderson, J. C., Cleveland, G., and Schroeder, R. (1989). "Operations Strategy – A Literature Review." Operations Management, 18(2), 1-26.

Andrews, K. R. (1980). The Concept of Corporate Strategy, Homewood, IL, Irwin.

Ansoff, H. I., and Slevin, D. P. (1968). "An Appreciation of Industrial Dynamics." Management Science, 14(7), 383-397.

Barlas, Y. (1985). "Validation of System Dynamics Models with a Sequential Procedure Involving Multiple Quantitative Methods", unpublished Ph.D. Thesis, Georgia Institute of Technology.

Barlas, Y. (1989). "Multiple test for validation of system dynamics type of simulation models." European Journal of Operational Research, 42, 59-87.

Barlas, Y., and Carpenter, S. (1990). "Philosophical roots of model validation: two paradigms." System Dynamics Review, 6(2), 148-166.

Barlas, Y., and Bayraktutar, I. (1992) "An Interactive Simulation Game for Software Project Management (Softsim)." 1992 International System Dynamics Conference, Utrecht, the Netherlands, 59-68.

Barlas, Y. (1994) "Model Validation in System Dynamics." 1994 International System Dynamics Conference, Stirling, Scotland.

Barlas, Y. (1996). "Formal Aspects of Model Validity and Validation in System Dynamics." System Dynamics Review, 12(3), 183-210.

Berlinski, D. J. (1976). On systems analysis: an essay concerning the limitations of some mathematical methods in the social, political and biological sciences. MIT Press, Cambridge, US.

Berson, A., and Smith, S. (1997). Data Warehousing, Data Mining, and OLAP, McGraw-Hill, New York.

Boehm, B. (1981). Software Engineering Economics, Prentice-Hall, New Jersey.

Bossel, H. (1977). Concepts and Tools of Computer-Assisted Policy Analysis, Birkhäuser Verlag, Basel.

Boulding, K. (1956). "General Systems Theory: The Skeleton of Science." Management Science, April, 197-208.

Bowers, J. (1994). "Data for project risk analyses." International Journal of Project Management, 12, 9-16.

Bowers, J. (1995). "Criticality in Resource Constrained Networks." Journal of the Operational Research Society, 46, 80-91.

Bowers, J. (1996). "Identifying Critical Activities in Stochastic Resource Constrained Networks." Omega, International Journal of Management Science, 24(1), 37-46.

Brandl, D., and Worley, J. (1993). "An implemented Object Model of the Software Engineering Process." Journal of Systems Software, 23, 171-181.

Burdick, R. B., Mullen, T. W., and Rodrigues, A. G. (1998). "The impact of software project management on quality." Cutter IT Journal, 11(9), 30-38.

Burns, J. R. (1977). "Converting Signed Digraphs to Forrester's Schematics and Converting Forrester's Schematics to differential Equations." IEEE Transactions on Systems Management and Cybernetics, SMC-7(10).

Burns, J. R., and Marcy, W. M. (1979). "Causality: Its Characterization in System Dynamics and KSIM Models of Socioeconomic Systems." Technological Forecasting and Social Change, 14(4), 387-398.

Cabanis-Brewin, J. (1999). ""So...So What??": Debate Over CCPM Gets a Verbal Shrug from TOC Guru Goldratt." PM-Network, 13(12), 49-52.

Cellier, F. (1982). Progress in Modelling and Simulation, Academic Press, London.

Chapman, C., and Ward, S. (1997). Project Risk Management: processes, techniques and insights, John Wiley & Sons, Chichester.

Checkland, P. (1981). Systems Thinking, Systems Practice, John Wiley & Sons, Chichester.

Checkland, P., and Scholes, J. (1991). Soft Systems Methodology in Action, John Wiley & Sons, New York.

Churchman, C. (1968). The Systems Approach, Dell, New York.

Cleland, D., and King, W. (1975). Systems Analysis and Project Management, McGraw-Hill, New York.

Conrad, T. (1997). "Managing Complex Software Projects – Experiences With Real-Time Mission-Critical Systems." Managing and Modelling Complex Projects, T. Williams, ed., Kluwer Academic Publishers, Dordrecht, 123-143.

Cooper, K. (1980). "Naval Ship production: A Claim Settled and a Framework Built." Interfaces, 10(6), 30-36.

Cooper, K., and Mullen, T. (1993). "Swords and Plowshares: The Rework Cycles of Defense and Commercial Software Development Projects." American Programmer, 6(5), 41-51.

Cooper, K. (1993). "The Rework Cycle: benchmarks for the project manager." Project Management Journal, 24(1).

Cooper, K. (1994). "The $2,000 hour: how managers influence project management through the rework cycle." Project Management Journal, 25(1), 11-24.

Cooper, K. (1997). "System Dynamics Methods in Complex Project Management." Managing and Modelling Complex Projects, T. Williams, ed., Kluwer Academic Publishers, Dordrecht.

Cooper, K. (1999). "Power of the People." PM-Network, 13(7), 43-47.

Coyle, R. G. (1973). "On Scope and Purpose of Industrial Dynamics." International Journal of System Science, 4(3), 397-406.

Coyle, R. G. (1977). Management System Dynamics, John Wiley & Sons, Chichester.

Coyle, R. G. (1996). System Dynamics Modelling: A Practical Approach, Chapman & Hall, London.

Coyle, G. (1999) "Qualitative Modelling in System Dynamics or What are the Wise Limits of Quantification." 1999 International System Dynamics Conference, Wellington, New Zealand, 30.

Dash, D. P., and Murthy, P. N. (1994). "Boundary Judgement in System Dynamics Modeling: An Investigation Through the Science of Complexity." Systems Practice, 7(4), 465-475.

Davidson, F., and Huot, J.-C. (1991). "Large-scale Projects – Management Trends for Major Projects." Cost Engineering, 33(2), 15-23.

DeMarco, T. (1982). Controlling Software Projects, Prentice-Hall, New Jersey.

Déry, R. (1993). "Revisiting the issue of model validation in OR: an epistemological view." European Journal of Operational Research, 66, 168-183.

Duncan, W. (1995). "Developing a project-management body-of-knowledge document: the US Project Management Institute's approach, 1983-94." International Journal of Project Management, 13(2), 89-94.

Duncan, W. (1999a). "Back to Basics: Charters, Chains, and Challenges." PM-Network, 13(4), 29-30.

Duncan, W. (1999b). "Reader Feedback – Response to a Letter to the Editor." PM-Network, 13(8), 5.

Easterby-Smith, M., Thorpe, R., and Lowe, A. (1991). Management Research – An Introduction, SAGE Publications, London.

Eden, C., and Harris, J. (1975). Management Decision and Decision Analysis, MacMillan Press Ltd, London.

Eden, C. (1988). "Cognitive Mapping: A Review." European Journal of Operational Research, 36, 1-13.

Eden, C. (1994). "Cognitive Mapping and Problem Structuring for System Dynamics Model Building." System Dynamics Review, 10(2/3), 257-276.

Eden, C., and Huxham, C. (1996). "Action Research for the Study of Organisations." Handbook of Organisation Studies, C. H. S. Clegg, W. Nord, ed., Sage Publications, London, 526-542.

Elmaghraby, S., Baxter, E., and Vouk, M. (1995). "An Approach to the Modeling and Analysis of Software Production Processes." International Transactions on Operational Research, 2(1), 117-135.

Elmaghraby, S., and Aggerwal, M. (1997). "On the Expected Completion Time of Diffusion Activity Networks (DiAN)." Managing and Modelling Complex Projects, T. Williams, ed., Kluwer Academic Publishers, Dordrecht.

Fagan, M. (1976). "Design and Code Inspections to Reduce Errors in Program Development." IBM Systems Journal, 15(3), 182-211.

Fagan, M. (1986). "Advances in Software Inspections." IEEE Transactions on Software Engineering, 12(7), 744-751.

Flood, L. (1987). "Complexity: a definition by construction of a conceptual framework." Systems Research, 4, 177-185.

Ford, D. N. (1995). "The Dynamics of Project Management: An Investigation of the Impacts of Project Process and Coordination on Performance," Ph.D. Thesis, Massachusetts Institute of Technology.

Ford, D. N., and Sterman, J. D. (1998). "Dynamic Modelling of Product Development Processes." System Dynamics Review, 14(1), 31-68.

Ford, D. N. (1998) "A Behavioral Approach to Feedback Loop Dominance." 1998 International System Dynamics Conference, Quebec City, Canada, 33.

Forrester, J. (1958). "Industrial Dynamics: A Major Breakthrough for Decision Makers." Harvard Business Review, 36(4), 37-66.

Forrester, J. (1961). Industrial Dynamics, MIT Press, Cambridge, US.

Forrester, J. (1968). Principles of Systems, Productivity Press, Portland.

Forrester, J. (1968). "A Response to Ansoff and Slevin." Management Science, 14, 601-618.

Forrester, J. W., Mass, N. J., and Low, G. W. (1974). "The Debate on World Dynamics: A Response to Nordhaus." Policy Sciences, 5(2), 169-190.

Forrester, J. W., and Senge, P. M. (1980). "Tests for building confidence in system dynamics models." TIMS Studies in the Management Sciences, North-Holland Publishing Company, 209-228.

Gass, S. I. (1993). "Model accreditation: a rationale and process for determining a numerical rating." European Journal of Operational Research, 66, 250-258.

Gemmill, D., and Edwards, M. (1999). "Improving Resource-Constrained Project Schedules With Look-Ahead Techniques." Project Management Journal, 30(3), 44-55.

Goldratt, E. (1984). The Goal, North River Press, New York.

Goldratt, E. (1997). Critical Chain, North River Press, New York.

Goldratt, E. (1999). Project Management the TOC Way, North River Press, New York.

Golenko-Ginzburg, D. (1988). "Controlled Alternative Activity Networks in Project Management." European Journal of the Operational Research Society, 37, 336-346.

Golenko-Ginzburg, D., and Gonik, A. (1997). "Project Planning and Control by Stochastic Network Models." Managing and Modelling Complex Projects, T. M. Williams, ed., Kluwer Academic Publishers, Dordrecht, 21-44.

Goodman, M. (1974). Study Notes in System Dynamics, Productivity Press, Cambridge, US.

Graham, A. K. (1976). "Parameter Formulation and Estimation in System Dynamics Models." D-2349-1, M.I.T. System Dynamics Group.

Hamilton, M. S. (1976) "Estimating Lengths and Orders of Delays in System Dynamics Models." 1976 International System Dynamics Conference, Geilo, Norway, 162-182.

Homer, J. B. (1983) "Partial-model testing as a validation tool for system dynamics." 1983 International System Dynamics Conference, Manor College, Chestnut Hill, MA, 920-932.

Homer, J. B., Sterman, J. D., Greenwood, B., and Perkola, M. (1993) "Delivery Time Reduction in Pulp and Paper Mill Construction Projects: A Dynamic Analysis of Alternatives." 1993 International System Dynamics Conference, Cancun, Mexico, 212-221.

Jessen, S. A. (1988) "Can Project Dynamics Be Modelled?" 1988 International Systems Dynamics Conference, La Jolla, California, 171.

Jones, T. (1998). Estimating Software Costs, McGraw-Hill, New York.

Kelly, T. J. (1970). "The Dynamics of R&D Project Management," M.I.T. MS Mgt (300)

Keloharju, R., and Wolstenholme, E. F. (1989). "A Case Study in System Dynamics Optimization." J. Operations Research Soc., 40, 221-230.

Kerzner, H. (1998). Project Management: a systems to planning, scheduling and controlling, John Wiley & Sons, New York.

Landry, M., and Oral, M. (1993). "In search of a valid view of model validation for operations research." European Journal of Operational research, 66, 161-167.

Landry, M., Banville, C., and Oral, M. (1996). "Model legitimisation in operational Research." European Journal of Operational Research, 92, 443-457.

Lane, D. C. (1995) "The Folding Star: a comparative re-framing and extension of validity concepts in system dynamics." 1995 International System Dynamics Conference, 111-130.

Lane, D. C., and Smart, C. (1996). "Reinterpreting 'Generic Structure': Evolution, Application and Limitations of a Concept." System Dynamics Review, 12(2), 87-120.

Leach, L. (1999). "Critical Chain Project Management Improves Project Performance." Project Management Journal, 30(2), 39-51.

Lee, G.-S., and Murata, T. (1994). "A B-Distributed Stochastic Petri Net Model for Software Project Time / Cost Management." Journal of Systems Software, 26, 149-165.

Lee, C., Lin, H., and Lu, Y. (1994). "The theory of the PM-Net model." E9401, College of Management, National Sun Yat-Sen University.

Lin, C., and Levary, R. (1989). "Computer-aided software development process design." IEEE Transactions on Software Engineering, 15(9), 1280-1293.

Lin, C. Y. (1993). "Walking on Battlefields: Tools for Strategic Software Management." American Programmer, 6(5), 34-40.

Liu, L., and Horowitz, E. (1989). "A formal model for software project management." IEEE Transactions on Software Engineering, 15(10), 1280-1293.

Madachy, R. (1996) "Modelling Software Processes with System Dynamics: Current Developments." 1996 International System Dynamics Conference, Cambridge, Massachusetts, 333-336.

McKenzie, M., Tausworthe, R. C., Lin, C. Y., and Reifer, D. J. (1984) "A Dynamic-System Simulation Model of the Software Development Process." 1984 Summer Computer Simulation Conference, July, 889-904.

Miser, H. J. (1993). "A foundational concept of science appropriate for validation in operational research." European Journal of Operational Research, 66, 204-215.

Morecroft, J. D. W. (1982). "A Critical Review of Diagramming Tools for Conceptualizing Feedback System Models." Dynamica, 8(1), 20-29.

Morecroft, J. D. W. (1984). "Strategy Support Models." Strategic Management Journal, 5(3), 215-229.

Morecroft, J. D. W., and Sterman, J. D. (1994). "Modeling for Learning Organizations." System Dynamics Series, Productivity Press, Portland.

Morris, P. W. G., and Hough, G. H. (1987). The Anatomy of Major Projects: a study of the reality of project management, John Wiley & Sons, Chichester.

Morris, P. (1994). The Management of Projects, Thomas Telford, London.

Neelamkavil, F. (1987). Computer Simulation and Modelling, John Wiley & Sons, London.

Nicholas, J. M. (1990). Managing Business and Engineering Projects: Concepts and Implementation, Prentice-Hall, New Jersey.

Nordhaus, W. D. (1973). "World Dynamics: Measurement Without Data." The Economic Journal, 83(332), 1145-1183.

Ogata, K. (1992). System Dynamics, Prentice-Hall, New Jersey.

Oral, M., and Kettani, O. (1993). "The facets of the modelling and validation process in operations research." European Journal of Operational Research, 66, 216-234.

Patrick, F. (1999). "Getting Out From Between Parkinson's Rock and Murphy's Hard Place." PM-Network, 13(4), 57-62.

Peterson, D. W. (1976) "Statistical Tools for System Dynamics." 1976 International System Dynamics Conference, Geilo, Norway, 224-241.

Peterson, D. W., and Eberlein, R. L. (1994). "Reality Check: a bridge between systems thinking and system dynamics." System Dynamics Review, 10(2-3), 159-174.

Phillips, L. D. (1982). "Requisite Decision Modelling: A Case Study." European Journal of Operational Research, 33, 303-311.

Pidd, M. (1998). Computer Simulation in Management Science, John Wiley & Sons, Inc., New York.

Pinto, J. (1999). "Some Constraints on the Theory of Constraints." PM-Network, 13(8), 49-51.

Project Management Institute (PMI) (1996). A Guide to the Project Management Body of Knowledge, Project Management Institute, North Carolina.

Pressman, R. S. (1997). Software Engineering: a practitioner's approach, McGraw-Hill, New York.

Pritsker, A. (1977). Modeling and Analysis Using Q-GERT Networks, John Wiley & Sons, New York.

Pugh III, A. L., (1983). DYNAMO User's Manual, Productivity Press, Cambridge MA.

Pugh-Roberts Associates (1993). "Program Management Modeling System.", PA Consulting Group, Cambridge.

Putnam, L., and Myers, W. (1997). Industrial Strength Software: Effective Management using Measurement, Institute of Electrical & Electronic Engineer.

Putnam, L., and Myers, W. (1999). "Get the Estimate Right." Cutter IT Journal, 12(7), 13-24.

Randers, J. (1980). "Elements of the System Dynamics Method." , Productivity Press, Cambridge, 320.

Richardson, G., and Pugh III, A. (1981). Introduction to System Dynamics Modelling, Productivity Press, Cambridge, US.

Richardson, G. P. (1991). Feedback Thought in Social Science and Systems Theory, University of Pennsylvania Press, Philadelphia.

Richardson, G. P. (1995). "Loop Polarity, Loop Dominance, and the Concept of Dominant Polarity." System Dynamics Review, 11(1), 67-88.

Richardson, G. P. (1996). "Problems for the Future of System Dynamics." System Dynamics Review, 12(2), 141-157.

Richardson, G. (1999). "Reflections for the Future of System Dynamics." Journal of the Operational Research Society, 50(4), 440-449.

Richmond, B. M., Vescuso, P., and Peterson, S. O. (1990). iThink, High Performance Systems, Lyme, NH.

Rizzo, T. (1999). "Operational Measurements for Product Development Organizations." PM-Network, 13(12), 31-35.

Roberts, E. B. (1964). The Dynamics of Research and Development, Harper and Row, New York.

Roberts, E. B. (1974). "A Simple Model of R & D Project Dynamics." R & D Management, 5(1).

Roberts, E. (1978). Managerial Applications of System Dynamics, Productivity Press, Cambridge, USA.

Rodrigues, A. (1994a). "The Role of System Dynamics in Project Management: A Comparative Analysis with Traditional Models." 1994 International System Dynamics Conference, Stirling, Scotland, 214-225.

Rodrigues, A. (1994b). "Doctoral Research Work Report of Progress 1994", University of Stirling, Stirling.

Rodrigues, A., and Williams, T. (1995). "The Application of System Dynamics to Project Management: An Integrated Model with the Traditional Procedures," Working Paper 95/2, Department of Management Science, University of Strathclyde, Glasgow.

Rodrigues, A., and Bowers, J. (1996a). "System dynamics in project management: a comparative analysis with the traditional methods." System Dynamics Review, 12(2), 121-139.

Rodrigues, A. G., and Bowers, J. (1996b). "The role of system dynamics in project management." International Journal of Project Management, 14(4), 235-247.

Rodrigues, A. (1997) "SYDPIM – A System Dynamics-based Project-management Integrated Methodology." 1997 International System Dynamics Conference: "Systems Approach to Learning and Education into the 21st Century", Istanbul, Turkey, 439-442.

Rodrigues, A. G., and Williams, T. M. (1997). "System dynamics in software project management: towards the development of a formal integrated framework." European Journal of Information Systems, 6(1), 51-66.

Rodrigues, A. G., and Williams, T. M. (1998). "System dynamics in project management: assessing the impacts of client behaviour on project performance." Journal of the Operational Research Society, 49(1), 2-15.

Rodrigues, A. G. (1999a). "Finding a common language for global software projects." Cutter IT Journal.

Rodrigues, A. G. (1999b). "Letter to the Editor on CCPM." Cutter IT Journal, 12(10), 4-5.

Rosenhead, J. (1989). Rational Analysis for a Problematic World, John Wiley & Sons, Chichester.

Roy, B. (1993). "Decision science or decision-aid science." European Journal of Operational Research, 66, 184-203.

Seborg, D., Edgar, T., and Mellichamp, D. (1989). Process Dynamics and Control, John Wiley & Sons, New York.

Senge, P. (1990). The Fifth Discipline, Doubleday/Currency, New York.

Senge, P., Ross, R., Smith, B., Roberts, C., and Kleiner, R. (1994). The Fifth Discipline Fieldbook, Doubleday.

Sequeira, I. (1999). "Automated Cost Estimating System Using Neural Networks." Project Management Journal, 30(1), 11-18.

Shearer, J., Murphy, A., and Richardson, H. (1967). Introduction to System Dynamics, Addison-Wesley, London.

Simon, P., Hillson, D., and Newland, K. (1997). Project Risk Analysis and Management Guide, APM Group Ltd., Norwich.

Smith, J. H. (1993). "Modeling muddles: validation beyond the numbers." European Journal of Operational Research, 66, 235-249.

Sterman, J. D. (1984). "Appropriate Summary Statistics for Evaluating the Historical Fit of System Dynamics Models." Dynamica, 10(2), 51-66.

Tausworthe, R. C., McKenzie, M., and Lin, C. Y. (1983) "Structural Considerations for a Software Life Cycle Dynamic Simulation Model." Computers in Aerospace IV Conference.

Tavares, L. V. (1999). Advanced Models for Project Management, Kluwer Academic Publishers, Dordrecht.

Turing, A. M. (1950). "Computing Machinery and Intelligence." Mind, LIX(236), 433-460.

Turner, R. (1990). "What are projects and project management?". Working Paper 2/90, Henley The Management College, Henley-on-Thames.

Turner, J. R. (1993). The Handbook of Project-Based Management, McGraw-Hill, London.

Uyttewaal, E. (1999). "Take the Path That is Really Critical." PM-Network, 13(12), 37-42.

Weil, B., and Dalton, W. (1993). "Risk Management in Complex Projects." , Pugh-Roberts Associates, Cambridge, USA.

Weist, J. (1964). "Some properties of schedules for large projects with limited resources." Operations Research, 12, 395-418.

Wernick, P., and Lehman, M. (1998) "Software Process Dynamic Modelling for FEAST/1." ProSim'98 International Workshop on Software Process Simulation Modeling, Silver Falls.

Wideman, R. M. (1992). Project and program risk management, Project Management Institute, Upper Darby, PA.

Williams, T. (1992). "Criticality in stochastic networks." Journal of the Operational Research Society, 6(2), 353-357.

Williams, T. (1993). "A Classified Bibliography of Recent Research Relating to Project Risk Management." Working Paper 93/6, Department of Management Science, University of Strathclyde, Glasgow.

Williams, T. (1995). "A Classified Bibliography of Research on Project Risk Management: Addendum." Working Paper 95/6, Department of Management Science, University of Strathclyde, Glasgow.

Williams, T., Eden, C., Ackerman, F., and Tait, A. (1995). "The effects of design changes and delays on project costs." Journal of the Operational Research Society, 46(7), 809-818.

Williams, T. (1996). "The two-dimensionality of project risk." International Journal of Project Management, 14(3), 185-186.

Williams, T. M. (1997). Managing and Modelling Complex Projects, Kluwer Academic Publishers, Dordretch.

Williams, T. (1998). "A Classified Bibliography of Research on Project Risk Management: Further Continued." Working Paper 98/7, Department of Management Science, University of Strathclyde, Glasgow.

Willis, B. (1995). "APM project-management body of knowledge: the European view." International Journal of Project Management, 13(2), 95-98.

Wirth, I., and Tryloff, D. (1995). "Preliminary comparison of six efforts to document the project-management body of knowledge." International Journal of Project Management, 13(2), 109-118.

Wolstenholme, E. F. (1982). "System Dynamics in Perspective." Journal of Operational Research Society, 33, 547-556.

Wolstenholme, E. F., and Coyle, R. G. (1983). "The Development of System Dynamics as a Methodology for System Description and Qualitative Analysis." Journal of the Operational Research Society, 34(7), 569-581.

Wolstenholme, E. (1990). System Enquiry: A System Dynamics Approach, John Wiley & Sons, Chichester.

Wolstenholme, E. F., Henderson, S., and Gavine, A. (1994). The Evaluation of Management Information Systems, John Wiley & Sons, London.

Wolstenholme, E. (1999). "Qualitative vs. quantitative modelling: the evolving balance." Journal of the Operational Research Society, 50(4), 422-428.

Woodworth, B., and Shanahan, S. (1988). "Identifying the critical sequence in a resource constrained project." Project Management, 6, 89-96.

Wright, G. (1985). Behavioural Decision Making, Plenum Press, New York.

Yourdon, E. (1993). "Editor's Preface." American Programmer, 6(5), 1-2.

Zeigler, B. (1976). Theory of Modelling and Simulation, John Wiley & Sons, New York.

Zellner, A. (1980). "Comment on Forrester's "Information Sources for Modeling the National Economy"." Journal of the American Statistical Association, 75, 567-569.

Zultner, R. (1999). "Project Estimation with Critical Chain: Third-Generation Risk Management." Cutter IT Journal, 12(7), 4-12