PhD Theses in Experimental Software Engineering

Vol. 8

Dietmar Pfahl

An Integrated Approach to Simulation-Based Learning in Support of Strategic and Project Management in Software Organisations

An Integrated Approach to Simulation-Based Learning in Support of Strategic and Project Management in Software Organisations

Vom Fachbereich Informatik
der Universität Kaiserslautern
zur Verleihung des akademischen Grades

Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigte Dissertation
von

Dipl.-Math.-Oec. Dietmar Pfahl

Fraunhofer-Institut für Experimentelles Software Engineering
(Fraunhofer IESE)
Kaiserslautern

Berichterstatter:	Prof. Dr. H. Dieter Rombach Prof. Dr. Dieter Ehrenberg
Dekan:	Prof. Dr. Jürgen Avenhaus
Tag der Wissenschaftlichen Aussprache:	01.10.2001

D 386

Acknowledgments

The research presented in this dissertation could not have been done without the help of many colleagues and friends who supported me in many ways during all the years. Only the names of some of them can be listed here.

In the first place, I thank Prof. Dr. Dieter Rombach for being my principal advisor and for giving me the opportunity to work in the inspiring and stimulating environment created by the Fraunhofer Institute for Experimental Software Engineering (IESE). Many thanks go also to Prof. Dr. Dieter Ehrenberg for acting as a referee of my work and to Prof. Dr. Theo Härder for chairing the Ph.D. committee at the University of Kaiserslautern.

Much of the work conducted in the scope of this dissertation was initiated while I was with Siemens Corporate Research. I am grateful to my former managers Monika Gonauser and Dr. Karl-Heinrich Möller for offering me the chance to work in the field of software process simulation. Very special thanks go to Karl Lebsanft who was my immediate superior at Siemens. He supported my work from the very beginning and even during all the years after I had left Siemens and joined Fraunhofer IESE. Prof. Dr. Günther Ruhe is the person to whom I owed most during the last years of my research conducted at Fraunhofer IESE. He was my internal supervisor and a continuous source of encouragement, help and friendship. At Fraunhofer IESE I also had the luck to work with Prof. Dr. Lionel Briand and Dr. Khaled El-Emam, two outstanding researchers who taught me most - if not all - I know about empirical research in the field of software engineering.

Many of my former and current colleagues at Siemens and Fraunhofer IESE encouraged me to do the work presented in this dissertation and actively contributed to its success. I am grateful to all of them. Those to whom I am particularly indebted are Prof. Dr. Colin Atkinson, Dr. Andreas Birk, Dr. Frank Bomarius, Marcus Ciolkowski, Dr. Christiane Differding, Pierfrancesco Fusaro, Jean-Francois Girard, Dirk Hamann, Dr. Janne Järvinen, Reinhard Kammerer, Dr. Oliver Laitenberger, Roland Laqua, Prof. Dr. Peter Liggesmeyer, Jörg Lottermoser, Michael Ochs, Prof. Dr. Mohsen Rezagholi, Dr. Louise Scott, Dr. Rini van Solingen, Harald Thomas, Axel Völker, Dr. Hélène Waeselynck, and Dr. Isabella Wieczorek. I also received much help from students and research assistants, namely Jörg Dorsch, Martin Geier, Marco Klemm, Nataliya Koval, Tatyana Krivobokova, and Ioan Teleaga. Of the many researchers who I met outside Siemens and Fraunhofer IESE I am most grateful to Dr. Marc Kellner, Dr. Antony Powell, Prof. Dr. David Raffo, Prof. Dr. Per Runeson, and Dr. Ioana Rus for inspiring discussions.

Finally, I am grateful to my parents and my sister for their never ending patience, support and understanding. And last but not least, I want to thank Ute Wellnitz, my wife, friend and partner, for the encouragement and love that I received from her during many years of hard work.

Abstract

Industrial software development is a highly complex, dynamic task, which is not only determined by the choice of the right technologies but also – to a large degree – by the knowledge and skills of the people involved. The success of software organisations depends on its ability to facilitate continuous improvement of products and processes (strategic level) and on the effectiveness and efficiency of product development (project level). On both levels management takes the key role.

The focus of this PhD research is on simulation-based learning to support both strategic and project management in software organisations.

The main contribution of the research work lies in the design, application and validation of a framework for Integrated Measurement, Modelling, and Simulation (IMMoS). The IMMoS framework supports managers to cope with the dynamic complexity of software development by providing guidance on building and using quantitative simulation models as a source for learning and improvement.

Simulation models are valuable tools for managers because they help them understand the effects of new technologies and policies on the performance of software development processes. Based on simulations, managers can explore and analyse potential improvements before implementation and empirical evaluation of the related process changes in a pilot project. In addition, quantitative simulation models can be used to support planning and control tasks.

The core element of IMMoS is the simulation modelling method System Dynamics (SD), which integrates quantitative (black-box) and explanatory (white-box) modelling in a natural way. The novelty of IMMoS is twofold. Firstly, it enhances existing guidance for SD model development by adding a component that enforces goal-orientation, and by providing a refined process model with detailed description of activities, products, and roles involved in SD modelling and simulation. Secondly, it describes how to integrate SD modelling with established static black-box and white-box modelling methods, i.e. goal-oriented measurement and descriptive process modelling.

IMMoS has been successfully applied in industrial software organisations. The effectiveness and efficiency of IMMoS is supported with empirical evidence from two industrial case studies and one controlled experiment.

Table of Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

AARR	Actual Average Requirements Replacement
ALF	Advanced Software Engineering Environment Logistics Framework
ASM	Analytic Summary Model
BU	Business Unit
CMM	Capability Maturity Model
CMPM	Cellular Manufacturing Process Model
COCOMO	Constructive Cost Model
CORONET	Corporate Software Engineering Knowledge Networks for Improved Training of the Work Force
CT	Corporate Technology
DM	Dynamic Model
DPM	Descriptive Process Model
EF	Experience Factory
EPG	Electronic Process Guide
EPOS	Expert System for Program and System Development
GENSIM	Generic Simulator
GUI	Graphical User Interface
GQM	Goal Question Metric
HFSP	Hierarchical and Functional Software Process Description and Enaction
IEEE	Institute of Electrical and Electronics Engineers
IMMoS	Integrated Measurement, Modelling, and Simulation
ISO	International Standardisation Organisation
JAD	Joint Application Development
MAM	Measurement-based Analytic Models
MVP-L	Multi-View Process Modeling Language
NASA	National Aeronautics and Space Administration
PEM	Process Enactment Model
PM	Process Modelling
PMIM	Project Management Integrated Model
PMO	Project Management Office
PMT	Project Management

PPM	Prescriptive Process Model
PROFES	Product Focused Process Improvement for Embedded Software
PSIM	Process/Project Simulator
PSM	Process Simulation Model
PTA	Process Trade-off Analysis
QIP	Quality Improvement Paradugm
QM	Quantitative Model
RESIM	Requirements Engineering Simulator
SD	System Dynamics
SDM	System Dynamics Model
SDOM	System Dynamics Operational Model
SDSM	System Dynamics Strategic Model
SE	Software Engineering
SEI	Software Engineering Institute
SEPG	Software Engineering Process Group
SEPS	Software Engineering Process Simulator
SESAM	Software Engineering Simulation by Animated Models
SLICS	Software Life Cycle Simulator
SPADE	Software Process Analysis, Design and Enactment
SPI	Software process Improvement
SPICE	Software Process Improvement and Capability Determination
SPEARMINT	Software Process Elicitation, Analysis, Review, and Measurement in an Integrated Modeling Environment
WBT	Web-Based Training

1 Introduction

Software industry is constantly facing increasing demands for "better, faster, cheaper" and the increasing complexity of software products and projects have significantly "raised the bar" for software developers and managers to improve performance.

Software industry has received help in the form of new technologies, such as tools, programming languages, and development methods. However, since software development is an inherently human-based activity, one of the key questions is "How can (new) technologies and people work together in order to deliver products with better quality, within schedule and budget?". A crucial role in answering this question plays the management function.

In general, management in software organisations takes place on two levels: project level and organisational level. Project management is responsible for the planning and successful execution of software (or system) development projects. Management on the organisational level is represented by the line management functions and responsible for all strategic issues that go beyond the scope of development projects, e.g. definition of business goals, management of business and development processes, human resource management, and – last but not least – definition of product goals and initialisation of projects in order to develop the required products.

Management on project and organisational level is interrelated in many ways. There are two main reasons for the existence of both project and organisational level management:

1. In order to make the highly complex development of software products as flexible as possible while keeping control over schedule, budget, and quality, an efficient and effective project organisation is most adequate.
2. In order to (a) support the execution of current projects, and (b) establish a systematic learning cycle that goes beyond the project scope, organisational level management is needed. Note that systematic learning from project experience is *the* prerequisite for continuous improvement.

Figure 1 summarises the interrelationship between project and organisational level.

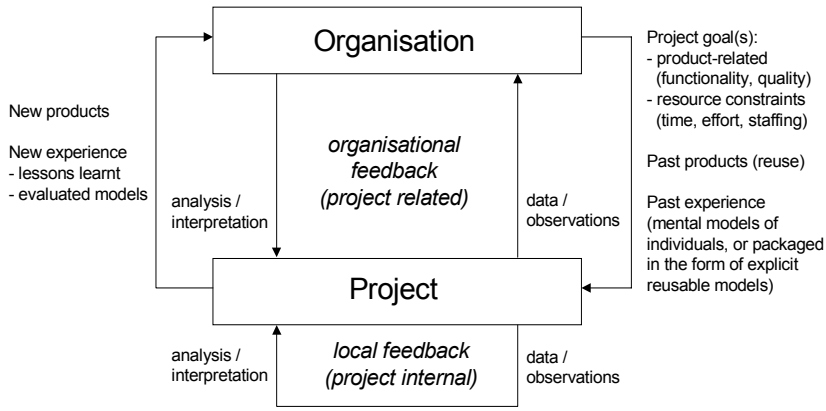


Figure 1: Interrelationship between project level and organisational level

1.1 The Problem of Complexity in Software Management

A major task of software management is decision making: choosing the right technologies, planning and supplying resources, controlling project, and finding the appropriate strategy to introduce innovative techniques, methods, and processes. But decision-making is difficult and risky because it is hard to reliably anticipate the impact of decisions on organisational and project performance. Particularly on organisational level, the evaluation of the impact of decisions, e.g. resulting in the introduction of new or change of existing processes or organisational structures, requires a tremendous investment in both time and money.

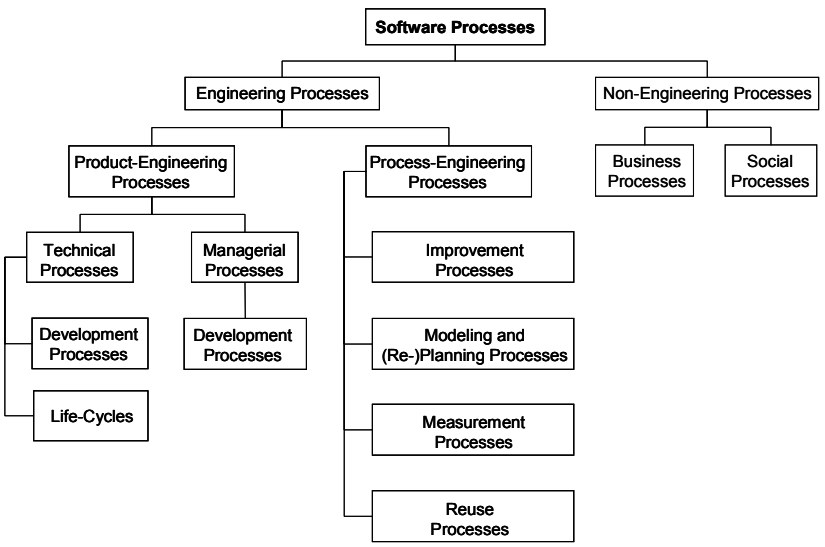


Figure 2: A tentative taxonomy of software processes (based on [RoV95])

The difficulties arise mainly for two reasons: huge complexity of software products, and – in order to be able to develop sophisticated software products – huge complexity of software processes. Both products and processes are the constituents of software engineering. Since products are the result of conducting processes in the scope of projects, the main sphere of impact for software managers is the process dimension. The diversity of software processes is large. As an illustration of process diversity, Figure 2 presents the tentative taxonomy of software processes suggested by Rombach and Verlage [RoV95].

1.1.1 Static and Dynamic Complexity

Senge distinguishes two major dimensions of complexity [Sen90]: static[1] complexity and dynamic complexity. Static complexity refers to the number of elements of a system, and the level of detail in which these elements are described. Dynamic complexity refers to behavioural aspects of a system and the difficulty with which causes and effects associated with the behaviour of the elements composing the system can be understood. Particularly systems that are essentially human-driven, as in software development, are characterised by high dynamic complexity. Large software products, like in telecommunication industry, typically are characterised by both static and dynamic complexity.

A standard approach to deal with complexity in the real world is to build models of reality and analyse them. A model is a set of propositions or equations describing in simplified form some aspects of the real world, which can be either the empirical world or the mental world that only exists in the head of the modeller. A model consists of a set of entities and associated attributes, described in terms of variables and relations defined on these.

There exist various types of models. They can be classified according to three criteria, i.e. explanatory power (black-box vs. white-box models), mode (static vs. dynamic models), and scale (quantitative vs. qualitative models).

Black-box models encapsulate relationships between variables, as, for example, when one seeks to build regression and other statistical models from a data set. They primarily reflect structure in the data. White-box models, on the other hand, must reflect entities of the real world, and relationships between the attributes of these entities. Hence, they convey insight into the structure of the process or product being modelled[2].

Static models are not able to capture and represent any sort of change over time of the real world. They are mere snapshots of the essential characteris-

[1] Senge uses the words "detail complexity". In this thesis, however, the naming "static complexity" is more appropriate as the distinction between static and dynamic complexity matches well with the distinction between static and dynamic models.

[2] For a discussion of the related topics the interested reader is referred to [KaM94].

tics of a part of the real world at a certain point in time. Dynamic models, in contrast, have the ability to express behaviour of elements or changes in the relationships between elements of the real world over time.

The decision whether a model is qualitative or quantitative depends on the scales[3] on which the variables contained in the model are defined. In the context of this thesis, models with variables that are defined on nominal scales only are called qualitative models. Models where all variables are defined on ordinal, interval, ratio, or absolute scales are called quantitative models[4]. Models that mix variables defined on nominal scales with variables defined on other scales are hybrid models. In the remainder of this thesis they will be included in the class of quantitative models.

It should be noted that quantitative models often use information that is contained in qualitative models. Similarly, dynamic models often use information that is contained in static models.

There are no restrictions in the possibility of combining criteria values. Thus, in total, eight model types can be distinguished, four static model types and four dynamic model types (cf. Figure 3).

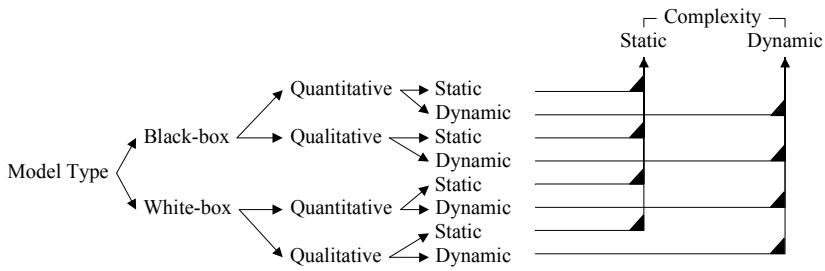


Figure 3: Types of models and their relation to complexity

Static models are adequate to capture static complexity, and dynamic models are adequate to capture dynamic complexity.

1.1.2 Established Modelling Approaches in Software Engineering

In the field of software engineering, generally, white-box modelling is associated with the discipline of process modelling, and black-box modelling is associated with the discipline of building analytical models that are expressed in terms of mathematical equations or in terms of logical or rule-based expressions.

[3] For a definition of scales the interested reader is referred to [FeP97].

[4] Mathematically speaking, also models where the variables are defined on ordinal scales have to be considered as qualitative. Under certain circumstances, however, models based on variables that are defined on ordinal scales can be considered as equivalent to quantitative models, i.e. models with variables that are defined at least on an interval scale [Spe80].

1.1.2.1 Process Modelling

A variety of methods for building white-box models have been proposed in the field of process modelling. The result of process modelling (PM) is a process model. A process model defines sub-sets of the engineering or non-engineering processes in software organisations (cf. Figure 2). A process model represents the entities of a process, and the relationships between the process entities. Figure 4 summarises essential entities of process models (cf. [Lon93]): activities, artefacts (that are consumed or produced by activities), tools (that are used by activities), roles (that perform activities), and actors (that perform one or more roles). Following the classification scheme proposed by Fenton and Pfleeger [FeP97] who distinguish three classes of entities, i.e., process, product, and resource entities, the entities presented in Figure 4 would classify as follows:

- Process: set of activities
- Product: set of artefacts
- Resources: set of tools, roles, actors

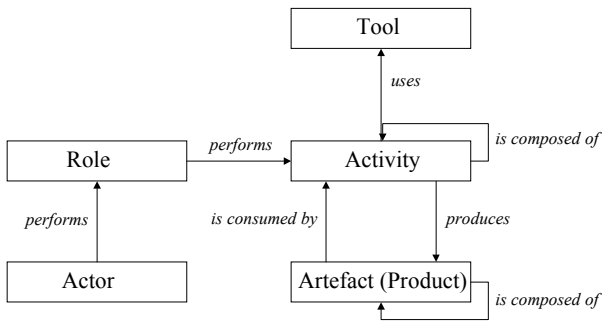


Figure 4: Basic entities of processes

The Software Engineering Institute (SEI) has defined a more comprehensive framework specifying the elements of a process model. Armitage and Kellner [ArK94] have published the main elements of this framework. An advanced framework that strives toward the integration of measurement-based analytic models (cf. Section 1.1.2.2) into process models has been proposed by Becker and Webby [BeW97]. Summary information about various approaches of process modelling (PM) can be found in [LiC91][ABG+92] [CKO92][FKN94][Pfa94a][BFL+95][FuW96].

Generally, two types of PM approaches can be distinguished [BHV97]: descriptive process modelling and prescriptive process modelling. Descriptive process modelling captures the current software development practices and organisational issues, i.e., the *actual process* and not the *official process* is represented (cf. [BFL+95]). Prescriptive process modelling specifies how software development practices and related organisational issues should be (official process). Figure 5 presents a tentative taxonomy of process models. The taxonomy is based on the work of McChesney [McC95].

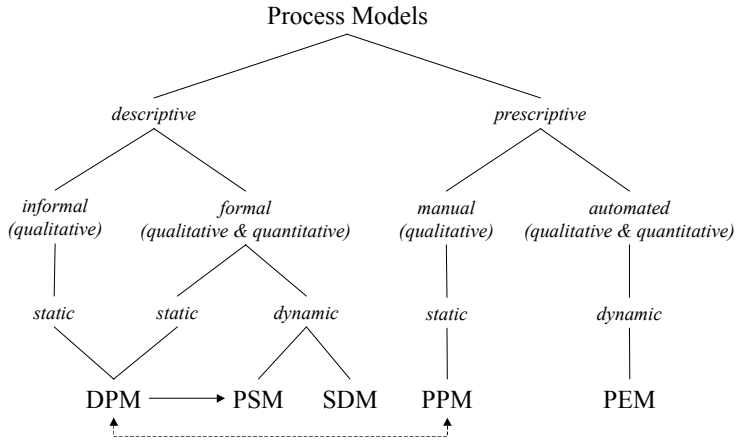


Figure 5: Taxonomy of process models (based on [McC95])

In the field of descriptive process modelling, many different modelling approaches have been proposed. McChesney divides the group of models resulting from descriptive process modelling according to their degree of formality [McC95]. Informal models are qualitative and static of nature. Formal models resulting from descriptive process modelling can be either qualitative or quantitative of nature. Both qualitative and quantitative formal models can be either static or dynamic. Static models resulting from descriptive process modelling are called descriptive process models (DPMs). Typical examples of informal DPMs are verbal process descriptions (e.g. phase models or life cycle models [Hum89][RoV95]). Examples of formal DPMs include graphical process representations with a defined semantic (e.g. statecharts, Petri-net based representations, activity networks, product flow charts, and actor dependency models [YuM94]), or formal process description languages (e.g. APPL/A [Ost87] and MVP-L [BLR+92][KLN+92][Ver98]). Dynamic models resulting from descriptive process modelling are either process simulation models (PSMs) or System Dynamics models (SDMs) [For61][For71][AbM91]. PSMs facilitate the simulation of process behaviour directly from the process description of a formal DPM. Examples of PSMs include models based on Petri-net based approaches like FUNSOFT [DeG94], SLANG [BFG92], and ProcessWeaver [Fer93], statechart-based approaches like Statemate [Har88] [HaP98][HuK89][KeH89][Raf96], and process description language approaches like MVPsim [Brö95][Brö97] and Prometheus [Vis94]. In contrast to PSMs, SDMs are not directly based on formal DPMs, although information contained in DPMs is a valuable input in building SDMs. In the context of software engineering, SDMs are developed based on structural and quantitative information of software processes and their organisational environment. The main purpose of SDMs is to capture feedback phenomena of software development systems[5] in a way that they can be analysed in order to improve process performance. Summary presentations dedicated to dynamic descriptive process modelling approaches can be found in [Brö95] [KMR99].

The group of models resulting from prescriptive process modelling can be divided in two categories: manual and automated models [McC95]. Manual models are qualitative and static of nature. They are called prescriptive process models (PPMs). PPMs can be based on (formal or informal) DPMs that have been altered in order to capture process improvements that are to be implemented. As soon as the process change is implemented and "lived" in the software organisation, the PPM turns back into a DPM. Typical examples of PPMs that are not derived from DPMs are software process standards such as IEEE 1074-1991[IEEE91], ISO 12207 [ISO95], or ISO 15504 (SPICE) [ISO98]. Automated models are dynamic models (either quantitative or qualitative) that aim at providing guidance during software development projects. They are called process enactment models (PEMs). PEMs are directed at aiding process actors by mechanically interpreting software process models, i.e. performing activities related to assistance, support, management and/or computer-assisted software production techniques. PEMs can be event-triggered, rule-based, or based on logic language, attribute grammars, automata (including finite state automata and Petri-nets), imperative programming languages, and abstract data types. Examples of PEMs include Adele [BEM91], ALF [OZG91], EPOS [KHL+94], GRAPPLE [HuL88], HFSP [Kat89], MARVEL [BaK91], MERLIN [EJS91], OIKOS [ACM90], SESAM [Lud+92][Sch93][DrL99], and SPADE [BFG92]. Summary presentation of several of the related modelling approaches can be found in [FKN94].

1.1.2.2 Measurement-Based Analytic Modelling

In software engineering, black-box models are measurement-based analytic models (MAMs) that are expressed in terms of mathematical equations (quantitative models) or in terms of logical expressions or rules (qualitative models). MAMs model attributes (e.g. size, number, quality, duration) and the relationships between attributes of real world entities. Quantitative and qualitative MAMs exist. Both quantitative and qualitative MAMs can be static or dynamic. A methodological framework for effective and efficient development of MAMs has been defined by the Goal / Question / Metric (GQM) approach [Bas93][BCR94b][vSB99].

Quantitative static MAMs are composed of one or more primitive quantitative models (QMs) of the form $y = f(x_1, \ldots, x_n)$. Primitive QMs are typically the result of statistical or other inductive analysis techniques (e.g., classification trees) applied to quantitative data. A QM defines the functional relation between one dependent variable (y) and one or more independent variables ($x_1, \ldots, x_n$). Briand et al. distinguish three types of primitive QMs, i.e., descriptive QMs, predictive QMs, and evaluation QMs [BDR96]. In the scope of industrial measurement programmes, usually sets of primitive QMs are defined. In order to be efficient and effective in defining sets of primitive

[5] A feedback system (or "closed" system) has a closed loop structure that brings results from past action of the system back to control future action [For71]. In this sense, software projects and software organisations can be interpreted as feedback systems.

QMs, it is recommended to proceed goal-oriented. In industrial environments, the GQM approach has successfully been applied for goal-oriented development of QMs [BDH+98][LSO+98][BDK+99][HPJ+99][vSB99]. Due to the top-down QM specification approach of GQM, clear hierarchical structures that define the relationships between primitive QMs are generated. Structured sets of primitive QMs are called analytic summary models (ASMs) [Kel88]. An example of an ASM for software project management is the well-known COCOMO model developed by Boehm et al. [Boe81] [BAB+00].

Qualitative static MAMs are typically the result of data mining techniques applied to qualitative data. An example of such a technique that has successfully been applied to software engineering problems is the rough-set analysis technique [Ruh96].

Quantitative dynamic MAMs are expressed in terms of first-order differential equations that are based on time series data [HKL87]. A first-order differential equation is a primitive dynamic model (DM). Systems of mutually interrelated first-order differential equations, i.e. higher order differential equations, can be used to describe and analyse management policies in software organisations [AbM91]. The related models are known under the name System Dynamics models (SDMs) [For61][For71]. A main feature of SDMs is the ability to capture feedback phenomena in real world systems.

Attempts to define methods for building qualitative SDMs have been made [OyK88][Rug94], but references to successful application in software engineering or managerial sciences could not be found in the literature.

1.1.3 Models that Capture Static Complexity

Taken from the set of models presented in Section 1.1.2, a sub-set of static models useful to capture static complexity has been summarised in Table 1.

			Static Complexity
Static Models	Black-box	Quantitative	primitive QMs, ASMs
		Qualitative	qualitative MAMs
	White-box	Quantitative	formal DPMs
		Qualitative	DPMs, PPMs

Table 1: Classification of static models

1.1.4 Models that Capture Dynamic Complexity

Taken from the set of models presented in Section 1.1.2, a sub-set of dynamic models useful to capture dynamic complexity has been summarised in Table 2.

			Dynamic Complexity
Dynamic Models	Black-box	Quantitative	primitive DMs, SDMs
		Qualitative	(qualitative SDMs)
	White-box	Quantitative	quantitative PSMs, SDMs, quantitative PEMs
		Qualitative	qualitative PSMs, (qualitative SDMs), qualitative PEMs

Table 2: Classification of dynamic models

1.1.5 Models are Tools for Learning

Referring to the statement made in Section 1.1.1 that a standard approach to deal with complexity in the real world is to build models of reality and analyse them, Figure 6 presents the generic process by which the use of models can become a tool of learning for software managers.

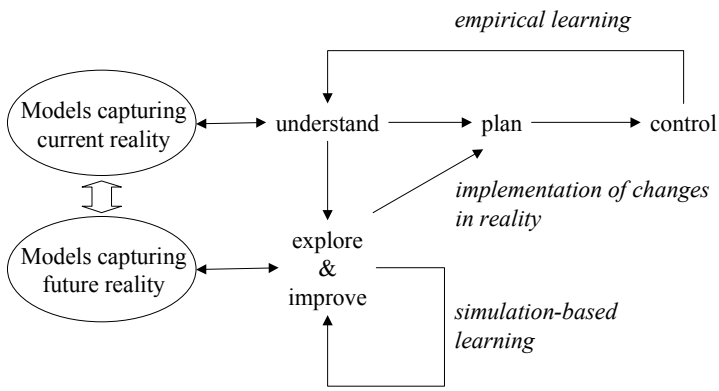


Figure 6: Generic process of model-based learning

The building and analysis of models that represent current reality helps software managers understand aspects of the real world that are in the focus of interest. The usage of models that represent current reality can help managers with many different kinds of planning tasks, e.g. project planning. As soon as tasks that have been planned with the help of models are performed, comparing expected outcomes with actual outcomes is a means to control the success of the activities involved in the task. In the case that the actual outcomes are different than the expected outcomes, a root cause analysis should be conducted. Such an analysis will produce new insights into the nature of the real world and will help improve the associated models. This kind of learning cycle can be called empirical learning.

There is, however, another potential dimension of model-based learning. If a manager had a model at hand that offers the possibility of representing a reality that is currently not in place, i.e. a virtual reality, then the manager

had the possibility to explore alternatives to the current reality, i.e. for the purpose of improvement. Typically, simulation models facilitate such kind of exploration, hence the name *simulation-based learning*. The transition from simulation-based learning to empirical learning happens as soon as management decides to implement one or the other alternative to the current reality, i.e. a process change. The model that represents the changed reality has become a model of current reality (given that the implementation of the process change was done correctly), i.e. it can be used for planning of tasks related to the real world and is subject to evaluation based on the results of controlling.

1.2 An Existing Framework for Model-Based Learning and Improvement

The systematic usage of models for learning and improvement in software organisations requires a methodological and organisational framework. An example of a successful framework for systematic learning and continuous improvement in software organisations is the *Quality Improvement* Paradigm (QIP) developed by Basili et al. at the Software Engineering Laboratory [Bas89]. The QIP strives to improve software development practice by making explicit the models that are underlying the software development activities, and using them as the key instrument (a) for defining and conducting development projects, and (b) to systematically learn from project experience on both project and organisational level. The organisational entity that facilitates experience-based learning is the so-called *Experience Factory*. An essential methodological element for empirical model building and analysis is goal-oriented measurement. The most advanced and successfully applied method for defining and conducting measurement programmes is the *Goal/Question/Metric* (GQM) method.

1.2.1 The Quality Improvement Paradigm (QIP)

The QIP[6] is a six-step procedure for structuring software development and improvement activities. It involves three overall phases: planning, execution, and evaluation. The planning phase consists of the explicit characterisation of the initial situation (QIP step 1), the identification of the goals to be achieved (QIP step 2), and the development of the implementation plan (QIP step 3). The implementation plan guides the systematic execution of the activities needed to achieve the goals (QIP step 4). The subsequent evaluation phase involves the analysis of the performed actions (QIP step 5) and the packaging of the experiences into new or improved reusable artefacts (QIP step 6).

[6] In a recent European research project, PROFES (Product Focused Process Improvement for Embedded Software) [BJK+98][HJO+98][BDK+99][PRO00], the QIP has been enhanced by integrating a method for explicit modelling of product-process dependencies. As a consequence, the six steps of QIP have been detailed into 12 steps. It should be noted, however, that the basic underlying principles of PROFES are the same as in the QIP.

QIP is based (a) on appropriate characterisation of the environment and providing a context for goal definition, and (b) on reuse of obtained experiences by packaging them in the form of structured knowledge represented by models. The improvement process established by QIP can be interpreted as an iterative process that implements two interrelated feedback cycles as illustrated in Figure 7:

1. The project feedback cycle (control cycle) provides feedback to the project during project execution in order to control project execution. This is done by analysing and interpreting qualitative and quantitative data collected during project execution. Its purpose is to optimise project performance, and to prevent and solve any kind of problems.
2. The corporate feedback cycle (capitalisation cycle) provides feedback to the organisation in order to facilitate organisational learning. It has a double purpose. Firstly, providing analytical information about project performance at project completion time by comparing the project data with the nominal range in the organisation and analysing concordance and discrepancy. Secondly, accumulating reusable experience in the form of software artefacts that are applicable to other projects and are, in general, improved based on project experience.

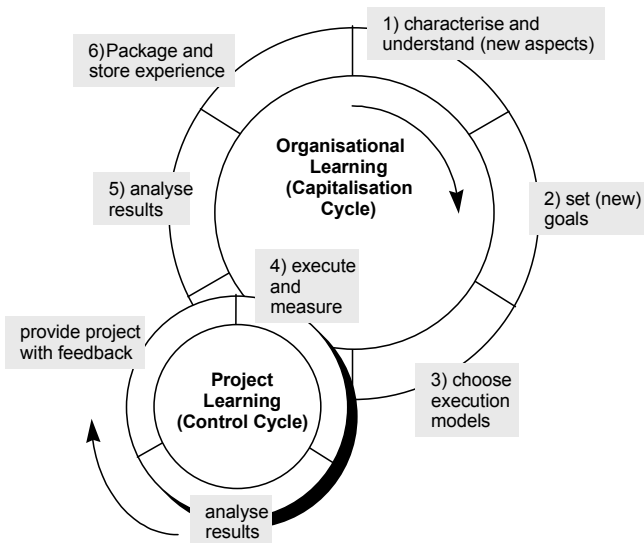


Figure 7: Capitalisation and control cycles in the QIP (adopted from [BaC95])

The capitalisation cycle is the key element in constituting organisational learning. Its steps can be characterised as follows [Bas92]:

1. Characterisation of the current project and its (organisational) environment.
2. Setting of the quantifiable goals for successful project performance and improvement.

3. Choosing of the appropriate process model and supporting methods and tools for the current project.
4. Execution of the processes, construct the products, collect and validate the prescribed data, and analyse it to provide real-time feedback for corrective action.
5. Analysis of the data in order to evaluate the current practices, determine problems, record findings, and make recommendations for improvements. The results of the analysis step can be both fed back to the current project in order to adjust/improve current project performance (project level), and to the packaging step (organisational level).
6. Packaging of the experience in the form of updated and refined models and other forms of structured knowledge gained from the current and prior projects and save it in an experience base for future projects.

In order to implement the QIP in a software organisation, an organisational entity that conducts all tasks that go beyond the responsibility of a project organisation is needed. In the context of the QIP, this entity has been given the name *Experience Factory* (EF) [BCR94a].

1.2.2 The Experience Factory

The Experience Factory is an organisational entity that is logically or physically separated from the project organisation (cf. Figure 8).

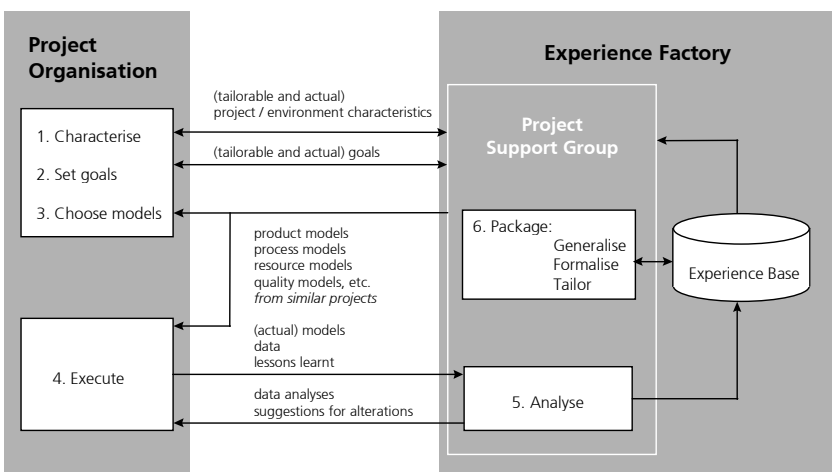


Figure 8: Organisational framework for systematic SPI (adopted from [Bas93], [BaC95])

Whereas the project organisation focuses on the development of a product, the priority of the Experience Factory is "to support project developments by analysing and synthesising all kinds of experience to various projects on demand. The Experience Factory packages experience by building informal,

formal or schematised, and productised models and measures of various software processes, products, and other forms of knowledge via people, documents, and automated support." [Bas89]

1.2.3 Goal-Oriented Measurement (GQM)

GQM (Goal/Question/Metric) is a well-known and widely used method for defining and executing goal-oriented measurement programmes. In the scope of this research it will be regarded as *the* unique reference method. GQM originated from the work lead by Basili at the University of Maryland and the NASA Software Engineering Laboratory in the 1980's [BaW84]. It has since been further formalised and developed into a practical methodology [BCR94b][BDR96][GHW95][vSB99].

1.2.3.1 GQM Principles

GQM represents a systematic approach to tailoring and integrating goals with: models of the software processes, software products, and with particular quality perspectives of interest. GQM focuses on the specific needs of the software project and of the development organisation. Measurement goals are defined on the basis of high-level corporate goals, and refined into metrics. In other words, GQM defines a certain goal, refines this goal into questions, and defines metrics that must provide the information to answer these questions. The GQM paradigm provides a method for top-down metric definition and bottom-up data interpretation.

The principles of GQM measurement are:

- A measurement programme must reflect interests of data providers and must be based on the knowledge of the people who are the real experts on the measurement goals. In this paper these are members of the software project team.
- Since the design of the measurement programme is based on the knowledge of the project team, only they can give valid interpretations of the collected data. Therefore, they are the only ones who are allowed to interpret measurement data.
- Due to the limited amount of time of project members, and their commitments to project planning, conflicts of interest may occur when all improvement efforts are also assigned to the project team. Therefore a separate team, a GQM team, should be created that facilitates the collection and analysis of measurement data by performing all operational activities not necessarily to be executed by the project team.

These principles imply that the members of the GQM team offer a service to the software project team by doing most of the technical work, related to setting up and performing the measurement programme. Essentially, during execution of the measurement programme, the GQM team provides a data

validation and analysis service, by organising 'feedback sessions' in which graphical measurement data is presented to the project teams.

1.2.3.2 GQM Process

The GQM process is divided into several stages. After the pre-study, the next stage is to identify a set of measurable quality goals. After the goals have been set, questions that define the goals are derived as completely as possible. The next step consists of specifying the metrics that need to be collected in order to answer the questions defined, and to track the conformance of products and processes to the defined measurable quality goals. Defined goals, questions and metrics are described in the GQM plan. The three layers (goals, questions, and metrics) of the GQM plan correspond to the following three levels:

- Conceptual level (Goal): The definition of the measurement goal specifies the object of measurement, the purpose of measurement, the quality model of interest, the role for whom the measurement results are of interest (viewpoint), and the environment in which the measurement programme takes place.
- Operational level (Question): A set of questions is used to define in a quantitative way the goal and to characterise the way the data will be interpreted. Questions try to characterise the object of measurement with respect to a selected quality issue and to describe either this quality issue from the selected point of view or the factors that may affect the quality issues.
- Quantitative level (Metric): A set of metrics - combined into a model - is associated with every question in order to answer the question in a quantitative way.[7]

The definition of the questions and metrics contained in a GQM plan is usually done with the help of so-called abstraction sheets. Basically, an abstraction sheet is a means for acquiring, structuring, and documenting all the relevant information provided by participants in the measurement programme. An abstraction sheet contains information about the measurement object and its associated attributes representing the quality focus (as specified by the measurement goal), and information about factors that have an impact on the quality focus (so-called variation factors). In addition, hypotheses about the performance of the quality focus attributes and the way in which the variation factors influence the performance of the quality focus attributes are documented. Based on this information, for each measurement goal, a set of questions, metrics, and models can be defined (for details see [BaW84] and [BDR96]).

[7] Cf. footnote 4 on page 4.

After the measurements have been specified, a mechanism for collecting measurement data is developed. This is described in the measurement plan and in the associated data collection forms. The data is then collected and validated during the software development project according to the measurement plan.

The collected data is analysed and discussed in feedback sessions. Feedback sessions are organised meetings involving members of the project team and the measurement team. It is an essential mechanism supporting analysis and interpretation of the measurement results. The main objective of feedback sessions is to discuss the preliminary findings and results of the measurement programme and derive interpretations by the project team from the data collected so far with the GQM experts.

After the end of the software development project all relevant information gathered during the project has to be packaged and stored for later retrieval and reuse. This is especially important for continuous learning and improvement.

Practical guidelines, examples and procedures for the GQM process in practice can be found in [vSB99].

1.2.4 Current White-Box and Black-Box Modelling in the QIP/EF/GQM Framework

State-of-the-art software engineering modelling strongly focuses on quantitative black-box and white-box modelling. There are three reasons for this:

- The advantage of quantitative models over qualitative models is that trade-off analyses can be conducted. An advantage that is particularly important for software managers where the finding and maintaining of an optimal equilibrium between cost, time, and quality (incl. functionality) is *the* main issue.
- The advantage of white-box models over black-box models is that changes in the model as a result of model analysis can directly transferred to the real world. This is due to the fact that the model structure is visible and represents entities in the real world.
- The advantage of black-box models is that mathematical analysis can be better applied. This is due to the fact that quantitative black-box models are usually represented in the form of mathematical equations.

Excluding qualitative software engineering models from the model classifications provided in Section 1.1.3 and Section 1.1.4, the remaining types of quantitative black-box and white-box models can be summarised as shown in Table 3. Quantitative black-box models that capture static complexity are either primitive QMs or ASMs. Both types of models are most effectively and efficiently developed by following the GQM method. Quantitative black-box models that capture dynamic complexity are either primitive DMs or SDMs. Both types of models are developed using standard dynamic modelling tech-

niques based on time series data analysis, or by following the System Dynamics method. Quantitative white-box models that capture static complexity are formal DPMs. Quantitative white-box models that capture dynamic complexity comprise quantitative PSMs, SDMs, or quantitative PEMs. Because PEMs aim at the automated guidance of software engineering processes and not at the analysis and improvement of existing processes and development practices, they are not in the focus of interest in the context of this thesis, and thus will not be further addressed.

		Complexity	
		Static	Dynamic
Model (static or dynamic, quantitative)	Black-box	primitive QMs, ASMs	primitive DMs, SDMs
	White-box	formal DPMs, PPMs	quantitative PSMs, SDMs, (quantitative PEMs)

Table 3: Quantitative models that capture complexity in software engineering

Figure 9 indicates the relationships between the various process model types, and the state-of-the-art modelling approach that is related to each models type: process modelling (PM) to develop DPMs, PPMs, and PSMs; goal-oriented measurement (GQM) to develop QMs and ASMs; System Dynamics to develop DMs and SDMs.

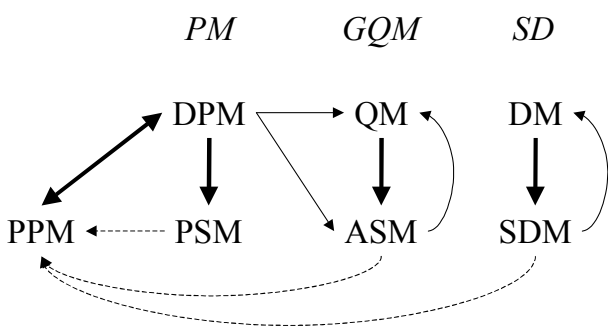


Figure 9: Relationships between models contained in a state-of-the-art experience base

The relations between the model types are characterised as follows:

- Bold solid arrows stand for the relation "is-part-of". For example, since PSMs are built upon formal DPMs, a DPM is usually part of the PSM. ASMs and SDMs are formed of QMs and DMs, respectively. The relationship between DPMs and PPMs is identity, i.e. a DPM becomes a PPM and a PPM becomes a DPM at certain phases of the improvement cycle, therefore the "is-part-of" relation is binary.
- Plain solid arcs stand for the relation "us-used-by" or "is-specified-by". For example, the qualitative information typically contained in a DPM, i.e. information about real world entities, attributes, and relationships between entities and their attributes, is used as an input for defining

goal-oriented measurement programmes[8], and thus influences the definition of QMs and ASMs. The development of QMs and DMs, on the other hand, is triggered by the development of ASMs and SDMs, respectively. This is mainly due to the fact that ASMs and SDMs are usually developed top-down, and thus specify to a certain degree the QMs and DMs upon which they are built.

- Dashed arcs stand for the relation "triggers-change-in". The type of model in which future improvements are documented is the PPM. The suggestions for the changes in a PPM result from analyses made with help of PSMs, ASMs, or SDMs.

Models that represent complete software development projects form an important class of models contained in a state-of-the-art experience base. All three types of models, ASMs, PSMs, and SDMs can be used to represent project complexity.

ASMs of software development projects capture mathematical relationships between selected input and output factors of the underlying development process (or process phases). Typical input factors include: estimated product size, project complexity, personnel capability, available tool support, resource and time constraints, etc. Typical output factors include: project duration and effort consumption. The focus of ASMs is on high-level relationships between independent and dependent model variables, facilitating point estimates that predict the impact of independent (input) variables on dependent (output) variables. The underlying process model is basically considered a "black-box", thus no detailed information about parameter changes during project performance is generated. Usually, the model equations contained in ASMs are derived through statistical analysis from empirical data of past projects. Prominent examples of ASMs that capture software project complexity are SLIM [Put78], COCOMO [Boe81][BAB+00], and Checkpoint [Jon91].

PSMs of software development projects capture the behavioural dimension of the underlying software process description. Due to their strict adherence to the underlying process model, Process Simulation Models (PSM) of software development projects provide operational guidance regarding the critical sequence of process steps, product flow, and information flow. In addition, these models often have the capability to check the integrity of the process. Prominent examples of PSMs that capture software project complexity are the Statemate-based models developed by Kellner et al. [HuK89][KeH89]. By using the Process Trade-off Analysis (PTA) method developed by Raffo, PSMs can be used to support the quantitative analysis of management decisions related to software process changes [Raf96].

[8] A well documented example of how descriptive process modelling is used for defining goal-oriented measurement programmes has been published by Bröckers et al. [BDT96].

SDMs of software development projects capture mathematical relationships between project, product, and process parameters, and, in addition, the time-dependent variation of several of these parameters. Compared to ASMs, System Dynamics Models (SDMs) of software development projects provide more details about the underlying process model. This is mainly due to the high flexibility of the System Dynamics Modelling approach, which allows for an adjustment of the level of granularity depending on the specific purpose for which the SDM has been developed. Compared to PSMs, SDMs of software development projects provide more flexibility in adequately capturing management decision rules, information feedback, delay, and non-linearity. The System Dynamics method can be classified as a continuous simulation modelling approach, although other paradigms such as discrete-event simulation can be emulated. Prominent examples of SDMs that capture software project complexity are the models developed by Abdel-Hamid et al. [AbM91], Lin et al. [Lin89][Lin93][LAS97], and Madachy [Mad94][Mad96].[9]

1.2.5 Disadvantages of Current White-Box and Black-Box Modelling

With regard to learning and improvement, the situation presented in Figure 9 can be summarised as follows:

1. Learning and improvement in software engineering can be based on three types of complex models: PSMs (white-box / dynamic), ASM (black-box / static), and SDM (white-box & black-box / dynamic).
2. For the development of models of each of the three types mature and well-documented modelling approaches have been defined.
3. The methodological integration between the related modelling approaches is lose (PM and GQM), or not existing (PM and SD, GQM and SD).

The fact that good methodological support exists for the development and application of either of the three model types upon which learning and improvement of software management can be based is definitely positive. Problems arise from the third finding, i.e. the lack of methodological integration between the three modelling approaches. In the current situation, in order to address the whole spectrum of complexity (static and dynamic) and use the whole spectrum of analysis capability (white-box and black-box models) managers have basically two options if they want to avoid the (effort consuming) modelling of all three types of models. Either they rely on PSMs and ASMs (option 1), or they rely on SDMs (option 2). Both options have disadvantages:

- Disadvantages of option 1: The relationship between model types PSM and ASM is not defined, therefore, it is not clear how analysis results

[9] More details are provided in Section 3.6.1.

addressing static complexity relate to analysis results addressing dynamic complexity. For example, if analyses based on process simulations indicate that the splitting of certain activities or concurrent performance of certain activities will shorten overall development time, how does this impact defect density of the artefacts produced by these activities? As long as the related (static) QMs are not integrated with the (dynamic) PSM, there is no possibility to answer this question.

- Disadvantages of option 2: Although SDMs are black-box and white-box at the same time, and much of the structural information contained in a SDM is based on static quantitative models, there exists no clearly defined method on how to integrate information contained in DPMs and QMs during the development of SDMs.

1.2.6 Strategies to Overcome Current Disadvantages with Model-Based Learning

In order to resolve the disadvantages related to options 1 and 2, the following possibilities exist:

- Strategy 1: The modelling approaches for PSMs and ASMs are integrated in a way that results from analyses based on one of the two model types can directly be interpreted in the context of the other model type. A main problem with this proposal is, however, the diversity in DPM – and thus PSM – approaches that would require a particular solution for each process modelling method.[10]
- Strategy 2: The modelling method System Dynamics is enhanced such that resulting SDMs smoothly integrate with QMs based on goal-oriented measurement (i.e., the current "static" GQM is enhanced toward "dynamic" GQM), and with relevant information contained in available DPMs (no matter according to which process modelling method they were built).

Raffo worked on strategy 1 by proposing the Process Trade-off Analysis (PTA) method to support the quantitative analysis of management decisions related to software process changes [Raf96]. The PTA method has been validated by using it in connection with Statemate-based PSMs in industrial environments. In principle, the PTA method is applicable to any type of PSM. So far, however, there is no integration of the analysis method available for any of the common process simulation modelling and analysis tools.

Strategy 2 is currently followed by Madachy who is about to integrate the COCOMO II model with a System Dynamics component (cf. chapter on "Dynamic COCOMO" in [BAB+00], p. 207 ff). This extension of a generic project ASM with dynamic elements of a SDM will certainly become a powerful planning tool for software project managers. The integration of a par-

[10] Although there exists a proposal of a meta-modelling process for the development of DPMs, the Elicit method developed by Madhavji et al. [MHH+94], there exists no generally accepted method for the development of comprehensive DPMs in industrial environments (cf. [Ver98], p. 26).

ticular ASM with elements of a SDM, however, does not deliver comprehensive guidance on how to conduct integrated development of static ASMs and dynamic SDMs in general.

1.3 Objective of Thesis

The objective of the research conducted in the scope of this thesis has been to integrate System Dynamics modelling with goal-oriented measurement (GQM) and descriptive process modelling in order to provide software managers with a comprehensive approach to model-based learning, which combines empirical learning with simulation-based learning.

Since empirical learning is state-of-the-art in software engineering for almost two decades and the increasing number of related publications is showing that it is becoming state-of-the-practice in industrial software organisations, the focus of the research will be put on simulation-based learning. It is expected that simulation-based learning will become state-of-the-practice in software engineering – as in any other engineering discipline – as soon as System Dynamics modelling has been fully integrated with complementary modelling practices currently used in the QIP/EF framework, i.e. descriptive process modelling (PM), and goal-oriented measurement (GQM).

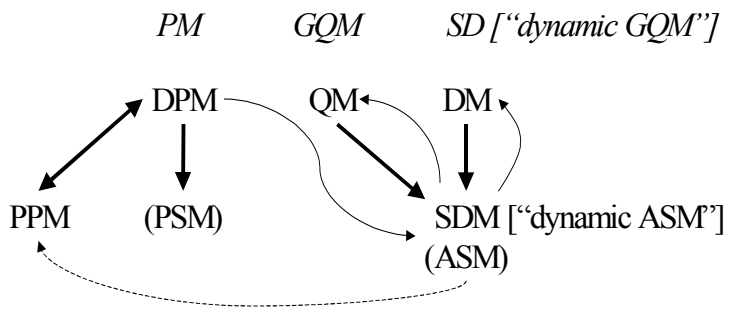


Figure 10: Relationships between models after integration of SD with PM and GQM

How the relationships between models in an experience base will be after integration of SD modelling with PM and GQM is shown in Figure 10. Compared to Figure 9, the QMs are used by SDMs for defining static functional relations, in the same way as DMs are used for defining dynamic functional relations. The elements of both QMs and DMs are specified by the SDM, which in turn takes relevant qualitative information from DPMs. ASMs, i.e. the structured collection of QMs do not longer exist stand-alone but are fully integrated with a SDM, which become "dynamic ASMs"[11]. PSMs become absolete because all relevant qualitative white-box information from the related DPMs is integrated with the SDM. Process simulations on any kind of

[11] This integration of ASMs and SDMs adds a new shade of meaning to the label "Analytic Structural Model" under which Raffo subsumes the class of SDMs [Raf96].

granularity can be run by the SDM[12]. In this setting, improvement suggestions, i.e. changes in the PPM, are triggered exclusively based on analysis of SDM simulation results.

The advantage of the integration of SD modelling with descriptive PM and GQM, i.e. an extension of GQM toward a "dynamic GQM", has several advantages:

1. Consistent and transferable interpretation of results from model analyses.
2. High degree of reuse of model entities and modelling activities.
3. Reduction of modelling and analysis effort due to reduction of modelling paradigms to the essential.

1.4 Research Approach

In the course of the research work related to the thesis, the following research methods have been applied: literature survey, action research (explorative case study), industrial case study, and controlled experiment. The flow of work is summarised in Figure 11.

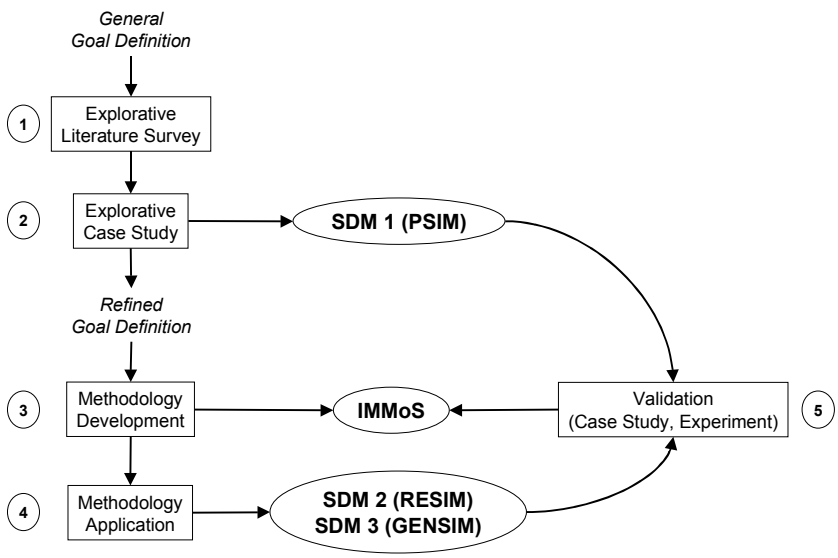


Figure 11: Work flow of research project

Literature survey (step 1) and action research (step 2) have been used to identify the needs of managers in software organisations with regards to

[12] Note that advanced SDM tools provide functionality that even allows for integration of descrete-event simulation with SDMs.

building and using SDMs in order to cope with the dynamic complexity of software development. Based on the literature surveys, information about the state-of-practice and state-of-the-art in SD modelling was collected. Action research was applied in order to gain insights into the actual impact of using SDMs in software organisations, and into the strengths and weaknesses of the currently available guidance for building and using SDMs (baselining) [Arg83]. For this purpose a SD modelling project (PSIM) was conducted in an industrial environment (a software division in the telecommunication industry).

Based on the results of the modelling project, the state-of-the-art in SD modelling was enhanced (step 3) by developing comprehensive guidance for System Dynamics model building, i.e. the IMMoS framework (Integrated Modelling, Measurement and Simulation). The elements of the IMMoS framework were used to build two additional SDMs (step 4), i.e. the RESIM model for an industrial environment (a software division in the automotive industry), and the GENSIM model for academia (University of Kaiserslautern).

Two industrial case studies[13] and a controlled experiment in a university setting were conducted in order to validate (step 5) the proposed IMMoS framework with regards to efficiency and effectiveness.

1.5 Research Hypotheses

For the validation of the proposed IMMoS framework two research hypotheses were investigated:

- Research Hypothesis 1: SDM development with IMMoS is at least as effective as SDM development without IMMoS.
- Research Hypothesis 2: SDM development with IMMoS is more efficient than SDM development without IMMoS.

Effectiveness is measured as the degree to which a developed SDM fulfils its purpose from the point of view of the model user. Efficiency is measured in terms of effort and time needed for developing a SDM.

1.6 Contributions of Thesis

The main contribution of this PhD research lies in the design, application and validation of a framework for Integrated Measurement, Modelling, and Simulation (IMMoS). The IMMoS framework will support managers in coping with the dynamic complexity of software development by providing guidance in developing and using quantitative simulation models as a source for systematic learning and improvement.

[13] Guidance on how to conduct case study research can be found in [Yin94].

The main achievements resulting from the development of the IMMoS framework can be summarised as follows:

1. Theoretical work:
   - Detailed description of a process for SDM development with:
     - definition of roles, responsibilities, activities, and work products,
     - support in specifying the simulation modelling goal, and
     - provision of guidelines, checklists, templates, and other materials.
   - Integration of SD modelling with the GQM modelling method.
   - Systematic use of information contained in static white-box and black-box models, i.e. DPMs and QMs, for SDM development.
2. Practical work:
   - Development of three SDMs for different modelling goals.
   - Integration of one SDM into a web-based training module for students on the topic of software project management.
   - Implementation of the IMMoS process model with SPEARMINT [BHK+99]. Based on this implementation, a web-based Electronic Process Guide (EPG) for IMMoS can be automatically generated.
3. Empirical work:
   - The effectiveness and efficiency of the IMMoS framework is supported with empirical evidence from two case studies and one controlled experiment.

1.7 Structure of Thesis

The results of the research are presented in four parts (cf. Figure 12). Part I relates to research step 1 (cf. Figure 11 in Section 1.4). It summarises the foundation of the research, i.e. the essence of model-based learning for managers (Section 2), and a brief description of the System Dynamics method (Section 3). Part II relates to research step 2. It presents the results of the action research phase. In particular, this includes a summary of the lessons learned from the explorative case study (PSIM project, cf. Section 4), which also served as a baseline for evaluating the effectiveness and efficiency of IMMoS. Part III relates to research step 3. In Part III the innovations of the research work are presented, i.e. the IMMoS framework and its components (Section 5 to Section 8). Part IV relates to research steps 4 and 5. It summarises the results of the empirical research, i.e. the validation of the effectiveness and efficiency of IMMoS (Section 10 to Section 14). This includes the descriptions of the RESIM and GENSIM projects, which both used the IMMoS approach for SDM development. The thesis concludes with a summary of contributions and achievements and an outlook to future work (Section 15).

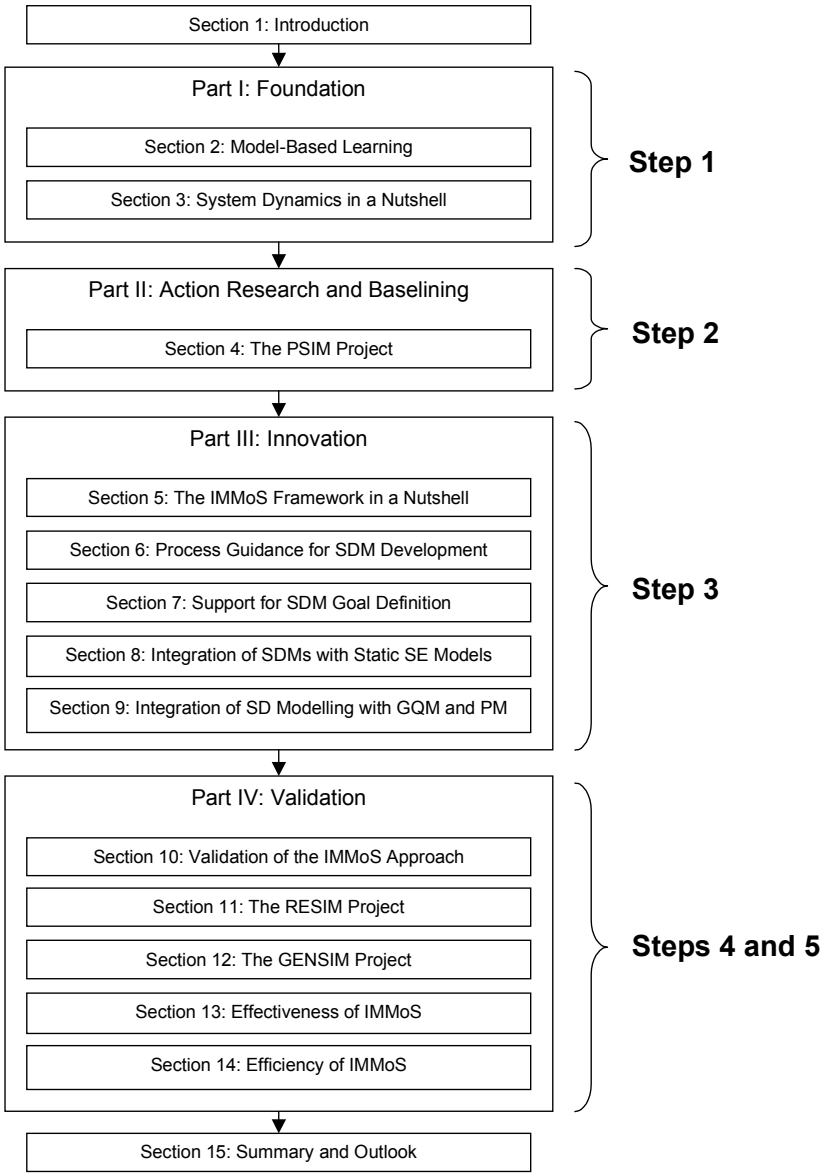


Figure 12: Structure of thesis

Part I: Foundation

2 Model-Based Learning

The QIP/EF framework can be seen as a powerful facilitator of systematic learning and continuous improvement. The success of the QIP/EF approach is essentially based on the prominent role of explicit models, which is a consequence of strictly following the inductive approach of the "scientific method" [Bas92], either in its evolutionary improvement oriented manifestation ("engineering method") or in its revolutionary improvement oriented manifestation ("empirical method"). In the context of software development, four important uses of models can be distinguished. The combination of these uses facilitates learning and improvement. The four uses are:

1. Capturing and packaging experience from past projects.
2. Proposing improvements upon current ways of conducting projects (evolutionary approach) or proposing new ways of conducting projects independent from past experience (revolutionary approach).
3. Support for planning of current projects.
4. Support for execution of current projects (control).

Note that the first two uses can be associated with the organisational level of management, whereas the last two uses are associated with project management.

In order to understand better how the use of explicit models facilitates learning and, as a consequence, continuous improvement, a closer look at the general structure of learning processes is helpful.

2.1 Learning is a Feedback Process

The feedback-loop character of learning has been recognised for long time, since around the turn of the 19th to the 20th century, when John Dewey described learning as an iterative cycle of invention, observation, reflection, and action ([Sch92], referenced according to [Ste94], p. 293). During the last century, the notion of learning as an explicit feedback process has found its way into many areas of the social and management sciences.

The single feedback loop shown in Figure 13 describes the most basic type of learning. The loop is a classical negative feedback whereby decision makers compare quantitative and qualitative information about the state of the real world to various goals, perceive discrepancies between desired and actual states, and take actions that (they believe will) cause the real world to move toward the desired state.

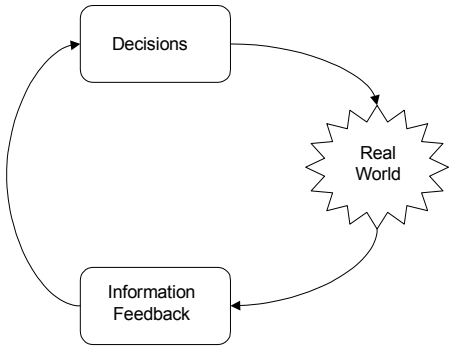


Figure 13: Learning is a feedback process

2.1.1 Single-Loop Learning

The feedback loop shown in Figure 13, however, obscures an important aspect of the learning process. Information feedback about the real world is not the only input used by decision-makers. Their decisions are the result of applying a decision rule or policy to information about the world as they perceive it [For61]. The decision rules are themselves conditioned by institutional processes, organisational strategies, and cultural norms. These are in turn governed by the mental models that a decision-maker has about the real world (cf. Figure 14).

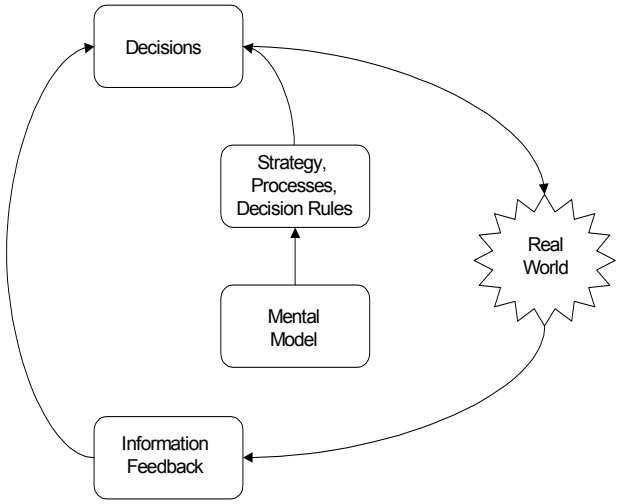


Figure 14: Single-loop learning

As long as the mental models of the decision-makers remain unchanged, the feedback loop in Figure 14 represents *single-loop* learning [ArS78][Arg85], or *adaptive* learning [Mil95]. Single-loop learning is a process where deci-

sion-makers learn to reach their current goals in the context of their existing mental models. It is focused on the operational level of management, based on detecting and correcting errors, competencies and routines. Single-loop learning does not result in change to the existing mental models, i.e. the understanding of the causal structure of the system in which the decision-makers act.

2.1.2 Double-Loop Learning

Since software development is a highly complex task that takes place in an ever faster changing world of development methods, techniques, and tools, it becomes necessary for decision-makers to become active players who adjust their mental models to reality, and use them as a tool for reflection and generation of improvement suggestions. This way of thinking, which involves continuously new articulations of the decision-makers' understanding, or reframing of a situation, and which leads to new goals and new decision rules – not just new decisions – is called *double-loop* learning [ArS78][Arg85]. When successful, such learning replaces the reductionist, partial, narrow, short-term perception of the world with a holistic, broad, long-term, and dynamic view that constantly redesigns decision-makers' policies and thus the organisations they live in. Figure 15 illustrates how the single-loop feedback is complemented by a second feedback loop that connects mental models with the information received from observing the real-world via interpretation and reflection, and thus makes it accessible to adaptation and change.

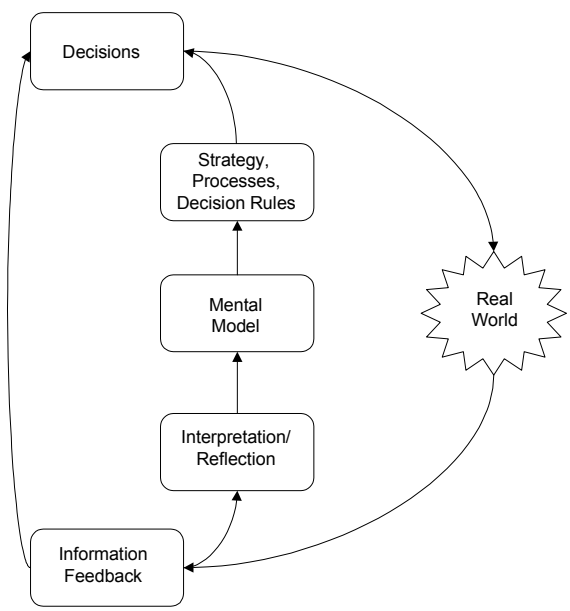


Figure 15: Double-loop learning

2.2 Organisational Learning

In order to capture the difference between individual and organisational learning, an extension of Figure 15 is needed. Since each of the decision-makers in a software organisation has their own (implicit) mental model of the real world, there exists a multiplicity of mental models in the organisation. This is adequately represented in Figure 16 by the duplication of mental models.

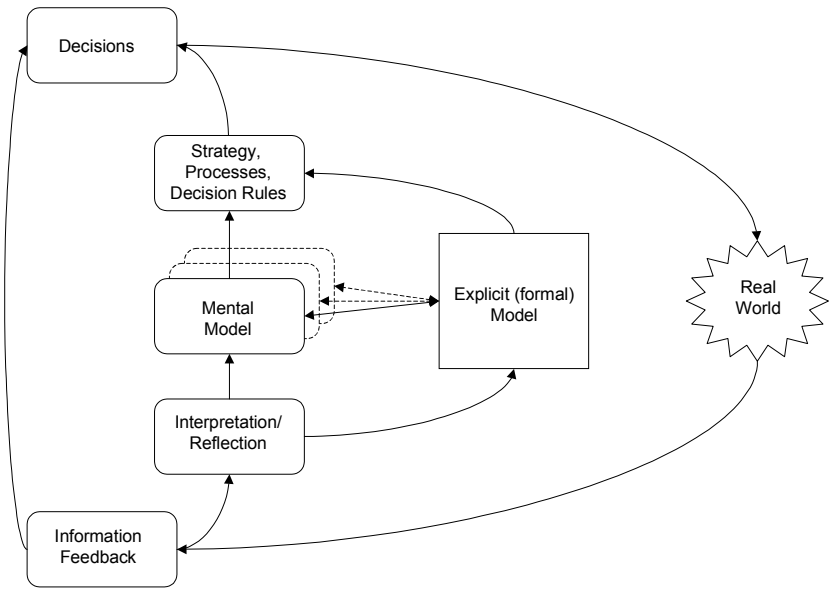


Figure 16: Organisational learning with models

Structure, processes and goals of the organisation, as well as the influences from the environment on the organisation will be perceived differently by the actors in the organisation, and thus the expectations about the effects of own decisions or decisions made by others: "Each member of the organization constructs his or her representation, or image, of the theory-in-use of the whole. That picture is always incomplete (…) Organization is an artifact of individual ways of representing organization." [AgS78]. In order for an organisation to be able to benefit from the double-loop learning on an individual basis it is important to integrate and streamline the individual mental-models, and use them establish double-loop learning on the organisation level. The establishment of this complex process is one of the major tasks of senior management.[14] The many decision-makers that concurrently act

[14] In order to explain the complex "process of organisational knowledge creation", Nonaka et al. developed a theory of organisational learning that is based on a process of transformations between explicit and implicit (tacit) knowledge [NoT95]. This process consists of the phases socialisation, externalisation, internalisation, and combination, which are continuously iterated. A comprehensive discussion of existing theories on organisational learning and their conceptual foundations has been provided by Wiegand [Wie96].

upon and reflect about software development in the organisation have to be considered, and all the important aspects that cause success or failure have to be accounted for [Mil95].

One important tool to establish organisational learning is to integrate the individual mental models of decision-makers into explicit formal models that are then used to (a) reflect upon the real world and (b) support managerial tasks such as decision-making, planning, and control.

2.3 Barriers to Learning in Software Organisations

Usually, the models contained in the experience base of a software organisation are static, i.e. they do not facilitate the analysis of behavioural aspects of software development, e.g. the variation of process parameters or product attributes over time during project execution. Static models, however, do only have limited value for helping managers in software organisations overcome two major difficulties [Sen90]:

- High dynamic complexity of software projects (which cannot be captured by static models)
- Presence of information feedback (triggering unexpected side-effects of management decisions), delay (of management decisions), and non-linearity of cause-effect relationships.

Based on a comprehensive study analysing potential barriers to systematic learning, Morecroft identified many ways in which the links of double-loop learning shown in Figure 15 can fail. Morecroft identified the following obstacles [Ste94]:

- Real World:
  - Unknown structure
  - Dynamic complexity
  - Time delays
  - Inability to conduct controlled experiments
- Information Feedback:
  - Selective perception
  - Missing feedback
  - Delay
  - Bias, distortion, error
  - Ambiguity
- Decisions:
  - Implementation failure
  - Game playing
  - Inconsistency
  - Performance is goal
- Interpretation/Reflection:
  - Misperceptions of feedback
  - Judgmental biases

    - Defensive routines
- Mental Models:
    - Unscientific reasoning
- Strategy, Processes, Decision Rules:
    - Inability to infer dynamics from cognitive maps

2.3.1 Dynamic Complexity

Despite the fact that much of the literature in psychology and other fields suggests learning proceeds via the simple feedback loops described in Figure 14, the real world is not so simple. In particular, learning cannot be considered as first-order negative feedback producing stable convergence to an equilibrium or stable output. Human-based processes, such as software development, involving more than one person are much more complex with high levels of dynamic complexity. For example, in a software organisation, the decisions and related actions of one agent form but one of many feedback loops that operate in the organisational system. These loops may reflect both anticipated and unanticipated side effects of the decision makers' actions; there may be positive as well as negative feedback loops, and the loops will contain many system entities and many non-linearities in the relationships between these entities.

Also, time delays between taking a decision and its effects on the state of the system under consideration are common and particularly problematic. Most obviously, delays reduce the number of times one can cycle around the learning loop, slowing the ability to accumulate experience, test hypotheses, and improve.

Dynamic complexity not only slows the learning loop but also reduces the learning gained on each cycle. In many cases, controlled experiments are prohibitively costly or risky. Often, it is simply impossible to conduct controlled experiments. Complex systems are in disequilibrium and evolve. Many actions yield irreversible consequences. The past cannot be compared well to current circumstances. The existence of multiple interacting feedback loops means that it is difficult to hold other aspects of the system constant to isolate the effect of the variable of interest. As a result, many variables simultaneously change, confounding the interpretation of changes in system behaviour and reducing the effectiveness of each cycle around the learning loop.

Delays also create instability in dynamic systems. Adding time delays to the negative feedback loops, which typically are associated to control tasks of managers, increase the tendency for the system to oscillate. As a result of too long time delays, decision makers often continue to correct apparent discrepancies between the desired and actual state of the system even after sufficient corrective actions have been taken to restore the system to equilibrium, leading to overshoot and oscillation[15]. In software organisations, oscillation and instability reduce the ability of decision-makers to control for con-

founding variables and discern cause and effect further slowing the rate of learning.

2.3.2 Limited Information

All human beings experience the real world through filters. No one knows the current number of defects in a software product, the current productivity rate, or the actual degree of completeness of a development project. Instead, managers receive estimates of these data, based on sampled, averaged, and delayed measurements. The act of measurement introduces distortions, delays, biases, errors, and other imperfections, some known, others unknown and unknowable. Above all, measurement is an act of selection. Measurement-based information systems in place select but a tiny fraction of possible experience.

Of course, the information systems governing the feedback that decision-makers in a software organisation receive and its characteristics can change as learning proceeds. The information systems itself are part of the feedback structure of the systems under consideration. Through the mental models of the agents, constructs such as "productivity" or "quality" emerge, metrics for these constructs are defined, and measurement programmes to collect data and report them are designed and implemented. These then condition the perceptions that agents form. Changes in the mental models of the agents are constrained by what they previously chose to define, measure, and attend to. As Sterman puts it [Ste94]: "Seeing is believing, and believing is seeing".[16] Even though sometimes this sort of positive feedback assists learning by sharpening the ability of decision-makers to perceive interesting features of the system, often, however, the mutual feedback of expectations and perceptions limits learning by blinding decision-makers to the anomalies that might challenge their mental models.

2.3.3 Confounding Variables and Ambiguity

To learn, managers must use the limited and imperfect feedback available to them to understand the effects of their own decisions, so they can adjust their decisions to align the state of the system (e.g. the software organisation, or a software project) with their goals (single-loop learning), and so they can revise their mental models (and the associated formal models) and redesign the system itself (double-loop learning). Yet much of the feedback received from the real world is ambiguous. Ambiguity arises because changes in the state of the system resulting from decisions are confounded with simultaneous changes in a host of other variables, both exogenous

[15] A trivial but widely known example of this kind of unwanted oscillation is the adjustment of the temperature of a shower that responds to the adjustment of the temperature with too long delay.

[16] Sterman provides a brief summary of the philosophical discussion about the critical role of beliefs in conditioning perceptions [Ste85].

(external to the observed system) or endogenous (internal to the observed system). The number of variables that might affect the system vastly overwhelms the data available to rule out alternative theories and competing interpretations. This identification problem plagues both qualitative and quantitative approaches.

In the qualitative realm, ambiguity arises from the ability of language to support multiple meanings. For example, in descriptive process modelling, one of the key tasks during knowledge elicitation for modelling the actual processes in place is to clarify terminology, in order to make sure that information collected from different individuals refers to the same or different entities of the process [BeB00].

In the quantitative realm, the main problem is to uniquely identify the structure and parameters of a system from its observed behaviour. As soon as the system has been identified, a vast set of statistical analysis methods can be applied to determine the functional relations between system parameters. In principle, elegant and sophisticated theory exists to delimit the conditions in which one can identify a system. In practice, however, data are often too scarce and the alternative plausible alternative specifications too numerous for statistical methods to discriminate among competing theories. The same data often support divergent models equally well, and conclusions based on such models are not robust. Research of Briand et al. in the field of software cost and quality modelling provides examples of these difficulties [BEF+98][BEW99].

2.3.4 Misperceptions of Feedback

Effective management is difficult in a world of high dynamic complexity. Decisions may create unanticipated side effects and delayed consequences. Attempts to stabilise a system may actually destabilise it. Decisions by one agent in the system may provoke reactions by other agents seeking to restore the balance upset by the decisions of the first agent. Decisions may move the system into a new regime of behaviour, where unexpected and unfamiliar dynamics arise because the dominant feedback loops have changed. Forrester calls such phenomena the "counterintuitive behavior of social systems" [For71]. It often leads to "policy resistance", the tendency for interventions to be delayed, diluted, or defeated by the response of the system to the intervention itself.

Sterman reports a series of studies that confirm these observations yielding the following statement [Ste94]: "These studies let me suggest that the observed dysfunction in dynamically complex settings arises from misperceptions of feedback. I argued that the mental models people use to guide their decisions are dynamically deficient. Specifically, people generally adopt an event-based, open-loop view of causality, ignore feedback processes, fail to appreciate time delays between and response and in the reporting of information, (…) and are insensitive to nonlinearities that may alter the strengths

of different feedback loops as a system evolves."[17] Sterman concludes that the misperceptions of feedback result from two basic and related deficiencies in the mental models of complexity: "First, our cognitive maps of the causal structure of systems are vastly simplified compared to the complexity of the systems themselves. Second, we are unable to infer correctly the dynamics of all but the simplest causal maps. Both are direct consequences of bounded rationality [Sim79] [Sim82]; that is, the many limitations of attention, memory, recall, information processing, and time that constrain human decision making."

2.3.5 Flawed Cognitive Maps of Causal Relations

Causal relations are a central feature of mental models. Individuals create, update, and maintain cognitive maps of causal connections among attributes of entities and actors. Often, however, individuals tend to think in mono-causal dependencies and have difficulty to conceptualise systems with side effects and multiple causal pathways. Studies have shown that the occurrence of feedback in mental models is practically not existent, thus reducing mental models to open-loop, event-level representations with intuitive decision trees relating possible actions to probable consequences [Axe76] [Dör80]. Often individuals focus on single causes that cease the search for explanations of a particular event when a sufficient cause is found. Sterman concludes that the heuristics managers and decision-makers use to judge causal relations lead systematically to cognitive maps that ignore feedback, multiple relationships, non-linearity, time delay, and the other elements of dynamic complexity [Ste94].

Senge pointed out that "systems cause their own crises, not external forces or individual's mistakes" [Sen90]. Individuals, however, have a strong tendency to attribute the behaviour of other actors in the system to dispositional rather than situational factors. But in complex systems, the same policy (decision rule) can lead to very different behaviour (decisions) as the state of the system changes. The attribution of system behaviour to individuals and special circumstances rather than to system structure systematically diverts the attention of managers and decision-makers from the high-leverage points where redesign of the system or governing policy can have significant, sustained, beneficial effect on system performance. Sterman summarises this as follows: "When we attribute [system] behavior to people rather than to system structure, the focus of management becomes the search for extraordinary people to do the job rather than designing the job so that ordinary people can do it."

[17] Compare also Senge's discussion of this topic [Sen90].

2.3.6 Erroneous Inferences about Dynamics

Even if the causal maps of causal system structure were perfect, learning, especially double-loop learning would still be difficult. In order to use a mental model to design a new strategy, process or organisation, a manager must make inferences about the consequences of decision rules that have never been tried and for which data is not available. To do so requires intuitive solution of higher-order non-linear differential equations, a task far exceeding human cognitive capabilities in all but the simplest systems [For71] [Sim82].

Individuals cannot simulate mentally even the simplest possible feedback system, the first-order linear positive feedback loop, i.e. the differential equation $dx / dt = gx$ yields pure exponential *growth* $x = x_b \cdot exp\ (gt)$. Even though such positive feedback processes are commonplace, individuals significantly underestimate exponential growth, tending to extrapolate linearly rather than exponentially. Sterman concludes that "bounded rationality simultaneously constrains the complexity of our cognitive maps and our ability to use them to anticipate the System Dynamics. Schemata where the world is seen as a sequence of events and where feedback, nonlinearity, time delays, and multiple consequences are lacking lead to poor performance in settings where these elements of dynamic complexity are prevalent. Dysfunction in complex systems can arise from the misperception of the feedback *structure* of the environment. But schemata that do account for complexity cannot be used reliably to understand the dynamics. Dysfunction in complex systems can arise from faulty mental simulation – the misperception of feedback *dynamics*. These two different bounds on rationality must both be overcome for effective learning to occur. Perfect maps without a simulation capability yield little insight; a calculus for reliable inferences about dynamics yields systematically erroneous results when applied to simplistic maps".

2.3.7 Unscientific Reasoning; Judgmental Errors and Biases

Most individuals are poor intuitive scientists, generally failing to reason in accordance with the principles of scientific method. Sterman lists some typical shortcomings [Ste94]:

- Indivduals do not generate sufficient alternative explanations or consider enough rival hypotheses.
- Individuals do not adequately control for confounding variables when they explore a new situation.
- Individuals' judgements are strongly affected by the frame in which the information is presented, even when the objective information is unchanged.
- Individuals suffer from overconfidence in their judgements (underestimating uncertainty), wishful thinking (assessing desired outcomes as

more likely than undesired outcomes), and the illusion of control (believing one can predict or influence the outcome of random events).

Hogarth discusses 30 different biases and errors documented in decision-making research [Hog87]. Sterman claims that "among the failures of scientific reasoning most inimical to learning is the tendency to seek evidence consistent with current beliefs rather than potential confirmation" [Ste94].[18] Successful learning, however, can only take place when issues are examined from multiple perspectives and the boundaries of mental models are expanded to consider the long-term consequences and side effects of any decision taken. This is particularly important for decisions taken on strategic management level.

2.3.8 Defensive Routines and Interpersonal Impediments to Learning

Organisational learning can be thwarted even if the system under consideration provides excellent information feedback and the decision-makers reason well as individuals. Individuals rely on their mental models to interpret the language and acts of others, construct meaning, and infer motives. Argyris and others document the defensive routines and cultural assumptions individuals rely on, often unknowingly, to interact with others [Arg85]. Defensive routines are used to save face, assert dominance over others, make untested inference seem like facts, and advocate own positions while appearing to be neutral, etc. defensive routines ensure that the mental models of individuals remain hidden, ill formed, and ambiguous.

2.3.9 Implementation Failure

In the real world, decisions are often implemented imperfectly. This can defeat the learning process because the management team evaluating the outcome of the (imperfectly) implemented decision may not be aware of the ways in which the decisions they thought they were implementing were distorted.

2.4 Simulation-Based Learning

In order to facilitate successful learning in such complex systems as software organisations and software projects, all impediments to learning need to be addressed adequately. Sterman claims that effective learning involves continuous experimentation in both real worlds and virtual worlds, with feedback from both, thus developing the mental models, the formal models, and decisions to be applied in the real world [Ste94]. Virtual worlds are formal mod-

[18] For a detailed discussion of the role of systematic falsification as a scientific method cf. Popper [Pop68].

els, or microworlds, in which decision-makers can simulate and thus test the expected behaviour of planned changes to the real world.

Virtual worlds have several advantages. First, they provide low-cost laboratories for learning. The virtual world allows time and space to be compressed or dilated. Actions can be repeated under the same or different conditions. Actions can be stopped at any time in order to reflect. Decisions that are risky or costly in the real world can be taken to the virtual world. Thus controlled experimentation becomes possible, and the time delays in the loop through the real world are dramatically reduced. In the virtual world, it is possible to push the system into extreme conditions, often revealing more insights about its structure than incremental adjustments to the current practice in the scope of limited pilot projects of the real world.

Table 4 summarises the advantages that virtual worlds (simulation models) can have on the learning processes of managers and decision makers, as compared to the barriers to learning without formal models.

	Real World	**Virtual World (Simulation Model)**
	Unknown structure	Known structure
	Dynamic complexity	Variable level of complexity
	Time delays	
	Inability to conduct cotrolled experiments	Controlled (laboratory) experiments
Information Feedback	Selective perception	Complete, accurate, immediate feedback
	Missing feedback	
	Delay	
	Bias, distortion, error	
	Ambiguity	
Decisions	Implementation failure	Perfect implementation
	Game playing	Consistent incentives
	Inconsistency	Consistent application of decision rules
	Performance is goal	Learning can be goal
Interpretation/ Reflection	Misperceptions of feedback	Mapping of feedback structure
	Judgmental biases	Discussability of group process and defensive behaviour
	Defensive routines	
Mental Models	Unscientific reasoning	Disciplined application of scientific reasoning (e.g., systematic falsification)
Strategy, Processes, Decision Rules	Inability to infer dynamics from cognitive maps	Simulation used to infer dynamics of cognitive maps correctly

Table 4: Comparison of empirical learning versus simulation-based learning

With regard to organisational learning, as it was presented in Figure 16, the fact that explicit formal models, which represent and integrate individual

mental models, are accessible to group discussion may help diminish individual misperceptions, erroneous inferences, unscientific reasoning, and judgmental errors and biases of individuals. However, as long as double-loop learning is exclusively seen as empirical learning that evolves mental models only after the fact, i.e. after decisions have been implemented and evaluated in the real world, all obstacles listed above, in principle apply to organisational learning, too.

Since software organisations and software development projects are complex systems, with high dynamic complexity, barriers to learning can be overcome through simulation-based learning. Often, in large-scale industrial software production environments it is very costly, if not impossible, to experiment with alternative development technologies (and processes) on real projects. When experimentation on the real system happens to be unfeasible, a common engineering practice consists of building a model that can be studied by simulation. For example, a simulation model of a software project is a mathematical abstraction that acts as a substitute for the real project but is more amenable to manipulation. It is tempting to adopt the principles of modelling and simulation to study the underlying assumptions and processes by which software organisations and projects are managed.

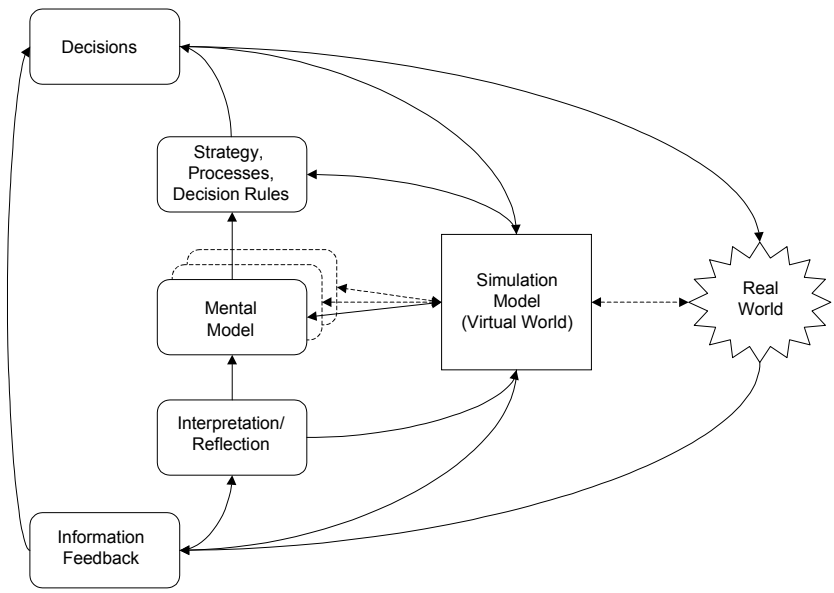


Figure 17: Simulation-based learning

Virtual worlds for learning and training based on simulation models are commonplace in many disciplines. The advantage of simulation in software organisations has been summarised by Christie [Chr99].

3 System Dynamics in a Nutshell

Simulation with System Dynamics models is used for learning about the dynamic complexity of systems, for the identification of optimal policies in existing systems, and for improvement of system behaviour through parameter or structural changes.

System Dynamics was introduced at MIT in the late 1950s to solve industrial problems; hence the original name of *Industrial Dynamics*. In his reference book [For61], Forrester gave two detailed examples dealing with cyclical patterns in production, inventory and employment in the manufacturing area. One of them was the first application of System Dynamics on a real case, the Sprague Electronic Company (electronic components industry). It was shown that variation in demand of the products was not an adequate explanation for an oscillation: the system structure had the inherent tendency to generate it. Sensitivity analysis was performed to indicate which parts were most crucial in determining this behaviour. A combination of new inventory and employment policies was proposed from model experimentation. The implementation of these policies in the real system, reported in [Fey78], supplied a mitigated result: during the first months, the situation was improved; however, after about one year, the managers tended to return to their old employment policy and cyclic behaviour patterns appeared again. Note that the structural explanation for long-term cyclic rise and fall of industrial activity has now become a classical result of the method (see e.g. [Ran80a] who mentions a study of the Norwegian pulp industry focused on inventory fluctuation).

Since the early developments at MIT, the method has been applied to model a wider variety of systems, hence the current name of System Dynamics. A collection of papers published in [Rob78a] reports studies from five broad application domains: manufacturing, marketing and distribution, research and development activities, management control and finance, societal problems (ecological, economical, and sociological).

The most famous of the societal models was the World model [MMR+72] developed by a research group at the MIT on the initiative of the Club of Rome in the early 1970s[19]. It aimed at studying the future of human activity in our finite world, and focused on five physical quantities: population, food production, industrial capital, production and non-renewable natural resources. This work generated much controversy, but gave rise to a new interest in policy modelling efforts in numerous countries.

[19] An updated version of the World model was published in 1992 [MMR92].

During the last ten years the management of software projects has emerged as a new application domain. The ideas underlying the models of software project dynamics originate mostly from the work previously performed at MIT on R&D projects under the direction of Prof. E. B. Roberts [Rob64] [Rob74]. Roberts developed a base model intended to capture the fundamental characteristics of an R&D project and its typical life cycle. The base model is supposed to be adapted to an organisations specific needs. It is reported in [Rob78a] that R&D project models based on the Roberts framework have been implemented by several industrial organisations, among which are Motorola Military Electronics and Sony Labs.

As pointed out in [AbM91], the management of software projects shares common features and problems with the management of R&D projects, i.e., as regards project planning, control of progress, and management of human resources (men and months are not interchangeable). Indeed, the System Dynamics models of software projects incorporate many structural elements of Roberts' model, including [LeL91]:

- typical project variables, such as workforce level, budget, scheduled completion date, number of errors produced, number of errors detected;
- managerial-related functions, e.g. staffing, planning and controlling;
- production-related functions, e.g. design, development, verification, rework;
- human-related functions, e.g. productivity, motivation, error rate, whose values are affected by the project's perceived status (schedule pressure) and the penalty-reward structure of the organisation.

During the 1990s many new applications of System Dynamics in software engineering, not only restricted to software project management, have been published. Based on an exhaustive literature survey, the following application domains of System Dynamics in software engineering have been identified (cf. Section 3.6 for summary descriptions):

- software project management [AbM91][CoM93][LAS97][HeH00],
- concurrent software engineering [PMB99],
- software requirements engineering [ChS00],
- the impact of process improvements on cycle-time [TvC95][Tve96],
- effects of software quality improvement activities [AFO93][Chi93] [Mad94][Mad96],
- software reliability management [Rus98][RuC99],
- software maintenance [CaS99],
- software evolution [LeR99],
- software outsourcing [RCH+00], and
- software engineering training [MaT00].

The tutorial published in [For71] gives a good introduction into the principles of System Dynamics and the underlying mathematics; additional material

may be found in [For61][RiP81][Bos92][Coy96]. The following sections briefly summarise the foundations and the main methodological elements of the System Dynamics method (Section 3.1 to Section 3.4). Section 3.5 provides an overview of the available tool support. Section 3.6 summarises reported applications of System Dynamics modelling in the software domain. In Section 3.7, an evaluation of the existing support for practitioners in applying the method is presented. Based on this evaluation, issues that still need to be resolved were identified.

3.1 System Dynamics Definition

The origins of System Dynamics can be traced back to the theories of servomechanism and cybernetics [Ric90]. Several definitions of System Dynamics have been suggested in the literature (e.g. [For61][Wol90]). The most comprehensive has recently been provided by Coyle ([Coy96], p.10):

"System dynamics deals with the time-dependent behaviour of managed systems with the aim of describing the system and understanding, through qualitative and quantitative models, how information feedback governs its behaviour, and designing robust information feedback structures and control policies through simulation and optimization."

A system, in this definition, is understood as a collection of elements that operate together for a common purpose.

3.2 System Dynamics Foundations

Management is the process of converting information into action via decision-making. Decision-making is in turn controlled by various explicit and implicit policies of behaviour. In the context of System Dynamics, a policy is a rule that states how operating decisions in an organisation are made, i.e. a policy defines which sort of information is relevant to control an action, and how the relevant information is processed in order to control the action.

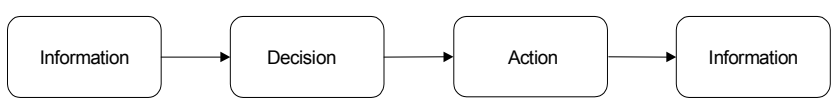


Figure 18: Open loop decision-making

There is not only a conversion from information via decisions to actions, but actions in turn produce new information. In the open loop decision-making process shown in Figure 18 there is no information feedback from the action to the decision. Organisations that are governed by decsision-making processes without information feedback are open systems.

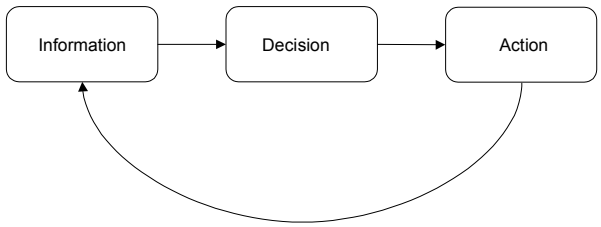


Figure 19: Decision-making process with direct information feedback

In a software organisations, however, most of the information that is produced by one decision-making process is used by other decision-making processes within the organisation, and, usually, there is information feedback, either direct or indirect. A decision-making process with direct information feedback is shown in Figure 19. Figure 20 shows two decision-making processes with mutual (indirect) information feedback. An organisation that is mainly characterised by decision-making processes with direct or indirect information feedback is a closed system or feedback system.

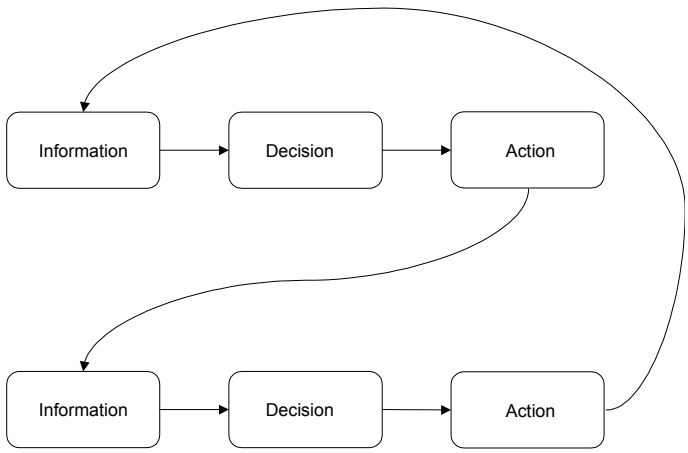


Figure 20: Decision-making processes with indirect information feedback

System Dynamics was mainly developed to analyse the behaviour of closed systems (also called *feedback systems*). The essential characteristic of feedback systems is that their behaviour is generated by its structure, i.e. the physical processes in the system and the set of decision rules that are used to control the physical processes based on information feedback.

For example, in a software project, physical processes relate to the creation, integration, test, and correction of work products, such as requirements specifications, design documents, program code, inspection reports, test specifications and reposts, and so forth. The physical processes would address concepts like productivity, quality (i.e. defect generation and detection), size, and other attributes associated with the actors, activities and arte-

facts involved. The associated decision rules would be related to issues like selection and allocation of work force, selection of requirements, decision upon when to start or stop certain activities, decision about which tool or method to apply, and so forth.

In order to change the behaviour of a closed system (in the example: the project performance) the structure has to be changed. In order to change the system structure, it must be explicitly represented. One way to capture the information feedback contained in closed systems is to use differential equations. In System Dynamics models, each system state (e.g., representing the quality of a work product or the size of the work force) and the associated rules to control this system state, is represented by one differential equation. The set of all differential equations represents the system structure and – if executed – can simulate the system behaviour. Technically, the simulation of the system of differential equations is facilitated by transformation of the differential equations into difference equations and numerical treatment.

3.3 Essential Steps of the System Dynamics Modelling Method

The development of System Dynamics models follows a method that was first described by Forrester in 1961 [For61]. During the last 40 years, this method was slightly enhanced and refined by several authors. The essential steps of the System Dynamics modelling method are as follows:

- Description of problem,
- Definition of reference mode,
- Identification of base mechanisms,
- Construction of causal diagram,
- Construction of flow graph (with model equations),
- Calibration of model,
- Verification and validation of model,
- Policy analysis based on model simulation.

3.3.1 Problem Description

The general goal of any System Dynamics modelling effort is to improve understanding of the relationships between information feedback structure and dynamic behaviour of a system, so that policies for improving problematic behaviour may be developed. From the System Dynamics perspective, a model is developed to address a specific set of questions, which form the problem statement. At the end of the modelling process, the simulation data generated with the help of the model shall be used to answer the questions contained in the problem statement.

In principle, any kind of behavioural phenomenon that occurs in the real world can be subject of the problem definition. Typical examples of questions in a problem statement include: What are the reasons for increasing schedule and effort overruns in the development projects? How should the development process be changed in order to avoid schedule and effort overruns in the future? What is the trade-off relationship between product quality and project schedule? How do schedule overruns of past projects affect the quality and timeliness of subsequent projects in the organisation? Etc.

3.3.2 Definition of Reference Mode

The reference mode is an explicit description of the (problematic) dynamic behaviour of one or more system parameters observed in reality. It acts as a catalyst in the transition from general speculation about a problem to an initial model, and it captures the dynamics of the tackled problem, i.e. behaviour patterns and related time horizon. The reference mode is typically represented by graphs that display the historical or hypothesised evolution of important system variables over time. Reference modes representing empirical behaviour patterns are typically based on measurement data. Reference modes representing hypothesised behaviour patterns should be based on expert opinion.

Examples of reference modes are shown in Figure 21 and Figure 22. Figure 21 is an example of an empirical reference mode that shows problematic project behaviour during the implementation phase. In the example shown, the problematic behaviour is characterised by the steady growth of the average size of program code per inspection (solid line), and by the concurrent decrease in the average number of defects found per inspection (dashed line).

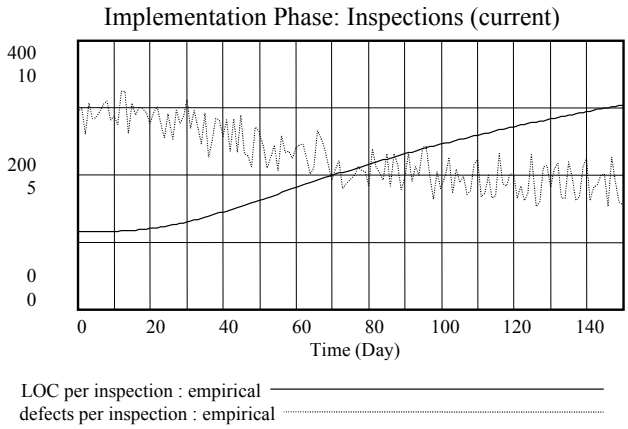


Figure 21: Reference mode (empirical)

Assume that it was found – e.g., based on statistical analysis – that the decrease in inspection effectiveness could be attributed mainly to the increase of the lines of code per inspection, and the noise in the average number of defects detected per inspection could be attributed to variation in preparation time of inspection participants. Now, in order to keep the effectiveness of the code inspections on a constant level, the goal would be to avoid increase of the average size of code inspected per inspection session. Figure 22 is an example of a hypothetical reference mode that describes the intended behaviour of the currently problematic variables after implementation of adequate process changes.

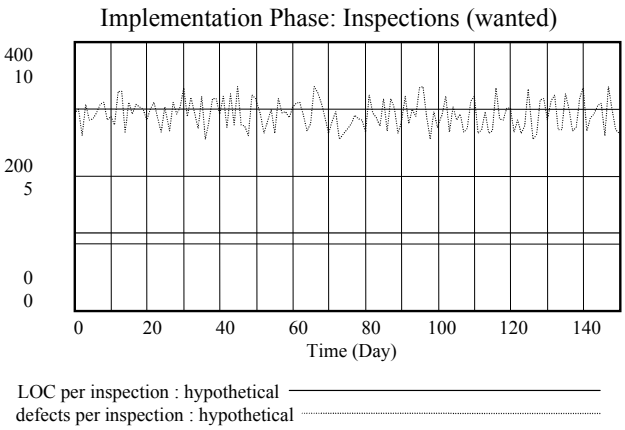


Figure 22: Reference mode (hypothetical)

3.3.3 Identification of Base Mechanisms

In order to analyse – and eventually change – the behaviour of observed objects in the real world, it is necessary to understand the important cause-effect relations of the factors that influence those variables that represent the observed behaviour. In System Dynamics, these cause-effect relations are called base mechanisms. The union set of all base mechanisms is assumed to be the minimal set of cause-effect relations that is able to explain the dynamic behaviour described in the reference mode.

For example, in order to explain the problematic behaviour described in the reference mode shown in Figure 21, the first question would be: What is the reason for the increase in average code size per inspection? Possible reasons could be increase of time pressure or decrease in motivation to follow the inspection guidelines. Again, there are causes for the increase in time pressure or the decrease in motivation. The possible causes have to be explored. Usually, assumptions about possible cause-effect relationships have to be elicited from project participants (managers and developers) for example in interviews. In order to keep the number of relevant cause-effect relations reasonably small, and to focus on the really essential effects, it is recom-

mended to collect as much empirical evidence as possible for any of the considered cause effect relations. Eventually, the set of base mechanisms is documented in the form binary relations. Two types of relations are possible:

Cause + → Effect + (positive impact)

Cause + → Effect – (negative impact)

In the first case (positive impact), an increase in the value of variable *Cause* results in an increase in the value of variable *Effect*. Similarly, a decrease in the value of variable Cause would result in a decrease in the value of variable *Effect*.

In the second case (negative impact), an increase in the value of variable *Cause* results in a decrease in the value of variable *Effect*. Similarly, a decrease in the value of variable Cause would result in an increase in the value of variable *Effect*.

3.3.4 Construction of Causal Diagram

Connecting all base mechanisms into one causal network and representing it in the form of a graph yields the causal diagram. In order to be able to generate the behaviour that is described in the reference mode, it is important that the interconnection of base mechanisms captures the essential information feedback associated with the physical processes under consideration.

For the example reference mode shown in Figure 21, the interconnection of the underlying base mechanisms could result in a causal diagram as shown in Figure 23. If schedule pressure is perceived, this will result in decreasing inspection effectiveness (because larger chunks of code are inspected per inspection session), which in turn results in a decrease of the number of defects detected. Other effects of increased schedule pressure include increased defect injection (due to less careful programming) and faster work progress (due to a reduced number of inspection sessions). Increased defect injection will increase the number of defects detected because the number of detected defects is a function of total number of defects contained in the code. More detected defects will cause more rework which in turn will slow down work progress. Slower work progress, in turn, will increase the time need, which again has a positive impact on schedule pressure.

A closer look at the causal diagram presented in Figure 23 exposes three nested information feedback loops with varying polarity:

- Loop 1: Schedule pressure (+) → Inspection effectiveness (-) → Defects detected (-) → Rework (-) → Work progress (+) → Time need (-) → Schedule pressure (-) [negative feedback loop]

- Loop 2: Schedule pressure (+) → Defect injection (+) → Defects detected (+) → Rework (+) → Work progress (-) → Time need (+) → Schedule pressure (+) [positive feedback loop]
- Loop 3: Schedule pressure (+) → Work progress (+) → Time need (-) → Schedule pressure (-) [negative feedback loop]

Loops 1 and 3 have negative polarity, i.e. they have a damping effect on the dynamics induced by an increase in schedule pressure. Loop 2, in contrast, has positive polarity, i.e. an increase in schedule pressure has a self-reinforcing effect. Whether the aggregate effect of all three loops is positive or negative can only be answered after a thorough analysis of loop dominance.

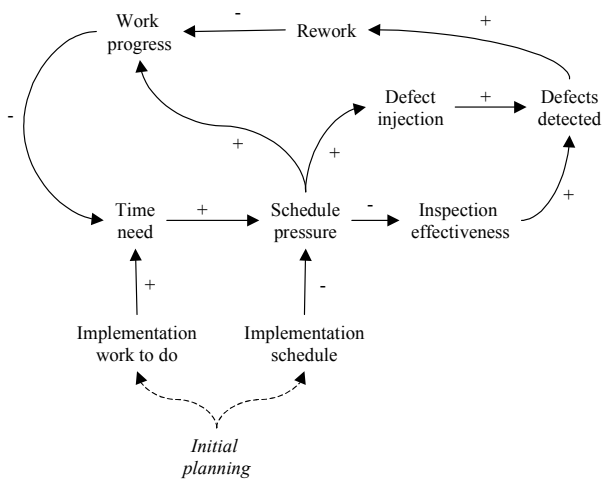


Figure 23: Circular causality underlying the "inspection effectiveness" problem

The philosophical position underlying the System Dynamics method is what Senge and other researchers call *Systems Thinking* [Sen90]. The essential step toward systems thinking is to recognise the presence of feedback mechanisms in the observed system, in a similar way as presented in the example above. In Systems Thinking, the behaviour of a system is considered as primarily being generated by its structure, i.e. the interaction of all the feedback loops over time. Therefore, in the example above, the decrease of inspection effectiveness is diagnosed as originating from endogenous mechanisms, rather than from external disturbances (e.g. initial planning of workload and schedule). The internal structure is deemed as the explanatory factor for behaviour, and an adequate System Dynamics model has to encompass any interaction that is essential to reproduce the behaviour of interest. This also includes the human decision processes, which are not modelled on the micro level of individuals but as aggregated flows resulting from a general policy structure[20]. For example, the hiring of a new developer typically arises within the framework of an (implicit or explicit) organisational manpower acquisition policy. This policy might be an aggressive policy

(i.e., hire and fire), or it might reflect reluctance to change the workforce level, or a mix of both depending on certain conditions.

3.3.5 Construction of Flow Graph

In order to be able to run simulations, the causal diagram has to be transformed into a formal model. In System Dynamics, the formal model consists of a set of mathematical equations. System Dynamics model equations are separated into two groups: level equations and rate equations. This terminology of levels and rates is consistent with the flow-structure orientation introduced by Forrester together with schematic conventions invoking the image of fluid-like processes [For61][For71]. Using these conventions, System Dynamics model equations can be represented (and manipulated) graphically in the form of flow graphs (cf. Figure 24 for available graphical symbols).

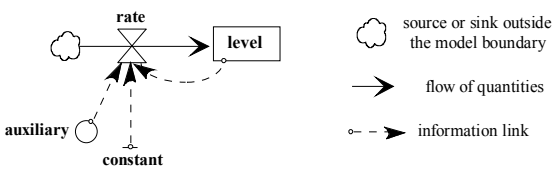


Figure 24: Schematic conventions of flow graphs

The levels in a System Dynamics model describe the state of the system. Knowledge about their values at time t is sufficient to retrieve the instantaneous value of any other variable. They accumulate (or integrate) the results of action in the systems, an action being always materialised by flows in transit. The derivative of a level, or equivalently the rapidity at which it is changing, depends on its input and output flows. In simulation models, the computation of a level is approximated by a difference equation of the form:

$$\text{Level}(t+dt) = \text{Level}(t) + (\Sigma \text{ input rates} - \Sigma \text{ output rates})\, dt$$

The rates are what change the value of levels. Their equations state how the available information is used in order to generate actions. A rate has four conceptual components: an observed condition is compared to a goal, and the discrepancy found is taken as the basis for action (flow) generation. The rate equation that formalises this policy is an algebraic expression that depends only on levels and constant values. Auxiliary variables can be used for intermediate computation.

[20] Note the underlying assumption, central to Systems Thinking, that people placed in the same structure tend to produce qualitatively similar results. Senge referred to this assumption when pointing out that "systems cause their own crises, not external forces or individual's mistakes" [Sen90].

As a concrete example of the use of these notations, let us present a simple mechanism aiming at controlling the number of software engineers working on a project:

Staff_level (t+dt) = Staff_level (t) + Hiring_rate (t) dt

The instantaneous hiring rate is given by:

Hiring_rate (t) = (Indicated (t) - Staff_level (t)) / Adjustment_time

As the staff level grows at a rate that depends on the previously observed state, the equation expresses the presence of direct information feedback (cf. Figure 19). The auxiliary "Indicated" is the goal sought by the loop, which may vary according to some (possibly non-linear) function not detailed here. "Adjustment_time" is a constant indicating the time needed to find and train a new engineer.

The level and rate variables must alternate along any feedback loop, which can be classified according to three attributes:

- polarity, negative (deviation-counteracting feedback) or positive (deviation-amplifying feedback),
- order (number of levels),
- linearity or non-linearity (in rate equations).

It is the presence of non-linear high-order loops in the system that makes the derivation of analytical solutions unfeasible. It can be shown that even simple low-order systems may exhibit a large variety of behaviour patterns; and in practice, any system of interest will involve a high-order, multiple-loop, non-linear structure.

3.3.6 Model Calibration

SDM calibration consists of attributing initial values to levels, setting constant parameters and equation coefficients, estimating lengths and orders of delays, and defining the table functions used to introduce non-linearities that cannot easily be expressed by algebraic functions. To all these problems comprehensive research has been conducted and suitable techniques have been proposed, details can be found in [Pet76][Gra80][Ham80].

3.3.7 Model Verification and Validation

Verification is done in order to check the internal correctness[21] (or appropriateness) of the SDM. The guiding question is whether the model has been constructed right. Validation, in contrast, is done in order to check the external correctness (appropriateness) of the SDM. The guiding questions whether the right model has been constructed. Model verification requires

expert knowledge about the System Dynamics modelling technique, whereas model validation requires expert knowledge about the real system.

Barlas has proposed a set of tests for SDM verification and validation [Bar85][Bar89][Bar94]. In addition to conducting verification and validation tests, Richardson and Pugh propose several criteria to evaluate "general usefulness" of SDMs [RiP81].

Table 5 summarises the possibilities to evaluate SDMs. Barlas distinguishes two categories of tests: structural and behavioural tests. The category of structural tests can be further divided into direct structural tests, which are based on theoretical and empirical evaluation checks, and behaviour-oriented indirect structural tests.

	Structural Tests		Behavioural Tests
	direct	indirect	
Verification	x	x	
Validation	x		x
General usefulness	x		x

Table 5: Tests for the evaluation of System Dynamics models

3.3.7.1 Verification

Verification of SDMs is done through application of direct and indirect (behaviour-oriented) structural tests.

Direct structural tests check for correct transformation of causal relationships into the flow graph, dimensional consistency of model equations, and the meaningfulness of model equations also for boundary values.

Indirect structural tests check for robustness of the model behaviour against minor changes of the model structure, appropriateness of behavioural changes when model parameters alter (parameter sensitivity), and appropriateness of the model behaviour when applying extreme values to decision rules (policy sensitivity). Sensitivity analysis [Tan80] is performed, in order to test whether all chosen factors are essential to reproduce a given behaviour mode, and to study the sensitivity of the model with respect to reasonable variations in parameter values or to a reasonable alternative in model formulation. Since it is a powerful means to identify the active and dormant part of the model, sensitivity analysis also gives an insight into influential places for policy implementation.

[21] Richardson and Pugh prefer the word *appropriateness* [RiP81] recalling Forrester's principle that the validity of SDMs "is a relative matter. The usefulness of a mathematical simulation model should be judged in comparison with the mental image or other abstract model which would be used instead." (cf. [For71], p. 4, chapter 3).

3.3.7.2 Validation

Validation of SDMs is done through application of direct structural tests and behavioural tests.[22]

Direct structural tests can be applied to check whether:

- all major causal relationships have been included in the model structure (face validity),
- the model variables have a meaningful correspondence with entities in reality (representative or conceptual validity),
- the model parameters have been calibrated correctly, i.e., the chosen variables and their initial values correspond to quantities that are meaningful in reality (empirical validity).

Mass and Senge [MaS80] have proposed tests for the appropriate selection of model variables based on empirical data. Statistical methods for evaluating empirical validity have been analysed and described by Morecroft [Mor85], Peterson [Pet75][Pet76][Pet80], and Grcic and Munitic [GrM96].

Behavioural tests are applied to check whether:

- the reference mode can be sufficiently well reproduced by the model,
- there are new or unexpected behavioural patterns in addition to the reference mode behaviour,
- there are behavioural anomalies when single cause-effect mechanisms are excluded,
- model enhancements produce unexpected behavioural patterns.

3.3.7.3 General Usefulness

Questions that help evaluate general usefulness refer to model structure, model behaviour, and model usability. With respect to model structure criteria like appropriate granularity and modularity should be judged. With respect to model behaviour usefulness can be judged by answering questions like:

- Does model simulation provide new insights / does it offer fruitful ideas?
- Can unexpected behaviour be explained through the model?
- Is it possible to generalise the model?

Model usability can be evaluated based on an assessment of the model user interface.

[22] A summary description of general concepts of model validity, their relation to measure validity, and their application to SDMs can be found in [Pfa97b].

3.3.8 Policy Analysis

Simulation-based policy analysis involves the use of the SDM to help investigate why particular decision rules have the effects they do, and to identify new decision rules that can be implemented to improve the problematic behaviour of the real system as described in the problem statement. Policy alternatives correspond to one or a mixture of two kinds of model manipulations: parameter changes and structural changes, i.e., changes in the form or number of model equations. Both involve changing how decisions are made. For example, sensitive policy parameters in a model indicate leverage points in the real world system, i.e. places where a change in quantities – without changing the cause-effect structure – would improve matters. Model changes involving new feedback structure, on the other hand, suggest new ways of processing information for decision-making in the real world for the purpose of improvement.

3.4 Proposed Guidance for SDM Development

Strictly speaking, there is no formal methodology defined for the development of dynamic models, e.g. in the form of a process model. In the related literature, process guidance for SDM development has only been provided on informal and coarse grain level[23]. Several authors, including Forrester [For61][For71], Roberts [Rob64], Randers [Ran73][Ran80b], Richardson and Pugh [RiP81], Bossel [Bos92], and Coyle [Coy96], have suggested sequences of modelling steps. In the following sub-sections, their work is briefly summarised[24].

3.4.1 Forrester (1961/71)

Forrester proposed the following list of "steps in an industrial dynamics study" [For61]:

1. Defining the objectives of the system under study
2. Observing symptoms
3. Detecting symptoms
4. Visualising the system at issues
5. Estimating the boundaries within which lie the causes of the trouble
6. Selecting the factors to be dealt with
7. Constructing a formal model of the preceding

[23] Very recently, Acuña et al. have recognised the lack of guidance for SDM model building in a comprehensive analysis of various descriptive process modelling methods (cf. [AAF+01], Table 2).

[24] The suggestion of Wolstenholme is fully contained in the sequence of modelling steps proposed by Coyle. Therefore, no separate section has been reserved for this author.

8. Using the model simulate system interactions under selected conditions
9. Interpreting the significance of the simulation results
10. Inventing system improvements
11. Repeating all of these steps to move closer and closer to the true problems and to better management policies

The list indicates that Forrester starts by deciding on a problem, proceeds to determine the scope of the system needed to generate the problem endogenously, and then chooses a set of descriptors from inside the system boundary sufficiently detailed to be able to treat the problem. Upon completion of a version of the formal model, it is run to insure that the assumed relations actually do reproduce the problem, and to help decide how the model can be improved. The iterative nature of modelling is recognised.

3.4.2 Roberts (1964)

The second list of "phases in the industrial dynamics approach" was proposed by Roberts [Rob64]:

1. Problem identification
2. Verbal description of the dynamic system theory affecting the problem
3. Mathematical model development
4. Computer simulation of the represented system
5. Analysis of results to determine model validity and factor sensitivity
6. Double-checking of, and data collection regarding, the sensitive areas in the model
7. Simulation experimentation to help identify improved system parameters and policies
8. Implementation of results of investigation in the real world problem areas
9. Evaluation of the effectiveness of the changes, and return to step 1 for continuing improvement, if necessary

Roberts differs from Forrester mainly in his greater emphasis on the need for data collection and sensitivity analysis to increase model credibility, and for special efforts to insure implementation of results. Roberts explicitly describes the largely non-quantitative, descriptive information constituting the basis for his formal model. He also shows how increased understanding of system characteristics can be obtained through systematic changes in model parameter values. Roberts' later work discusses the implementation problem more thoroughly, pointing out the important role of the model user

(or customer) during problem definition and model validation [Rob74][Rob78b].

3.4.3 Randers (1973/80)

Randers proposed the third list of activities for "conceptualizing dynamic models" [Ran73][Ran80b]:

Phase I: Model conceptualisation

1. Familiarisation
2. Problem identification
3. Definition of reference mode
4. Identification of organising concepts and base mechanisms

Phase II: Model formulation

5. Real world description and definition of model boundaries
6. Construction of causal diagram
7. Identification of system levels
8. Detailed definition of (formal) model structure
9. Model parametrisation
10. Model testing
11. Running simulations
12. Policy experimentation

The difference of Randers' proposal to those of Forrester and Roberts is mainly in the more detailed description of modelling activities, which are accompanied by a list of guidelines for the model builder. In addition, by splitting the modelling process into two phases, model conceptualisation and model formulation, Randers aims at reducing the repetition of the full modelling cycle for each single model enhancement. The goal of the conceptualisation phase (steps 1-4) is to arrive at a rough conceptual model capable of addressing a relevant problem. The formulation phase (steps 5-12) comprises two processes: the test of the dynamic hypothesis, which is a preliminary check to see that the basic mechanisms included in the conceptual model actually reproduce the reference mode, and model improvement, which extends and elaborates on the initial model until it is sufficiently versatile and detailed to serve the intended purpose.

The first step after familiarisation with the customer organisation and problem definition is the definition of the reference mode, which acts as a catalyst in the transition from general speculation about a problem to an initial

model. The reference mode captures the dynamics of the tackled problem, i.e., behaviour patterns and time horizon. It is typically represented by graphs displaying the historical or hypothesised evolution of major system variables over time. Having specified the target aspects of system behaviour in that way, the basic real-world mechanisms producing the reference mode must be identified: a causal diagram with few generic loops is issued.

The implementation of the initial model requires turning this model skeleton into a set of equations. The model variables must be chosen: first the levels, then the rates and finally the auxiliaries. At this point, it is useful to build a flow diagram that visualises the parameter interactions. Then, the rate equations are precisely defined.

Having implemented and tested the initial model, the incremental refinement can start. A refinement step may either extend the model boundary by incorporating new causal mechanisms, or refine the existing structure; some examples are the dissection of a level, the replacement of a constant value by a variable, the addition of a new loop, etc. The refinement process is guided by the need to generate a new behaviour mode, to test the effect of a management policy, or to satisfy the customer's expectation with respect to realism and predictive accuracy. The refined models will show a variety of behaviour modes, including the reference mode.

Once credible model structure and parametrisation is obtained by iteration, it is expected that realistic conclusions can be drawn from policy experimentation with the model.

3.4.4 Richardson and Pugh (1981)

Richardson and Pugh propose five stages of model conceptualisaton [RiP81]:

1. Problem definition
2. Model conceptualisation
3. Model formulation
4. Simulation
5. Evaluation

Within these five stages the modeller develops a statement of the context and symptoms of a problem, sketches reference behaviour modes, articulates the purposes of the modelling study, defines the system boundary, and develops a view of system structure in terms of feedback loops of action and information. Figure 25 summarises how these stages and concerns fit together. The connections shown in Figure 25 suggest the overlapping nature of stages in the modelling process. A clear statement of model purpose, for example, contributes to the process of model conceptualisation as well as to the definition of the problem. Feedback structure is likewise a

focus of two different stages of model development, non-quantitative model conceptualisation in terms of diagrams, and model formulation in terms of mathematical equations.

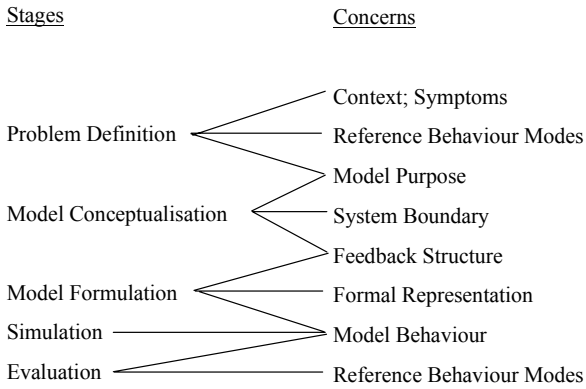


Figure 25: Stages of SDM development [RiP81]

The process proposed by Richardson and Pugh is very similar to that of Randers, but more technical detail is provided for the model formulation and evaluation stages. It is also interesting, that the statement of model purpose includes the model user, an initial view of the policies the model user would like to be able to simulate with the model, and the kind or degree of implementation desired. Richardson and Pugh distinguish three categories of implementation purposes: education and consciousness raising, actual adoption of policy recommendations, or adoption of the model as an ongoing policy-testing tool.

3.4.5 Bossel (1992)

Bossel distinguishes five main stages of the System Dynamics modelling life cycle [Bos92]:

1. Development of the conceptual model (qualitative):
   - Problem description and definition of model purpose
   - Definition of model boundaries
   - Verbal model description
   - Identification of model elements and model structure
   - Construction of causal diagram
   - Qualitative analysis
2. Development of the simulation model (quantitative)
   - Identification of model variables
   - Specification of functional relationships
   - Quantification
   - Definition of (formal) simulation model (flow diagram and model equations)

- Validation of model structure
- Model reduction

3. Simulation of system behaviour
   - Selection of simulation tool
   - Implementation of model
   - Selection of numerical integration method
   - Selection of time step
   - Definition of start conditions
   - Selection of system parameters
   - Specification of exogenous influences
   - Definition of scenarios
   - Definition of result presentation (tables, graphs, reports, etc.)
   - Visualisation of state trajectories
   - Sensitivity analysis
   - Validation of model behaviour
4. Mathematical system analysis
   - Development of a generic model
   - Identification of points of equilibrium
   - Identification of attractors
   - Stability analysis
   - Linearisation
   - Analysis of catastrophic behaviour
5. Change of system behaviour
   - Identification of criteria for the evaluation of behaviour
   - Model changes and optimisation in search of better solutions
   - Stabilisation of unstable systems by changes in parameter values and model structure
6. Identification of generic structures (for reuse)

Compared to the other proposals, the process suggested by Bossel is the most detailed with regard to techniques that can be applied during system analysis and identification of reusable model structures.

Stage 1 deals with the development of the causal structure of the System Dynamics model. Main driver for the definition of model content and model expressiveness is the model purpose. The definition of the model structure requires the identification of model boundaries, and of external influence factors that have an impact at certain points in the model structure. The main activity during stage 1 is the identification of structural elements of the model that are of relevance for the behaviour of the real system. Because the identification of these elements requires the close collaboration between system analyst (model builder) and subject matter experts with knowledge about the real system, first a verbal model description using common natural language is developed. The verbal model is then analysed and transformed into a causal diagram.

Since the qualitative structural model developed in stage 1 cannot be simulated, in stage 2 of the modelling process a formalisation and quantification of the causal diagram is conducted. For this purpose various simulation modelling languages can be used, from standard programming languages to dedicated system dynamics modelling languages. The formalisation of the causal diagram requires the identification of model variable, the specification of the functional relationships between variables, and their quantification. The way in which the formalisation is done should reflect on the type of problem to be solved, the model purpose, and the model user. The formalisation also includes the validation of the model structure through analysis of the model behaviour. In order to improve the model structure alternative model representations can be tested and a reduction of the model structure to its essential cause-effect relationships is recommended.

Stage 3 of the modelling process comprises the selection of the modelling tool, and the implementation of the system dynamics model. After validation of behavioural validity the model is ready for simulation. While reproduction of past (empirical) behaviour can be based on actual external influences on the system, simulations of future behaviour does require the development of scenarios that define future influences. The quality of the simulation-based analysis strongly depends on the plausibility, consistency, and completeness of the specified scenarios. By evaluating the behavioural validity of the model within the interesting intervals of the system parameters, the original modelling task has been resolved.

In addition to the analyses conducted during stage 3, stage 4 offers a variety of mathematical analysis methods that can help to gain insight into the whole range of system behaviour. Starting point for the mathematical system analysis are the level equations. Based on the analysis of the level equations it is possible to identify points of equilibrium and attractors, and to gain insights about system stability and leverage points for changes of behaviour.

Stage 5 of the System Dynamics modelling process deals with the identification of critical parameters and opportunities for improvements of the model behaviour. The goal of the improvement can be the stabilisation of unstable system behaviour or its optimisation with regard to certain evaluation criteria.

3.4.6 Coyle (1996)

Coyle proposes a five-stage approach to system dynamics modelling and analysis [Coy96]:

1. Problem recognition (who cares, and why)
2. Problem understanding and system description (influence diagrams)
3. Qualitative analysis (bright ideas and pet theories)

4. Simulation modelling (formal model building and testing)
5. Policy testing and design:
   a) Exploratory modelling and policy design by simulation (assessment by judgement)
   b) Policy design by optimisation (objective function)

The five stages proposed by Coyle are very similar to those of Bossel. Most interesting is probably the mentioning of so-called "bright ideas" and "pet theories" during qualitative analysis (stage 3).

Bright ideas emerge from experience with other problems. One may have seen something like the set of feedback loops for this problem in some other case, and what was learned then may be applied in the current analysis.

According to Coyle, pet theories are frequently even more useful. They are the views of experienced people in the system of what is wrong with it. The views themselves may be found to be wrong on deeper analysis – and the reasons why they are wrong are of great interest – but they are almost always a useful source of knowledge about the problem and should be searched for by the analyst.

3.5 System Dynamics Tools

System Dynamics model equations can be implemented by using a general purpose programming language like FORTRAN, Pascal and C, or by using a dedicated SDM development software. The first dedicated System Dynamics modelling language and tool was DYNAMO, developed at the MIT in the late 1950s. The first version of DYNAMO, which was available in 1960, bears little relationship to what is now produced. It ran on mainframes, was purely textual, had very limited input/output functionality, and no interfaces to other applications. Today, there exist currently a lot of software packages supporting the development and simulation of SDMs. The characteristics and features of some of these packages are summarised in Table 6, providing references for further reading. A presentation of the "evolutionary lines" of System Dynamics software can be found in Coyle [Coy96].

Tool name (year of first version)	Type of developer	Reference	Characterisation
COSMIC/COSMOS (1984)	COSMIC Holding Company (UK; commercial)	[Coy96]	Descendent of DYSMAP; strong optimisation functionality; comprehensive model analysis functionality; graphical modelling environment
DYNAMO (1960)	Pugh-Roberts Associates (USA; commercial)	[Dyn91]	First System Dynamics simulation language; no graphical modelling language is provided; originally only batch simulation; in the meanwhile, an enhanced version – DYNAMO Plus – supports the building of GUIs and conducting interactive simulation runs
DynaMan (1992)	University of Milano (Italy; non-commercial)	[BFL+92]	Proprietary system of Politechnico di Milano (research prototype) with graphical modelling environment and strong model analysis functionality (similar to Vensim); not further developed; today out of use
DYSMAP2 (1970)	University of Salford / Salford Software Ltd. (UK; commercial)	[VaD87] [Coy96]	The DYSMAP software package was originally developed at the University of Salford based on DYNAMO; today, DYSMAP2 is a commercial tool with purely textual modelling language but comprehensive simulation and analysis functionality
EXTEND (1992)	Imagine That Inc. (USA, commercial)	[RuC99]	Integrates discrete (event-driven/ queuing systems, etc.) and continuous modelling; allows for hierarchical modelling; provides many pre-defined building blocks
MicroWorld Creator (1992) MicroWorld S**4 (1993)	MicroWorlds Inc. (USA, commercial)	[Die92] [Die93]	Supports a textual simulation language, but can import STELLA/Ithink and DYNAMO models; support for design of GUIs for interactive simulations, exploration and playback modes; can models can be connected to management information systems
MIMOSE (1992)	University of Landau-Koblenz (Germany, non-commercial)	[MSF92]	Proprietary University development with special focus on hierarchical modelling; mainly geared at applications in sociology; provides a graphical modelling environment
PowerSim (1992)	POWERSIM AS (Norway, commercial)	[Coy96]	A descendant of Ithink, developed in the context of a Norwegian government sponsored project aimed at improving the quality of high school education using SDMs. This project resulted in the development of Mosaic, an object oriented system aimed primarily at the development of simulation based games for education. Powersim was later developed as a Windows based environment for the development of system dynamics models that also facilitates packaging as interactive games or learning environments; a graphical modelling language is provided
STELLA/IThink (1984)	High Performance Systems Inc. (USA, commercial)	[Ste90]	STELLA/IThink was the first System Dynamics tool that provided a graphical modelling language; recent versions provide a comprehensive functionality for batch and interactive simulation, input/output, and analysis (e.g., sensitivity analysis, Monte Carlo); hierarchical modelling is supported on two levels; the authoring version provides functionality for creating learning environment or management flight simulators
VENSIM (1992)	Ventana Systems Inc. (USA, commercial)	[Ven97]	Supports a graphical simulation language; provides functionality for structural analysis of the model (e.g., causal tracing, loop detection), behavioural analysis (e.g., multi-variate sensitivity analysis, Monte Carlo), optimisation, filtering, model validation, interactive simulation (gaming), storage and comparison of simulation runs, and design of GUIs and learning environments; models can be decomposed into views

Table 6: Selection of System Dynamics software packages.

3.6 System Dynamics Applications in Software Engineering

Published examples of SD applications in software development cover a variety of issues such as:

- software project management,
- concurrent software engineering,

- software requirements engineering,
- the impact of process improvements on cycle-time,
- effects of software quality improvement activities,
- software reliability management,
- software maintenance,
- software evolution,
- software outsourcing, and
- software engineering training.

3.6.1 Software Project Management

In the area of software project management, several different models have been published. Some of them are briefly summarised below.

3.6.1.1 Abdel-Hamid and Madnick

Abdel-Hamid and Madnick developed a first generic software project simulation model with SD in the late 1980s. The book by Abdel-Hamid and Madnick gives a complete description of the model equations and the hypotheses that led to their formulation [AbM91].

Human Resource Management		Captures the hiring, training, assimilation and transfer of people; evaluates the workforce available, newly hired or experienced.
Software Production	**Manpower Allocation**	Evaluates the fraction of manpower allocated to QA, rework, development & test.
	SW Development	Focuses on productivity. Determination of the % of tasks completed.
	QA & Rework	Models the occurrence of errors in tasks, their partial detection (QA) and imperfect removal (rework), evaluates the number of errors that will pass into subsequent phases of development.
	System Testing	Models the propagation of residual errors into subsequent phases of development, and their elimination during testing, determines the percentage of tasks tested.
Controlling		Measures perceived progress and adjusts the job size estimates.
Planning		Adjusts schedule & workforce level according to the estimates.

Table 7: Overview of the model's four subsystems

The model proposed by Abdel-Hamid and Madnck originates in concepts contained in the generic model of R&D project life cycles published by Rob-

erts in the 1960s [Rob64][Rob74], and is based on a comprehensive analysis of small and medium-size software development projects conducted at the NASA during the 1980s. The proposed model consists of about 200 model equations. It integrates three kinds of functions that influence a project's dynamics: managerial related, production-related and human-related functions. It is confined to the development phase, that is, neither the requirements definition phase, nor the maintenance phase is taken into account: as a result, stable requirements are assumed all along the process.

The model consists of four subsystems, each including several kinds of influential functions: the Human Resource Management subsystem; the Software Production subsystem; the Controlling subsystem; the Planning subsystem. Table 7 outlines their main features. All subsystems are interrelated and interdependent; the local modification of a system parameter triggers a chain reaction in the whole system that will eventually affect the initially changed parameter. Simulation output shows behaviour patterns of essential management parameters, i.e. scheduled completion date, estimated cost, manpower loading, and cumulative man-day expenditure. Batch simulations of the model have been performed, in order to gain insight into the general process of software development, and to investigate the impact of some managerial policies on typical example projects. Experimental results can be found in [Abd90][AbM91][Abd93a][Abd93b][ASR93].

Abdel-Hamid and Madnick's model does not consider any breakdown of the project work, assuming the highest possible level of aggregation. This way, it cannot provide a detailed analysis of the intermediate schedule milestones. This consideration also avoids the model to consider a planned staff profile for the project and, hence, in replicating the "Raleigh curve", there is no explicit consideration for the "natural" changes in the work intensity. Finally, the model also considers stable requirements for the project, something extremely unlikely in most medium-large size software projects.

The work of Abdel-Hamid and Madnick brought a significant contribution to the field of software project simulation. Due to the fact that the model has been entirely published and thoroughly commented it has been re-used by other authors as the basis for their studies. For example, Aranda, Fiddaman, and Oliva report the extension of Abdel-Hamid and Madnick's model with a longer time horizon, covering successive software releases and market diffusion [AFO93].

3.6.1.2 Lin et al.

The models developed by Lin et al. at the Jet Propulsion Laboratory consider an explicit breakdown of the software project work into the classic life-cycle stages, providing a more detailed analysis for the schedules, budgets and staff allocation to the project than Abdel-Hamid and Madnick.

The Software Life Cycle Simulator (SLICS) focuses on the problem of requirements changes being introduced throughout the life-cycle [Lin89][LeL91]. An interesting feature of this work is the use of an input and output expert system having fuzzy logic at the interface of the dynamic model. The fuzzy logic has been introduced to handle imprecise information such as subjective input variables (e.g. effectiveness of training methods) or some delay variables (e.g. length of time during which work pressure is applied). The input expert system checks the plausibility of input values, performs preliminary analysis using conditional statements, and transforms the fuzzy values into numerical ones that will be used as inputs to the System Dynamics model. The purpose of the output expert system is to make recommendations based on experimentation with the model; it supports the analysis of output variables using fuzzy algorithms.

The Software Engineering Process Simulator (SEPS) is an extension of SLICS [LAS97]. It provides an important perspective of a software project described as a dual life-cycle process of engineering (i.e. product development) and management (i.e. decision-making). Also important is the proposed procedure to support empirical validation of the model based on several tests. The model has been validated against real data from a completed space shuttle software project, covering a time horizon of about 8 years: the supplied results were accurate within a range of 10% of errors. An experiment has been conducted to test the hypothesis that experienced managers were not able to distinguish between genuine and simulated data [Lin93].

3.6.1.3 Cooper and Mullen

In the scope of their consultant activities at Pugh-Roberts Associates analysing defence and commercial projects, Cooper and Mullen developed a generic SDM that focuses on the rework cycles in software development [CoM93]. In their model they introduced the important concepts of the rework cycle and monitoring ramps. This considers explicitly that rework is generated in the project and remains undiscovered until the later stages. The consequent gap between the perceived and the real progress explains the occurrence of the "90% syndrome". Cooper and Mullen's system provides a very flexible way of capturing the project work structure because the model is developed based on generic "building blocks" to capture the major project activities. This includes design, construction, procurement, testing, staffing categories, and program management. The procedures to apply the model in practice are based on the calibration for a "problem-free" scenario, followed by "what-if" analyses where disturbances are introduced. Despite the modularity of the simulation model, it still assumes a high level view aiming to support the strategic management of large design and construction programs, which often include several projects being implemented in parallel.

3.6.1.4 Rodrigues and Williams

Rodrigues and Williams developed the Project Management Integrated Model (PMIM) in order to demonstrate the usefulness of complementing traditional project management approaches with SDMs [RoW96]. The PMIM considers the use of SDMs at both strategic and the operational management levels providing support to the planning and monitoring functions. In planning, the role of the models focuses on estimating future results and performing risk analysis, while in monitoring they are aimed at diagnosing the project past behaviour. At the strategic level, the high-level System Dynamics Strategic Model (SDSM) is used, which covers the whole project life-cycle and captures the major software development milestones. At the operational level, a more complex model (SDOM) captures in more detail the individual life-cycle phases. The structure of both models is based on an appropriate breakdown of the project into major sub-tasks, and in consistence with the traditional work breakdown structure. The use of the SDSM focuses on providing quick and preliminary assessment of major strategic decisions and risks before a detailed plan is produced. This is particularly important at the early stages when a detailed plan for the whole life-cycle is not available. The SDOM focuses on the project sub-tasks in more detail providing quantitative data to support work scheduling and resource allocation. The PMIM framework has been validated in an industrial case study.

3.6.1.5 Henderson and Howard

Henderson and Howard developed a SDM to analyse large-scale software development projects [HeH00]. The CMPM, the Cellular Manufacturing Process Model, represents a component-based process strategy that uses concurrency and distribution to reduce cycle time. In CMPM, networks of semi-autonomous cells co-operate to produce a complex large-scale system. The model views development as a manufacturing activity where systems are built from components, which are a mixture of self-built, reused and bought-in components. The model is hierarchical, any component may be a product of others. Predicting the cost, quality and schedule outcome of CMPM depends upon the behaviour within cell (intra) and co-operative behaviour between cells (inter) in a dynamic enviroment. CMPM proved to be useful to develop better understanding of both inter and intra cell behaviour and to evaluate effects on project cycle time and predictability.

3.6.2 Concurrent Software Engineering

Powell et al. used a SDM to evaluate strategies for lifecycle concurrency and iteration [PMB99]. Two significant methods for achieving improved cycle-time capability are concurrent software engineering and staged-delivery. Concurrent software engineering exploits the potential for simultaneous performance of development activities between projects, product deliveries, development phases and individual tasks. Staged-delivery enables lifecycle

iteration to supply defined chunks of product functionality at pre-planned intervals. The simulations substantiated the assumption that both techniques provide a promising route to reduced cycle-time, increased product quality and lower development costs.

3.6.3 Software Requirements Engineering

Christie and Staley built a SDM that simulates the Joint Application Development (JAD) process, as implemented by the Computer Sciences Corporation with particular focus on the requirements development process [ChS00]. The model not only explores the organisational issues of requirements development but also the social issues, namely, how effectiveness of "people interactions" does affect the resulting quality and timeliness of the output. The importance of social modelling in determining the outcomes of software processes is demonstrated.

3.6.4 Impact of Process Improvements on Cycle-Time

A common problem faced by most software development organisations attempting to shorten their cycle time is selecting among the numerous process improvement technologies. In order to analyse the impact of improvement technologies on software development cycle time, Tvedt developed a modular SDM that captures a basic waterfall-like development process [TvC95][Tve96]. The effectiveness of new technologies like software inspections can be investigated by experimentation with this model.

3.6.5 Effects of Software Quality Improvement Activities

As in software project management, in the area of software quality improvement and technology change, several different models have been published. Some of them are briefly summarised below.

3.6.5.1 Aranda et al.

Aranda et al. developed a management flight simulator or a "microworld" to support policy evaluation and management decision-making related to the use of TQM (total quality management) techniques for software development [AFO93]. For example, based on simulation runs the long-term gains and short-term costs of quality initiatives like Quality Function Deployment (QFD) or formal inspections can be analysed. The flight simulator has been used as a computer-based learning environment enabling cross-functional product development teams to practice decision making by simulating the entire software product lifecycle in accelerated time.

3.6.5.2 Chichakly

Chichakly reports that High Performance Systems, Inc., used small SDMs in their consulting activities with software organisations to help analyse the following two real-case problems [Chi93]:

- Introduction of a new technology, namely object-oriented design and programming. The SDM was intended to help manage the transition, and incorporated both long-term and effects of the technology (intensive code reuse) and short term ones (inexperience of developers, loss of existing code base of past products, use of new tools).
- Quality improvement. The SDM was focused on the software development life-cycle (specification, design and coding), and aimed at optimising quality initiatives: the introduction of techniques such as reviews and prototyping was simulated at several project stages.

3.6.5.3 Madachy

Madachy developed a dynamic simulation model of an inspection-based software lifecycles process to support quantitative process evaluation [Mad94][Mad96]. His model serves to examine the effects of inspection practices on cost, schedule and quality throughout the lifecycle. Madachy used System Dynamics to model the interrelated flows of tasks, errors and personnel throughout different development phases. The model has been calibrated to industrial data.

3.6.6 Software Reliability Management

Rus et al. describe the use of a process simulator to support software project planning and management [Rus98][RuC99]. The modelling approach here focuses on software reliability, but is just as applicable to other software quality factors, as well as to cost and schedule factors. The original simulator was developed using the System Dynamics approach. As the model evolved by applying it to a real software development project, a need arose to incorporate the concepts of discrete event modelling. The System Dynamics model and discrete event models each have unique characteristics that make them more applicable in specific situations. The structure of the System Dynamics model is presented and the use of the discrete event model to construct a software reliability prediction model for an army project, the Crusader, is described in detail.

3.6.7 Software Maintenance

Cartwright and Shepperd built a SDM to analyse the dynamic behaviour of software systems from a maintenance perspective [CaS99]. As an approach

to understand maintenanace processes and trends, the dynamics generated by information feedback loop was of particular interest. The authors report on a major case study of an information system that was developed for a multi-national organisation. In this case study change documents data was examined over a period of four years. The data includes information on the time taken for each change at six different stages in the change request life cycle and evidence of interesting feedback processes was noted.

3.6.8 Software Evolution

Lehman and Ramil developed a SDM to analyse the impact of feedback in the global software process, i.e. the evolution of software during successive releases [LeR99]. Based on their analyses they could conclude that process dynamics, which typically involve feedback phenomena, are a powerful determinant in the software (and systems) development and evolution process.

3.6.9 Software Outsourcing

The primary objective of the research of Roehling et al. is to determine how software organisations can improve their software outsourcing strategies and processes [RCH+00]. This research utilises System Dynamics simulation modelling to explore the dynamics of outsourcing relationships, including both positive and negative outcomes, as well as to provide potential decision support for strategic outsourcing decisions. The model's current implementation, applicability and usefulness are demonstrated with an example use case and analysis of simulation results. The described maintenance outsourcing model represents fundamental software project components, such as staffing, schedule, learning and costs, which are combined with outsourcing-specific rework, overhead and work effort scheduling.

3.6.10 Software Engineering Training

Litton's Guidance and Control Systems Division used System Dynamics to create mostly small-scale models for investigating managerial process issues and supporting personnel training. In a paper, Madachy and Tarbet argue that System Dynamics simulation is well suited to exploring process issues and even small models are highly valuable for providing insight into dynamic trends [MaT00]. The SDMs have helped managers understand the key factors in complex scenarios. Modelling is also used to support training of software managers. Topics including earned value techniques, productivity estimation, requirements volatility effects and extrapolation of project tracking indicators have been successfully presented with simulation models.

3.7 Open Issues

Based on the review of the available System Dynamics literature it can be concluded that the modelling method as such is mature and well documented, that adequate tool support for implementing and analysing SDMs is available, and that successful applications of System Dynamics in the field of software engineering have been reported. But it can also be recognised that no standard life cycle model for the development of SDMs has emerged, and that no detailed process model for the development of SDMs exists.

Most of the System Dynamics work published consists of presenting models and the results of their application. There is, however, very little support in helping industrial practitioners on how to efficiently and effectively build System Dynamics models. Available guidance is mainly related to technical aspects like identification model variables, parameter estimation, verification and validation, and system analysis based on simulation. In addition, the available guidance does not reflect upon the specific context of software engineering, i.e., alignment with systematic improvement, customisation to software management tasks, support for reuse of data and models that are typically available in software organisations. Detailed support for model building in the form of a process model that covers issues like problem definition, identification of stakeholders (role model), description of entry and exit criteria of modelling activities, definition of a product model, provision of templates and checklists, and guidance on systematic information reuse from other modelling activities and existing models is missing. This lack of sufficiently detailed guidance for SDM developers in software organisations has recently been reconfirmed in a study on process modelling by Acuña et al. [AAF+01].

In order to overcome these weaknesses of the System Dynamics method and to integrate it smoothly with state-of-the-art SPI methodologies, action research was conducted in the scope of the project PSIM within an industrial organisation (cf. Section 4 for details). The objective of this research was to apply the System Dynamics approach "as-is", i.e. from scratch, only based on the available literature, in an industrial software organisation. During the application, the impact on the software organisation and the strengths and weaknesses of the System Dynamics approach could be analysed. Based on the lessons learned from this explorative case study, improved guidance for System Dynamics modelling in SW organisations was developed (cf. Part III of the thesis).

Part II: Action Research and Baselining

4 The PSIM Project

As part of the research for this thesis, a first SDM development project was conducted in the years 1994 and 1995 in a large software development department of Siemens' former Private Communication Networks Group. As a result of this project, the SD model PSIM (Project SIMulator) emerged [Pfa94][PfK95].

4.1 PSIM Project Background

Previous to the PSIM project, the software development processes of the Private Communication Networks Group had been assessed against the Capability Maturity Model (CMM) [PCC+93], and a subsequent re-engineering project was currently ongoing. One goal of this re-engineering project was to improve cycle time such that the overall development time of new software versions could be reduced. A result of this effort was the redesign of the software development processes so that activities within individual development phases could be carried out in parallel, thus facilitating comprehensive concurrent development with well-defined synchronisation points.

4.2 PSIM Project Objectives

The PSIM project was a pilot project. Its purpose was to explore the feasibility of SDM development in an industrial software organisation and to demonstrate the usefulness of SDMs for the planning, control, and improvement of software development processes. As a by-product, simulations with the PSIM model should demonstrate the improvements of the new concurrent development process over the old process.

The duration of the PSIM project was 18 months, from April 1994 to September 1995. The total project effort accumulated to about 1.25 person years, including about 0.25 person years for the customer organisation at Siemens.

From the research point of view, the main objective of the PSIM project was to explore the potentials of the System Dynamics approach in a real setting. The project goals of the potential PSIM model users can be separated into short term and long term goals. The major short-term goals were related to the planning and control of software development projects. The following issues were of primary interest:

- defect generation, detection, and propagation along development phases;
- trade-off effects between project duration, product quality, and man-power allocation (effort);
- effects of unexpected change requests;
- interdependence between software features;
- interdependence between concurrent projects.

In the long run, it was intended to use PSIM as a kind of general-purpose support tool for continuous process improvement.

4.3 PSIM Model Scope

The SD model PSIM covers the software development phases high-level design (HLD), low-level design (LLD), implementation (IMP), and unit and integration test (TST). To reduce the static complexity of the overall model, each development phase was modelled as a separate view. All views are mutually interrelated, reflecting critical aspects of the interdependency between development phases. The interdependencies are mainly determined by the product flow and its defect co-flow, as well as by joint resources, namely human workforce and room availability for conducting inspections. A rough idea of PSIM's overall model structure is given in Figure 26.

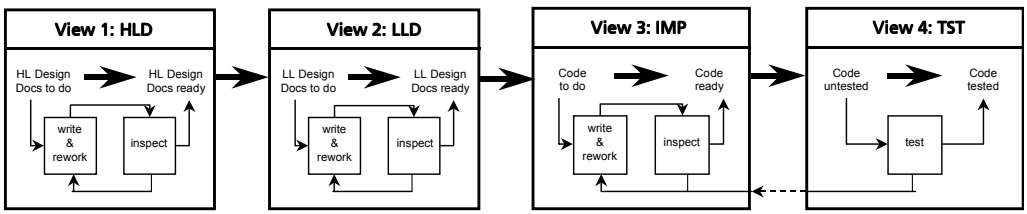


Figure 26: PSIM overall model structure

To each of the software development phases exist both a causal diagram containing the qualitative cause-effect relationships, and a formal flow graph containing the mathematical equations of the simulation model. Details about implementation and functionality of the PSIM model can be found in Appendix A (see also [Pfa94b][PfL00c]).

4.4 Modelling Steps

When the idea was born to use System Dynamics within a software organisation at Siemens, no detailed guidance for the construction of SDMs was available. In the SD literature, only a general modelling process and some rough guidelines were suggested. Based on the recommendations provided

in [Ran80b] and [RiP81], the development of SDMs at Siemens was planned to be done in five steps, which were expected to be iterated several times [PfK95]:

- Step 1: Problem definition. In principle, any kind of behavioural phenomenon that occurs in the system can be subject of the problem definition. It is important, however, that the problem to be analysed is related to dynamic behaviour patterns, and that the scope of the problem definition is reasonably well focused.
- Step 2: Definition of the reference mode. The reference mode is an explicit description of the (problematic) dynamic behaviour of one or more system parameters observed in reality. It acts as a catalyst in the transition from general speculation about a problem to an initial model, and it captures the dynamics of the tackled problem, i.e. behaviour patterns and related time horizon. The reference mode is typically represented by graphs that display the historical or hypothesised evolution of important system variables over time.
- Step 3: Definition of the causal diagram. Having specified the target aspects of system behaviour in Step 2, the basic real-world mechanisms producing the reference mode must be identified: a causal diagram with few generic loops is issued.
- Step 4: Definition of the flow graph and model equations (including implementation of user interface, and model documentation). The implementation of the initial model requires turning the causal diagram into a set of equations. The model variables must be chosen: first the levels, then the rates and finally the auxiliaries. At this point, it is useful to build a flow diagram that visualises the parameter interactions. Then, the rate equations are precisely defined. For static verification, the plausibility of any introduced feedback mechanism must be examined, given the knowledge about the real system. Also, the chosen variables and their initial values have to correspond to quantities that are meaningful in the real system. Some systematic checks are performed on the model equations: dimensional analysis ensures their consistency; rate and auxiliary equations are examined under extreme conditions in order to expose flaws in their formulation.
- Step 5: Simulation for validation and problem solution. After a model version has passed static verification, dynamic testing begins. It is checked whether the values taken by variables during the simulation remain within their valid range even under extreme conditions. In case of unexpected behaviour modes, the opinion of experts in the real system is requested. Sensitivity analysis is performed in order to test whether all chosen factors are essential to reproduce a given behaviour mode, and to study the sensitivity of the model with respect to reasonable variations in parameter values or to a reasonable alternative in model formulation. Since it is a powerful means to identify the active and dormant part of the model, sensitivity analysis also gives an insight into influential places for policy implementation. Once credible model structure and parameter-

isation is obtained by iteration, it is expected that realistic conclusions can be drawn from policy experimentation with the model.

4.5 Knowledge Acquisition Activities

In the context of SDM development, knowledge acquisition is necessary to gather information that helps to specify the reference mode, to derive the causal diagram, to estimate the parameters of the model equations, to calibrate the SDM, and to validate the SDM.

It should be noted that in larger software organisations, there is not a single source of information available. Usually, knowledge is distributed among experts (personal knowledge), process descriptions, measurement databases and many other information sources. Moreover, it is often the case that no 'hard facts' are available and therefore personal interpretations and perceptions of the real system "software organisation" have to contribute to the evolving model.

4.5.1 Organisations and Roles Involved

The development of the PSIM model required extensive knowledge elicitation from different experts. In total, five roles were involved in knowledge acquisition. They are listed in Table 8, indicating role name, role label, and number of persons assuming a role. All roles, except the SDM developer were assumed by members of the customer organisation at Siemens.

Due to uncertainty about the potential applications, the SDM user was not clearly defined from the beginning of the project. First, it was assumed that the members of a Software Engineering Process Group (SEPG) would be the target role, then the project manager seemed to be the most appropriate user, finally it was discussed whether the members of the Project Management Office (PMO) might be the right group of persons to apply the model on a regular base.

Role name	Label	Number of persons
SDM developer	B	1
Facilitator	F	1
SDM user	U	1-5
SDM maintainer	M	1
Software development expert	E	8

Table 8: Organisations and roles in the PSIM project.

The group of software development experts included the project manager, one sub-project manager, one test expert, one configuration management

expert, one quality assurance expert, one requirements engineering expert, one process management expert, and one tools and methods expert.

It should be noted that one person can assume several roles, i.e. the facilitator or SDM user can also be a software development expert. In the PSIM project, for example, the process management expert also assumed the role of the facilitator, the tools and methods expert was also responsible for maintaining the SDM, and the overall project manager was at the same time a candidate for assuming the role of the model user.

4.5.2 Interviews and Reviews

Interviews and subsequent review meetings were the most important activities for knowledge acquisition. A total of 17 interviews and review meetings were conducted, each one with one interviewer and one to three interviewees. The typical duration of an interview or review meeting was one to two hours. Table 9 lists the topics that were covered by an interview or review meeting, and shows the number of meetings and the roles involved (using the same labels as in Table 8).

Interviews and Review Meetings		
Topic	**Number of Meetings**	**Roles involved**
Modelling goals / Problem definition	3	F, U, M
Overall SW development process / Model granularity	3	F, E
Details about test phases	2	E, M
Details about early phases	1	E
User interface	2	U, M
Measurement data for model calibration	3	E
Model validity	3	F, M, U

Table 9: Topics of review and interview meetings

In addition to face-to-face meetings, telephone conversations were conducted and presentations were given to potential users and management.

4.5.3 Data Sources

In order to build the PSIM model, subjective and objective measurement data was needed (cf. [FeP97] for a general discussion of subjective and objective measures). Siemens selected a large project, which was considered to be representative for the organisation, and from which process information and data on several subsequent software versions (V1, …, Vn) could be provided. When the PSIM project started, software version V1 was just about to go into the field, and software version V2 was in its requirements analysis phase.

In the course of the project, mainly two databases could be exploited: a database (DB-A) with information on inspections and tests conducted during software development, and a database (DB-B) with defects detected during field trials and by the customer. From database DB-A, full information about design documents, code inspections, and unit, integration and system tests of software version V1 could be extracted. Typical attributes in the database were the type of document inspected, the effort spent for inspections, and the number of defects found during inspections and development tests.

In addition to databases DB-A and DB-B, information could be extracted from the available project management documents, e.g. planned and actual schedules, effort consumption, and code size.

Since the complete data set for software version V1 was available early during the PSIM project, it could be used for calibrating the PSIM model. As development of the next software version V2 proceeded, new data was continuously added to the databases. This data could be used to validate (and re-calibrate) the PSIM model. After validation, the PSIM model could then be used for planning and controlling the development of software version V3.

4.6 Modelling Results

Based on the results of the knowledge acquisition activities, quantitative models describing essential aspects of the software development process were derived, and to each of the software development phases both a causal diagram containing the qualitative cause-effect relationships, and a more formal flow graph containing the mathematical equations of the simulation model were developed.

4.6.1 Identification of Modelling Goal

As mentioned before, a clear and well-focused goal definition for the modelling project could not be achieved. In the beginning of the project, the SD facilitator was expected to provide the relevant input. His goal was to make PSIM a full micro-world [Die94, Mor88] that would be able to reproduce any important aspect of the software development process in his organisation. In particular, this micro-world should be used to demonstrate that the new development process, designed to enforce concurrent development, was better than the old process. In contrast to this broad organisational, process-oriented goal definition, later in the course of the modelling project, the project manager, one of the potential PSIM users, was mainly interested in getting a project-centred decision support tool that he/she could use for high-level planning and control.

Reference mode: An explicit agreement between model builder and model user(s) on what the reference mode for the PSIM model was, could not be achieved. This was mainly due to the fact that the modelling goal was not

defined precisely enough. As a consequence, the experts were often not able to formulate hypotheses on typical (and/or problematic) dynamic behaviour of the software development system. The fact that historical time series data was not available in sufficient detail worsened this problem.

4.6.2 Dynamic Hypotheses

As a direct consequence of not having a well-focused reference mode in place, the experts used to state assumptions about causal relationships that govern project behaviour only very vaguely. Nevertheless, based on the available data related to the development and field test of software version V1, the SDM developer could come up with several analogy-based mathematical models, namely: a defect slippage model (addressing defect injection, defect propagation, defect detection, etc.), a size prediction model (estimating number of KLOC and number of test cases based on design size), an inspection effectiveness model, a test case effectiveness model, and productivity models (for each phase). These mathematical models were supposed to largely represent the dynamic hypotheses, i.e. the key dynamics that govern the system behaviour (i.e. project performance). Unfortunately, due to lack of time, most of the experts were not available for a detailed discussion of these models. Therefore, it could not be decided whether the SDM developer had identified the most important dynamic hypotheses, and whether they were represented correctly.

4.6.3 Flow Graphs and Model Equations

Based on the interviews, the available process handbook, and the quantitative data, flow graphs were developed for each development phase. In total, the whole model contained 212 variables, and hence, 212 model equations. Among the 212 model variables, 26 were of type level. Table 10 provides a list of the level variables.

PSIM level variables				
Global (1)	HLD (7)	LLD (7)	IMP (7)	TST (4)
Average Overtime	HLD Defects Detected HLD Defects Generated HLD Defects Undetected HLD Inspections HLD Project Variation HLD Work Accomplished HLD Work Remaining	LLD Defects Detected LLD Defects Generated LLD Defects Undetected LLD Inspections LLD Project Variation LLD Work Accomplished LLD Work Remaining	IMP Defects Detected IMP Defects Generated IMP Defects Undetected IMP Inspections IMP Project Variation IMP Work Accomplished IMP Work Remaining	TST Defects Detected TST Project Variation TST Work Accomplished TST Work Remaining

Table 10: List of all levels contained in the PSIM model

4.6.4 Causal Diagrams

For the purpose of documentation, and to ease communication with experts, causal diagrams were developed. They represent the major causal relationships contained in the PSIM model.

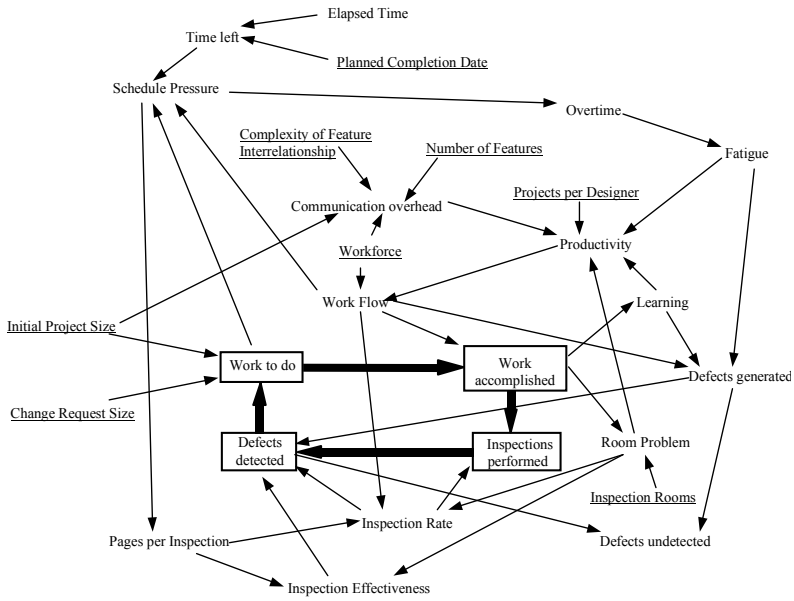


Figure 27: An extract of the causal diagram of phase HLD

Figure 27 shows an extract of the causal diagram of phase HLD, indicating the core process with those level variables that are subject to measurement during project execution (surrounded by boxes), and the network of influencing factors (exogenous model parameters are underlined). The core process can be described as follows: The set of customer requirements defines the amount of work to do (level variable: *Work to do*). The design activity produces a set of design documents (level variable: *Work accomplished*). Before the design documents can be released to the subsequent phase (LLD) they are subject to inspections (level variable: *Inspections performed*). If defects are detected during an inspection (level variable: Defects detected) the design document has to be - at least partially - reworked and goes back to the (virtual) amount of work to do (level variable: *Work to do*).

4.6.5 Model Calibration

The model was calibrated based on the historical data of software version V1 (sources: expert knowledge and database) and the standards defined by the process handbook (PHB). Table 11 lists the variables that were adjusted to

the version V1 data, indicating the information source (E = expert; DB = database; PHB = process handbook).

PSIM model variable	HLD	LLD	IMP	TST
Number of features	E			
Workforce	E			
Defects detected after phase TST	DB			
Size of HLD documents (total)	DB & E			
Size of LLD documents (total)		DB & E		
Size of code (KDLOC / total)			DB & E	
# of test cases (total)				DB & E
# of minor defects detected in inspections (total)	DB	DB	DB	
# of major defects detected in inspections (total)	DB	DB	DB	
Total # of defects found per inspection (average)	derived	derived	derived	
Planned duration of inspections	PHB	PHB	PHB	
Planned # of pages per inspection	PHB	PHB		
Planned # of KLOC per inspection			PHB	
Actual # of inspections performed	DB	DB	DB	
Actual # of pages per inspection (average)	derived	derived		
Actual # of KLOC per inspection (average)			derived	
# of minor defects detected in test (total)				DB
# of major defects detected in test (total)				DB
Total # of defects found per test case (average)				derived
Projects per person (average)	E	E	E	
Complexity of feature interrelation	E	E	E	E
Planned schedule (milestones)	E	E	E	E
Actual schedule (milestones)	E	E	E	E

Table 11: Software version V1 data for PSIM model calibration

4.6.6 Model Validation

For model validation, the simulation results were compared to the actual data of software version V2, which was developed during the modelling project. These comparisons at several points in time (i.e. after completion of HLD, after completion of LLD, etc.) showed good results with respect to model validity. The model predictions anticipated the actual project behaviour with a maximal deviation of less than 10%.

4.7 Lessons Learned

Based on the experience from the PSIM project the following lessons learned seem to be worth being mentioned.

4.7.1 Familiarity with SD Concepts

System Dynamics concepts and principles are something new in the software domain. Therefore, users cannot be expected to have a clear understanding of the benefits, the potential pitfalls, the possible results of an SDM development project. Moreover, the active contribution that is required from them during model building is difficult to motivate. In the PSIM project, this led to uncertainty about the 'right' individuals to be brought into the project, and to unclear expectations about the necessary effort to be provided by them. As a consequence, there were situations where the required experts were not accessible for the model builder. This caused delays in the modelling process. Another serious consequence from missing familiarity with SD concepts was an unclear and in part unrealistic expectation about the modelling results.

The lesson learned is that sufficient time should be spent in the beginning of the modelling project to train all project participants in the basic SD concepts, and to create insight into the needs of a modeller and the support that he/she should receive during the project. Another effect of a sound training would be that unfocussed modelling goals and unrealistic expectations about simulation results could be avoided.

4.7.2 Realistic Expectations

In the PSIM project the initial goal of SDM development, as defined by the facilitator, was too ambitious. Hearing about "simulation of a model that represents the reality of the software department" led to the implicit persuasion of people, that simulation of such a model will automatically answer any possible question that could be asked about the reality of the department and consequences of possible changes to it. Eventually, the SDM was expected to fully represent the overall software development process and serve as a laboratory for experiments with process changes in any direction. As a consequence, and mainly due to the lack of focus, it was hard to find the appropriate scope and granularity for the subsequent modelling steps.

Another observation was that, in general, people seemed to expect answers to their problems to be expressed in numbers (in the sense of point estimates), generated by simulation runs. This expectation is reflected in the way problems were formulated, e.g. "How many days will the schedule overrun be if we assume a workforce gap of 12 developers in the project XYZ?". It took time to convince potential users of the PSIM model and other project participants that the big benefit of the project might arise from the modelling effort itself: by learning about the situation of the department, its staff, and its processes. In a way this is an effect often observed during measurement programs in software organisations: not the collected data and associated models are the most valuable result, but the side effects of the program itself. By conducting the measurement program participants get clarity about different existing perceptions of the reality and come up with

an improved common understanding of their processes and organisational environment.

The lesson from this experience is that when starting an SD project it is important to explain what can realistically be expected: learning about and better understanding of the real system through modelling and simulation in order to improve the decision making capacity.

4.7.3 Clarity about Modelling Goals

As mentioned in the previous sub-sections, it seems to be difficult and time consuming to come up with a well focused and precisely defined goal for a SDM development activity. In the PSIM project, for instance, this problem could never be completely resolved. Starting a SDM development project without a well-defined goal, however, has a high probability of undesired consequences. Some of these consequences showed up in the PSIM project:

- An accepted and generally understood reference mode could not be defined.
- The future user of the SDM was not clearly defined. In the course of the modelling project, different candidate model users with different modelling goals were identified. A final decision could never be achieved.
- The responsibility for the maintenance of the SDM could not be clarified.

Having a fuzzy goal and changing expectations about the model bears the implicit danger that the model ownership will never really shift to the target organisation. Thus, clarity about the goals, the effort, and the expected answers must be considered crucial for success. It is necessary that potential model users have a clear understanding about the purpose of the modelling exercise, and the model builder must clearly identify what kind of results the model user expects. Again, this reminds of a similar experience with measurement programs, where goal-orientation, as defined by the GQM approach, is a necessary prerequisite for success. In fact, it was a lesson learned from the PSIM project that GQM-like concepts should be adopted when developing SDMs:

- Goals: Who wants to learn about what issue in a particular environment in order to better understand the underlying informal and organisational mechanisms that influence decision-making? For example, is he/she the head of department who wants to learn about the reasons why a recent process change did not have the desired effect on the department performance? Or is he/she a project leader who wants to find out the reason why it was impossible to accelerate the cycle time for design completion?
- Questions: What kind of information in what granularity is required to enable the model user to learn about a certain problem area? Inexperienced model users tend to add more and more expectations on the capabilities of the model to provide answers to questions arising along

with the modelling activities. Experience has shown that different individuals coming into contact with the model easily get the impression that their 'own little questions' should also be answered, as it seems to be easy to add ‘a few more’ equations to the model. Adding equations to a model in an unfocused way, however, makes it increasingly more difficult to understand the real system to be captured by the model.

- Metrics: What kinds of numbers have to be produced by simulation runs to answer the questions behind the modelling goal?

4.7.4 Model Size and Complexity

Inexperienced users (and model builders) tend to add more and more information and details to an already existing model. In a few cases, this might be the right way to better and better mirror the reality that is represented by the model. But experience shows that a more detailed model is not automatically a better model. It is obvious, that all the basic facts and related information about the modelling purpose have to be fed into the model in order to get realistic results. The art in modelling is to stop as soon as the adequate level of detail has been reached. Improved simulation precision can only be expected as long as basic facts are missing. But whenever information is added to a model without making a 'real difference', the effect might be counterproductive. The more details are included, the more difficult it becomes to understand all the interrelations of the model parts and variables. Hence, adding detail to a model might weaken its explanatory power.

'Good' SDMs try to figure out the minimal set of feedback loops sufficient to generate realistic system behaviour. Understanding the base mechanisms that determine the behaviour of a software organisation - in other words: learning about the organisation - is sometimes much more complicated as it seems to be at the first glance. This is specifically true as far as 'human aspects' are concerned. To understand a single feedback effect requires hard work. Combinations of feedback loops might be even harder to grasp. Therefore, it is advisable to model small parts of the reality in isolation, before integrating them into more complex structures. Combining well-understood partial models eases the interpretation of the behaviour of the integrated model.

4.7.5 Efficiency in Model Building

Having in mind the difficulties mentioned above it is clear that adequate guidance is needed for SDM development. With only a roughly defined modelling process there is a high probability of losing time and spending effort without payback. To a certain degree this could also be observed in the PSIM project, where (too) many iterations of the modelling steps occurred. Another typical problem for efficient modelling is insufficient availability of empirical data for model calibration and/or validation. Although

lack of data is encountered more frequently in immature organisations, even in high-maturity organisations there is always some probability that a complete and satisfactory set of 'hard data' is missing. Overcoming this lack in a reasonably short period of time requires a good common understanding of how to design the concerned model parts and how to substitute missing data by other types of information (e.g. subjective measures).

The experience from the PSIM project re-affirmed what had been said in the SD literature before: There are intermediate work products in a SDM development project, which are crucial. One of these work products is the reference mode. It must be clear for every participant in SD modelling that without a well-defined reference mode the success of the modelling project is at risk. Therefore, the model builder must be enabled to conduct the modelling process in a way that the definition of a well-defined reference mode is enforced. Another crucial work product is the causal diagram. In the PSIM project, due to insufficient experience with SDM development and lacking guidance, this excellent tool, which is perfectly suited to knowledge acquisition in early phases, was not used in an effective manner. In PSIM, causal diagrams were mainly used for documentation purposes, i.e. in situations when the contents of a flow diagram had to be explained to software development experts during review meetings.

4.8 Summary and Conclusion

During the PSIM project, the following lessons have been learned:

LL1 – Familiarity with System Dynamics concepts: The SDM developer must be skilled and experienced with the System Dynamics method, techniques and tools in order to facilitate efficient and effective model building.

LL2 – Realistic expectations: The SDM users must be informed about the potentialities of the System Dynamics method in order to avoid wrong expectations.

LL3 – Clarity about modelling goals: The SDM developing process needs good support in defining the right modelling goals and good guidance to stick to them during the whole modelling activity.

LL4 – Model size and complexity: It is better to build several small SDMs in iterations than to build one big SDM that tries to capture every possible aspect of interest in a 'big bang' approach.

LL5 – Efficiency in model building: The SDM developing process needs detailed guidance in eliciting knowledge from subject matter experts, in taking advantage from established static modelling methods software engineering, and in reusing information from existing static models.

Based on the experiences made during the PSIM pilot project, the assumption that SDMs are well suited to complement existing qualitative and quantitative methods that are commonly used to analyse processes in software organisations (i.e., process assessment, process modelling, and goal-oriented measurement) was substantiated. Moreover, the feedback received from managers and subject matter experts at Siemens strengthened the belief that SDMs - if developed systematically - can become powerful tools for decision makers, helping them anticipate the impact of process changes to be implemented in the course of improvement (or re-engineering) projects. In particular, a SDM development project can be successful even without resulting in a SDM that generates realistic numbers. The modelling activity itself is a catalyst for establishing continuous learning.

4.9 Refinement of Research Objective and Formulation of Research Hypotheses

Based on the lessons learned, the quite general objective statement of the research conducted in the scope of this thesis (cf. Section 3.7) could be refined and related research hypotheses could be formulated.

4.9.1 Research Objective

After having conducted the PSIM project, the research objective was refined, i.e. it aimed at the development of a framework that supports efficient and effective development of SDMs. In particular, this framework should 1) address the lessons learned from the PSIM case study, 2) build upon existing guidance for SDM development (cf. Section 3.3 and Section 3.4), and 3) integrate SDM development with established static SE modelling approaches of the QIP/EF framework like process modelling (PM) and goal-oriented measurement (GQM). The framework that supports efficient and effective development of SDMs has been given the name IMMoS (Integrated Measurement, Modelling, and Simulation). IMMoS will be described in detail in Section 5 to Section 8.

4.9.2 Research Hypotheses

Related to the refined research objective, two research hypotheses have been defined:

- Research Hypothesis 1 (H_1): SDM development with IMMoS is at least as effective as SDM development without IMMoS.
- Research Hypothesis 2 (H_2): SDM development with IMMoS is more efficient than SDM development without IMMoS.

These two research hypotheses will be used to validate the IMMoS framework (cf. Section 10 for a detailed discussion and presentation of results).

Part III: Innovation

5 The IMMoS Framework in a Nutshell

IMMoS (Integrated Measurement, Modelling, and Simulation) is a framework for effective and efficient SDM development. IMMoS has the following characteristics:

1. It improves existing process guidance for SDM development.
2. It supports SDM goal definition.
3. It enhances the descriptive and explorative power of current state-of-the-art SE experience bases by adding a type of dynamic simulation models, i.e. SDMs, that smoothly integrate with existing static black-box and white-box models, i.e. QMs and DPMs.
4. It combines SDM development with static process modelling (PM), and integrates SDM development with measurement-based quantitative modelling (GQM).

Characteristics 1 and 2 directly relate to PSIM lessons learned LL1 – LL4, the joint set of all characteristics relates to lesson learned LL5 (cf. Section 4.8). Each of the characteristics is represented by an IMMoS element (cf. Figure 28). The IMMoS elements are described in detail in Section 6 to Section 8.

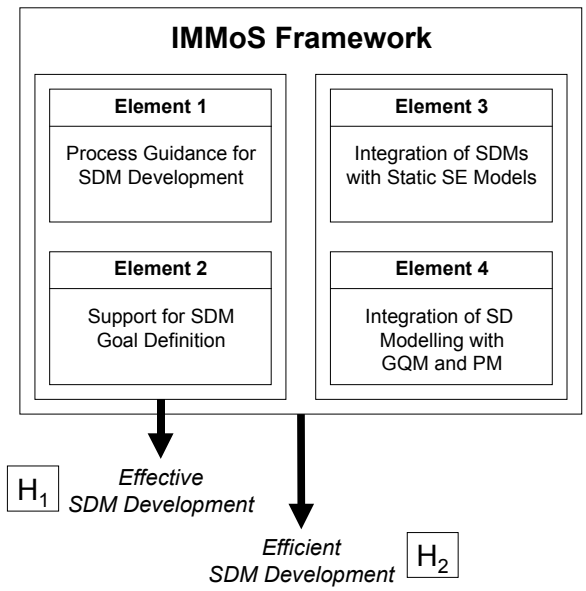


Figure 28: Elements of the IMMoS framework

IMMoS element 1 (Process Guidance), providing process guidance for SDM development, consists of a SDM life cycle model and a SDM development role model, product model, and process model. Role model, product model, and activity model can be combined into a product-flow model. Details are presented in Section 6.

IMMoS element 2 (Goal-Orientation), providing support for SDM goal definition, consists of a SDM goal definition template specifying five relevant dimensions that capture the problem definition during SDM development, namely SDM viewpoint (user role), scope, dynamic focus, purpose, and environment. How certain viewpoint – purpose combinations influence SDM usage is illustrated with the help of selected usage scenarios. Details are presented in Section 7.

IMMoS element 3 (Integration of Models) describes how static SE models like DPMs and QMs are integrated with SDMs. The descriptions are illustrated with an example. Details are presented in Section 8.

IMMoS element 4 (Integration of Methods) describes how SDM development relates to process modelling (PM) and goal-oriented measurement (GQM). Particular focus is put on the integration of SDM development with GQM, enhancing the established GQM method towards "Dynamic GQM". Details are presented in Section 9.

It is expected that IMMoS elements 1 and 2 guarantee the effectiveness of SDM development. The set of all IMMoS elements is expected to improve the efficiency of SDM development. The validation of the IMMoS framework with regard to its impact on effectiveness and efficiency of SDM development is based on a case study and a controlled experiment. Details are presented in Part IV of this thesis (with a summary in Section 10).

6 Process Guidance for SDM Development

Process guidance for SDM development, is provided through a set of models that support the SDM developer:

- The IMMoS Phase Model defines the SDM life-cycle (cf. Section 6.1).
- The IMMoS Role Model defines the roles that are typically involved in a SDM development project (cf. Section 6.2).
- The IMMoS Product Model defines all work products and the end product that occur in a SDM development project (cf. Section 6.4).
- The IMMoS Process Model provides a control-flow-oriented description of the sequence of activities that should be followed in a SDM development project. [25]

The IMMoS Process Model describes each SDM development activity in detail through a set of attributes, such as involved roles and input/output products. With the help of these attributes it is possible to represent the IMMoS Process Model in the form of a product-flow model.[26]

6.1 IMMoS Phase Model

The IMMoS Phase Model structures the SDM life-cycle into four phases (cf. Figure 29):

- Phase 0: Pre-study for the identification of the prospective model user and the modelling goal (SDM goal definition)
- Phase 1: Development of the initial model (reproduction of the reference mode)
- Phase 2: Enhancement of the initial model (to be used for problem analysis/solution)
- Phase 3: Application and maintenance of the enhanced model for problem analysis/solution.

Phase 0 of the IMMoS Phase Model is needed to prepare the actual model building activities. It focuses mainly on the definition of prospective SDM users and identification of SE subject matter experts that can be approached by the SDM developer during the modelling activities. Another important task of Phase 0 is the specification of the modelling goal. If no SDM user can

[25] A first version of the IMMoS Process Model has been published in the form of a technical report [Pfa98a].

[26] A product-flow oriented representation of the IMMoS Process Model is shown in Appendix E. This representation was created with the process modelling tool SPEARMINT [BHK+99].

be identified and no precise SDM goal definition can be achieved, the modelling activity should be stopped.

Phases 1 – 3 of the IMMoS Phase Model represent the major iteration cycles that SDMs typically follow, i.e. first an initial SDM is developed that is able to reproduce the reference behaviour (cf. Section 3.3.2). Then, in Phase 2, this initial SDM is enhanced such that it can be used for problem solving. It might be the case that the SDM user is only interested in a singular problem solution, e.g. when the goal is to evaluate alternative improvement suggestions. In this case, the modelling activities would stop at milestone 3. If the goal is to use the model repeatedly for the same purpose, e.g. when the SDM is used for planning or training, then maintenance and continuous improvement of the SDM is necessary. This is done in Phase 3.

While Phases 1 – 3 of the IMMoS Phase Model more or less reflect the current state-of-the-art in structuring SDM development activities (cf. Section 3.4), the inclusion of Phase 0 does represent an important enhancement by putting more focus on the early SDM development stages.

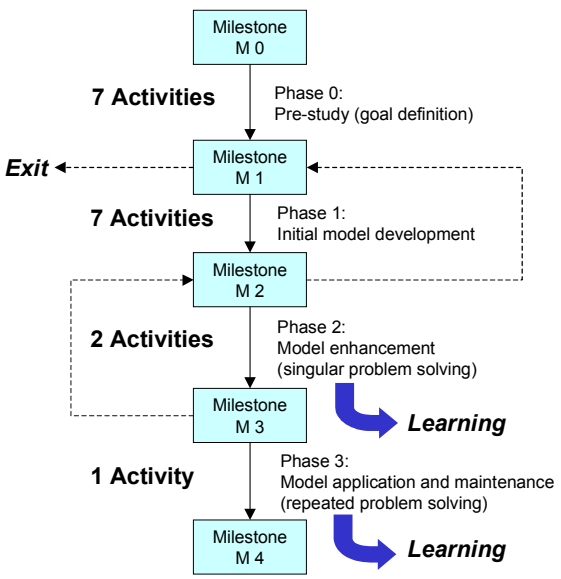


Figure 29: IMMoS phase model

6.2 IMMoS Role Model

The IMMoS Role Model defines the roles that are typically involved in a SDM development project:

- SDM Customer (C)
- SDM User (U)

- SDM Developer (D)
- Facilitator (F)
- Moderator (M)
- SE Subject Matter Expert (E)

6.2.1 SDM Customer

The SDM Customer is the sponsor of the SDM development project. For the SDM development project to be successful it is important that the SDM Customer knows about the cost and benefit of developing and using SDM. This includes a basic understanding of typical application areas of SDMs. The SDM Customer is responsible for the identification of potential SDM Users, and of providing the human resources (i.e. SE Subject Matter Experts) for the SDM development and maintenance task.

6.2.2 SDM User

The SDM User, i.e. the future user of the SDM in the software organisation, is responsible for providing the necessary information for SDM goal definition. In addition, the SDM User participates in all phases of the SDM life cycle, particularly during verification and validation activities, and during the definition of the SDM user interface (when desired). During the SDM application, the SDM User triggers enhancements of the existing model, e.g., recalibration of the model parameters due to changes in the real world.

6.2.3 SDM Developer

The SDM Developer is responsible for technically sound SDM development. In order to fulfil this task, the following skills are needed:

- Sufficient theoretical and practical knowledge about the SD modelling approach (gained through training, relevant literature, and – ideally – active participation in previous SDM development projects). This may include – even though it is not necessary – basic understanding of the principles of system theory and cybernetics, as well as knowledge about the mathematical characteristics of systems of differential and integral equations (including their analytical and numerical solution).
- Sufficient knowledge about at least one SD modelling tool.
- Sufficient communication skills and ability to apply knowledge elicitation, moderation, and presentation techniques.
- Sufficient knowledge about GQM and PM.
- Basic knowledge about the organisational and technical characteristics of the organisational environment in which the SDM development project takes place are useful.

6.2.4 Facilitator

The Facilitator helps with establishing contacts, and planning and arranging meetings. This role is often taken over by C, U, or even D – when D is familiar with the customer organisation. Because the responsibility of the Facilitator is strictly limited to technical support during the conduct of a SDM development project, it will not be explicitly mentioned in the following sections.

6.2.5 Moderator

The Moderator is responsible for preparing and guiding workshops and meetings of the SDM Developer with three or more SE Subject Matter Experts.

6.2.6 SE Subject Matter Expert

The SE Subject Matter Experts are responsible for providing the relevant SE information needed for SDM building. This includes managerial and technological information about how software is developed (processes, methods, techniques, tools, plans, measurement data, etc.) in the organisation.

6.3 IMMoS Product Model

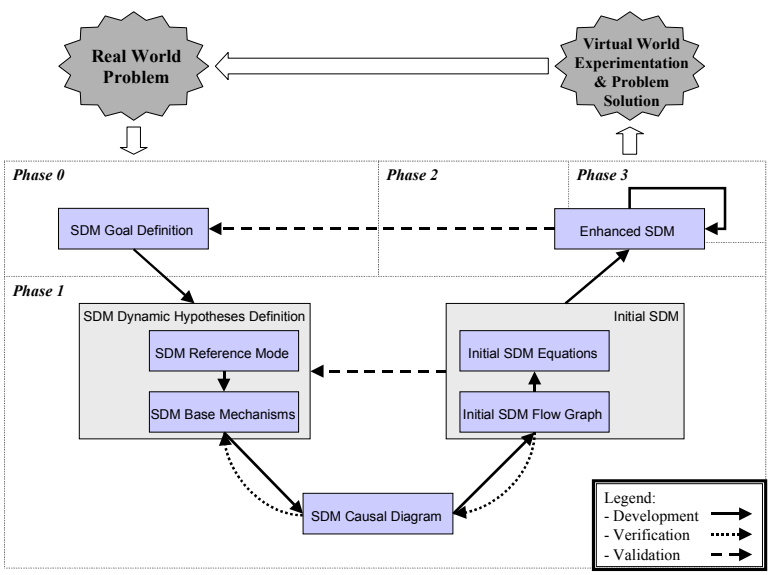


Figure 30: IMMoS product model

Several work products and end products are generated during a SDM development project. These products form the IMMoS product model. Figure 30 presents the IMMoS product model, showing only the essential constructive SD modelling products.

A complete list of work and end products is provided below, with products ordered according to SDM life-cycle phases (constructive modelling products are printed in bold):

Phase 0: Pre-Study

- SDM Agreement: Tentative agreement of SDM Customer to sponsor a SDM development project.
- SDM Customer Sheet: Characterisation of the customer organisation and potential application domain.
- SDM Management Briefing Materials: Presentation slides, experience reports, and other general information materials that are useful to introduce the SD modelling approach on a managerial level.
- SDM Management Briefing Minutes: Minutes of the management briefing session.
- **SDM Goal Definition**: Identification and specification of the problem to be analysed with the help of a SDM.
- SDM Project Plan: Planning of the SDM development tasks (duration, effort, available resources and tools, etc.).
- SDM Project Logfile: A document for recording the SDM project progress, and intermediate SDM development information/results, created and maintained by the SDM Developer. This document can also be used as a source of learning about SDM development.
- SDM Development Contract: Written agreement upon the SDM development project, signed by the SDM Customer.

Phase 1: Initial Model Development

- SDM Technical Briefing Materials[27]: Presentation slides, experience reports, application examples, process descriptions, and other technical information materials that are useful to introduce the SD modelling approach on a technical level.
- SDM Technical Briefing Minutes: Minutes of the technical briefing session.
- SDM Development Workshop Minutes
- **SDM Dynamic Hypotheses Definition**: The definition of the dynamic hypotheses consists of two elements:
  - **SDM Reference Mode**: Graphical representation of the dynamic behaviour of interest.

[27] A System Dynamics tutorial based on SDM Customer Briefing Materials and SDM Technical Briefing Materials developed for the Siemens AG was presented at the ESCOM'2000 Conference [PfL00a].

    - **SDM Base Mechanisms**: Assumptions about the principle causal relations generating the dynamic behaviour of interest.
- **SDM Causal Diagram**: Representation of the minimal network of causal relations necessary to generate the dynamic behaviour of interest.
- SDM Verification Report 1: Documentation of the findings resulting from the verification of the SDM Causal Diagram against the set of SDM Base Mechanisms.
- **Initial SDM**: The Initial SDM consists of three elements:
    - **Initial SDM Flow Graph**: Formal graphical representation of the SDM Causal Diagram, using typed model variables, and distinguishing information links from material flows.
    - **Initial SDM Equations**: Mathematical equations precisely defining the relations between variables in the Initial SDM Flow Graph.
    - Initial SDM GUI (optional): An interactive graphical user interface that allows for easy model execution and analysis of simulation runs.
- SDM Verification Report 2: Documentation of the findings resulting from the verification of the Initial SDM against the SDM Causal Diagram.
- SDM Validation Report 1: The SDM Validation Report 1 consists of two elements:
    - Initial SDM Simulation Results: Simulation data resulting from executing the Initial SDM in order to reproduce the SDM Reference Mode.
    - Initial SDM Simulation Analysis: Documentation of findings resulting from the validation of the Initial SDM against the SDM Dynamic Hypothesis Definition based on an analysis of the Initial SDM Simulation results.

Phase 2: Model Enhancement

- **Enhanced SDM**: The Enhanced SDM consists of four elements:
    - Enhanced SDM Flow Graph: Formal graphical representation that includes the modifications, extensions, and refinements of the Initial SDM Flow Graph. The enhancements are done in way such that policies that are supposed to overcome the observed problems defined in the SDM Goal Definition can be experimented with.
    - Enhanced SDM Equations: Mathematical equations precisely defining the relations between variables in the Enhanced SDM Flow Graph.
    - Enhanced SDM GUI (optional): An interactive graphical user interface that allows for easy model execution and analysis of simulation runs.
    - Enhanced SDM Causal Diagram: If new/alternative base mechanisms were found when enhancing the Initial SDM, the SDM Causal Diagram is updated in order to adequately represent the underlying causal structure of the Enhanced SDM. This is done for documentation purpose only.
- SDM Validation Report 2: Documentation of the findings resulting from the validation of the Enhanced SDM with regards to the correct implementation of suggested policies supposed to adequately handle the problems defined in SDM Goal Definition. The SDM Validation Report 2 consists of two elements:

- Enhanced SDM Simulation Results: Simulation data resulting from executing the Enhanced SDM in order to explore its potential for policy experimentation with regard to the problem defined in the SDM Goal Definition.
- Enhanced SDM Simulation Analysis: Documentation of findings resulting from the validation of the Enhanced SDM based on an analysis of the Enhanced SDM Simulation Results.

Phase 3: Model Application

- **Enhanced SDM**: The Enhanced SDM consists of four elements, i.e. Enhanced SDM Flow Graph, Enhanced SDM Equations, Enhanced SDM GUI, Enhanced SDM Causal Diagram. If necessary (e.g., due to changes in the modelled reality), the Enhanced SDM is updated accordingly.
- SDM Simulation Results: Simulation data and associated analyses resulting from executing the Enhanced SDM for the purpose defined in the SDM Goal Definition.

6.4 IMMoS Process Model

This section provides a detailed description of the IMMoS process activities. Section 6.4.1 provides an overview representation of the IMMoS Process Model using a control-flow-oriented representation. In Section 6.4.2, each process activity is described in detail. Finally, Section 6.4.3 provides a brief specification of the templates offered by the IMMoS Process Model.

A product-flow-oriented representation of the IMMoS Process Model can be found in Appendix E. Based on this representation, which was created with the help of the SPEARMINT tool, a web-based Electronic Process Guide (EPG) has been generated automatically [BHK+99].

6.4.1 MMoS Process Activities – Overview

A summary of the IMMoS process activities is shown in Figure 31, indicating the relationship to standard SE modelling activities, i.e. goal-oriented measurement (GQM) and process modelling (PM). In the column "PM", the labels "DPM" and "PPM" distinguish whether the relationship is with descriptive (DPM) or prescriptive (PPM) process modelling. An "x" in the column "PM" indicates that the relationship can be with both, DPM and PPM. A detailed discussion of the relationships to GQM and PM is provided in Section 8 and Section 9. The labels used in to denote the involved roles have been defined in Section 6.2.

A detailed description of each IMMoS process activity is provided in Section 6.4.2.

Phase 0: Pre-study				
Activity id	Activity name	Roles involved	GQM	PM
0.1	First contact	C, D		
0.2	Characterisation	C, D, F	x	DPM
0.3	Management briefing	C, D, F, potential U		
0.4	Identification of SDM users	C, potential U		
0.5	Problem definition	C, D, U	x	DPM, (PPM)[28]
0.6	Technical feasibility check	C, D, E, F, U		
0.7	Planning and contract	C, D		

Phase 1: Initial Model Development				
Activity id	Activity name	Roles involved	GQM	PM
1.1	Technical briefing	C, D, E, F, U		
1.2	Definition of dynamic hypotheses	D, E, F, M, U	x	DPM, (PPM)
1.3	Definition of the causal diagram	D		
1.4	Review of the causal diagram (verification 1)	D, E, F, M, U		
1.5	Implementation of the initial SDM	D, E, F, U	x	DPM, (PPM)
1.6	Review of the initial SDM (verification 2)	D		
1.7	Test of the initial SDM (validation 1)	D, E, F, M, U	x	

Phase 2: Model Enhancement				
Activity id	Activity name	Roles involved	GQM	PM
2.1	Enhancement of the initial SDM	D, E, F, U	x	DPM, (PPM)
2.2	Test of the enhanced SDM (validation 2)	D, E, F, M, U	x	

Phase 3: Model Application				
Activity id	Activity name	Roles involved	GQM	PM
3.1	Application and maintenance of the SDM	D, E, U	x	x

Figure 31: IMMoS process activities

[28] Normally, a problem definition starts out with the current process baseline, i.e. the DPM. It might happen, however, that problem to be solved addresses the evaluation of a potential PPM that has been defined based on some rationale. If a PPM is involved in activity 0.5, it will also be used in activities 1.2, 1.5, and 2.2.

6.4.2 IMMoS Process Activities – Detailed Description

The detailed description of the 17 IMMoS activities uses a uniform set of attributes. The attributes are defined as follows:

- ID: Activity identification number.
- Name: Activity name.
- Entry condition: The set of conditions that have to be fulfilled before the activity can be started.
- Exit condition: The set of conditions that have to be fulfilled before the activity is completed.
- Role: The set of IMMoS roles that should be involved in carrying out the activity (cf. Section 6.2), and their specific responsibilities.
- Input: The set of IMMoS products (cf. Section 6.4), and other documents that must be available for the activity to be successfully conducted.
- Output: The set of IMMoS products that have been created or modified during the conduct of the activity.
- Description: Summary descriptions of the tasks that need to be carried out when conducting the activity.
- Methods and techniques: Methods and techniques that describe how the tasks of the activity should be carried out, e.g. presentation, meeting (interview, discussion, review), test techniques, etc.
- Guidelines: Best practices, recommended procedures, and "rules of thumb" that should be followed when conducting the activity.
- Materials and tools: materials and tools that should be used when conducting the activity. Materials include templates (cf. Section 6.4.3) and checklists.

6.4.2.1 IMMoS Phase 0: Pre-Study

ID	0.1
Name	First contact
Role	D, C • C: Is empowered and willing to decide whether the development of a SDM should be further investigated. • D: Is an SDM expert.
Input	--
Output	SDM Agreement

Entry condition:

--

Exit condition:

SDM Agreement exists in written form.

Description:

1. Establishment of contact between C and D. The initiative can come from either role.
2. Agreement on the intention to develop and use a SDM in the customer organisation.
3. Rough identification of the SDM application scope and purpose.

Methods and techniques:

Meeting

Guidelines:

--

Materials and tools:

Brochures, experience reports, web-pages, and other references that inform about successful SDM development and usage in software organisations.

ID	0.2
Name	Characterisation
Role	C, D, F • C: Can provide relevant information about the customer organisation to D. • D: Can conduct interviews (oral or in writing) in order to elicit information about the customer organisation. • F: Supports C and D.
Input	SDM Agreement
Output	SDM Customer Sheet

Entry condition:

SDM Agreement exists.

Exit condition:

- D knows the customer organisation sufficiently well (subjective judgement).
- The SDM Customer Sheet exists in written form.

Description:

1. Familiarisation of D with the SDM user organisation, first identification of problems, and listing of findings/issues resulting from document elicitation and the interview with C.

2. Characterisation of the SDM user organisation (size, products, projects, etc.)

Methods and techniques:

Knowledge elicitation techniques[29]:

- Interviews (unstructured or exploratory)
- Document elicitation

Guidelines:

--

Materials and tools:

- Information about the customer organisation: organisation chart, description of business goals, assessment results, process models (actual and official), findings from past or current SPI (or measurement) programmes, quantitative models, etc.
- SDM Customer Sheet template

ID	0.3
Name	Management briefing
Role	C, D, F, potential U • C: Responsible for organisation of briefing session and invitation of participants (C, D, potential U) • D: Responsible for preparation and conduct of briefing session. • F: Responsible for the organisation of the briefing session. • Potential U: Participation in briefing session on invitation by C.
Input	SDM Customer Sheet
Output	SDM Management Briefing Materials, SDM Management Briefing Minutes

Entry condition:

SDM Customer Sheet exists.

Exit condition:

SDM Management Briefing Minutes exist.

Description:

The purpose of the management briefing is to inform the SDM Customer and (potential) SDM Users about the key principles of System Dynamics, and about the types of problems for which SDMs can help find solutions.

[29] A useful taxonomy of knowledge elicitation techniques can be found in [BiS98].

1. Assembly of SDM Management Briefing Materials by D.
2. Preparation and planning of management briefing (identification of participants, date, place, invitation, etc.) by D in collaboration with C (and with support from F).
3. Conduct of management briefing by D.
4. Writing of the management briefing session minutes and distribution to all participants (and other interested parties).

Important topics of the management briefing include:

- How can SDMs support management in software organisations?
- How are standard SPI methods complemented (e.g., process assessments, GQM, process modelling)?
- What types of problems can be tackled with the help of SDMs?
- How does a "typical" SDM project look like? (This includes the presentation of a typical SDM development project plan)
- What are the costs and benefits of SDM development projects?
- What are the success factors of SDM development projects?

Methods and techniques:

Presentation with discussion.

Guidelines:

- Recommended duration of the presentation: 30-45 min
- Recommended duration of the discussion: 30 min

Materials and tools:

Brochures, experience reports, web-pages, and other references that inform about successful SDM development and usage in software organisations.

Standard slide set of SDM Management Briefing Materials [Pfa98c].

ID	0.4
Name	Identification of SDM users
Role	C, potential U • C: Responsible for identification and appointment of SDM User (U). • U: Commitment to participate in development of a SDM. Commitment to use the SDM for problem solving.
Input	SDM Management Briefing Minutes
Output	--

Entry condition:

The SDM Management Briefing Minutes exist.

Exit condition:

U has been identified and formally appointed or SDM development project has been cancelled.

Description:

1. Identification and appointment of U by C.
2. Information of D about this decision (or, alternatively, about the cancellation of the project).

Methods and techniques:

--

Guidelines:

The appointed SDM User (U) should have participated in "Management briefing" (Activity 0.4).

Materials and tools:

--

ID	0.5
Name	Problem definition
Role	C (optional), D, U • C: Checks whether SDM Goal Definition is in line with business goals. • D: Supports U during problem definition. • U: Responsible for problem identification and definition.
Input	• SDM Customer Sheet • If available: process models (DPM or PPM). • If available: measurement-based quantitative models (QMs).
Output	SDM Goal Definition, SDM Project Logfile

Entry condition:

U has been identified (→ Activity 0.4).

Exit condition:

SDM Goal Definition exists in written form or SDM development project has been cancelled.

Description:

1. Identification of a problem that – if solved – would help U with his/her daily work.
2. Formal documentation of the problem definition (SDM Goal Definition).

3. Notes should be taken of all relevant information that could be used to define the dynamic hypothesis in Phase 1, e.g. first assumptions about cause-effect relationships, suggestions of potential problem solutions, relevant existing models, subject matter experts, etc. This kind of information is recorded in the SDM Project Logfile.

Methods and techniques:

Knowledge elicitation techniques:

- Interview (semi-structured or unstructured)
- Focused discussion (goal-related)

Guidelines:

- The problem definition should be well-focused and be stated in concrete terms. The SDM Goal Definition template should be used.
- In order to be suitable for a System Dynamics analysis, the problem has to deal with phenomena that show dynamic behaviour.
- In order to be suitable for a System Dynamics analysis, the system that is going to be investigated for problem solution, has to be viewed as a feedback (or closed) system.[30] This assumption implies that a change in the system structure – and not an alteration of the inputs – is in the focus of interest of the problem solution.

Materials and tools:

SDM Goal Definition template

ID	0.6
Name	Technical feasibility check
Role	C, D, E, F, U • C: Responsible for providing access to documents and SE Subject Matter Experts (if needed). • D: Responsible for checking the (technical) prerequisites for SDM development and usage. • E, U: Support of D • F: Support of C and D
Input	SDM Goal Definition
Output	--

Entry condition:

SDM Goal Definition exists.

[30] A feedback system has a closed loop structure that brings results from past action of the system back to control future action. In other words, a feedback system is influenced by its own past behaviour. Opposed to feedback systems, open systems are characterised by outputs that respond to inputs but where the outputs are isolated from and do not have influence on the inputs. For a discussion of the differences between "feedback" and "open" systems see Forrester [For71].

Exit condition:

Technical feasibility of SDM development has been approved or SDM development project has been cancelled.

Description:

The technical prerequisites for development of an SDM has to be checked, i.e. adequacy of problem definition, availability of experts and data, process/ organisational maturity. This may include that the prerequisites for planning, conducting, and using goal-oriented measurement programmes are checked.

Methods and techniques:

- Technical review
- Structured interview

Guidelines:

- The number and type of prerequisites to be checked depends on the SDM Goal Definition. For example, the development of a SDM for the purpose "planning" may require the existence of empirical data for SDM calibration. The development of a SDM for the purpose "controlling" may – in addition – require the existence of ongoing measurement programme, in order to provide the data to which the SDM simulation results will be compared.
- Relevant technical prerequisites have to be checked on part of the customer organisation, and on part of the SDM Developer.
- A classification and detailed description for developing and using SDMs can be found in [Pfa97b]. Success factors for planning and conducting measurement programmes are provided in [JeB93][JBB94] [31].

Materials and tools:

- Checklist Ch-A.1 [Pfa98c]: Prerequisites for planning, executing, and using measurement programmes.
- Checklist Ch-A.2 [Pfa98c]: Prerequisites for the development and usage of quantitative models with particular focus on System Dynamics modelling.

[31] A reprint of this list of success factors can also be found in [OfJ97].

ID	0.7
Name	Planning and contract
Role	C, D • C: Responsible for contract placement. • D: Responsible SDM project planning.
Input	SDM Goal Definition
Output	SDM Project Plan, SDM Development Contract

Entry condition:

Technical feasibility check was successful (→ Activity 0.6).

Exit condition:

SDM Project Plan exists and SDM Development Contract has been signed by C and D, or SDM development project has been cancelled.

Description:

The effort and time planning for development phases 1 and 2 has to be done by D and proposed to C (SDM Project Plan). If C agrees to the proposed SDM Project Plan, the SDM development Contract is signed. The signing of the contract imposes responsibility on C to provide all necessary resources (access to documents and SE Subject Matter Experts) by D.

Methods and techniques:

Planning meeting.

Guidelines:

Typically, the development of the Initial SDM will consume more than 50% of the total time and effort.

Materials and tools:

- IMMoS Process Model.
- Experience from past SDM development projects (effort, duration).

6.4.2.2 IMMoS Phase 1: Initial Model Development

ID	1.1
Name	Technical briefing
Role	C, D, E, F, U • C: Responsible for communicating the project goals (project kick-off) to all participants (particularly E) • D: Responsible for preparation and conduct of briefing session. Note: organisation (e.g. invitation of participants, room reservation) may be supported by Facilitator) • E, U: Participation in briefing session on invitation by C. • F: Responsible for the organisation of the briefing session.
Input	SDM Goal Definition, SDM Project Plan
Output	SDM Technical Briefing Materials, SDM Technical Briefing Minutes

Entry condition:

SDM Development Contract has been signed.

Exit condition:

SDM Technical Briefing Minutes exist.

Description:

The purpose of the technical briefing is to:

1. Officially start the SDM development project.
2. Inform SDM User and SE Experts about the modelling goals and the SDM modelling project plan.
3. To inform the SDM User and SE Experts about the principles of System Dynamics and how their knowledge will be used to build a valid SDM in order to solve the stated problem.

The following tasks have to be performed:

1. Assembly of SDM project kick-off information (project goals and plan) by C (with support from D and F).
2. Assembly of SDM Management Briefing Materials by D.
3. Preparation and planning of technical briefing (identification of participants, date, place, invitation, etc.) by D in collaboration with C (with support from F).
4. Conduct of technical briefing by C and D. C informs about the project goals and plan. D informs about the technical aspects of System Dynamics modelling (including examples and illustration of benefits).

5. Write minutes of the technical briefing session and distribute to all participants (and other interested parties).

Important topics of the technical briefing include:

- Underlying principles of the System Dynamics method.
- Characteristics of SDMs. In particular, difference to other types of models (statistical models, process models).
- Important steps of the SDM development process.
- Basic concepts/elements of the SDM method (reference mode, base mechanisms, causal diagram, flow graphs, etc.). The concepts should be introduced and explained by using a standard demonstration model.

Methods and techniques:

Presentation, and tutorial with discussion.

Guidelines:

- Recommended duration of the kick-off presentation (by C): 5-10 min
- Recommended duration of the technical presentation (by D): 90-120 min (incl. demo)
- Recommended duration of the discussion: max. 45 min

Materials and tools:

- Brochures, experience reports, web-pages, and other references that inform about successful SDM development and usage in software organisations.
- Standard slide set of SDM Technical Briefing Materials [Pfa98c].
- Demonstration (computer-based)

ID	1.2
Name	Definition of dynamic hypotheses
Role	D, E, F, M, U • D: Responsible for planning and conducting modelling workshops, and – based on the results – definition of dynamic hypotheses. • E: Responsible for provision of expert knowledge, and related documents. • F: Supports D, E, and U. • M: Responsible for moderation of workshops. • U: Responsible for keeping the modelling focus and helping to find SE subject matter experts. (Note: an SDM facilitator may support the organisation of modelling workshops and the identification of E).
Input	SDM Customer Sheet, SDM Goal Definition, SDM Project Logfile If available: • Process Models (DPMs and/or PPMs) • Measurement-based Quantitative Models (QMs)
Output	SDM Development Workshop Minutes (optional), SDM Project Logfile, SDM Dynamic Hypotheses Definition (SDM Reference Mode, SDM Base Mechanisms)

Entry condition:

Technical briefing has been conducted and SDM Goal Definition exists.

Exit condition:

SDM Dynamic Hypotheses exist.

Description:

Dynamic hypotheses are hypotheses on the underlying dynamics that are assumed to cause the problem stated in the SDM Goal Definition. The definition of the dynamic hypotheses requires the identification and explicit description of relevant elements of the mental models of subject matter experts. The definition process includes:

- the specification of the reference mode,
- the identification of organising concepts,
- the identification of the most relevant cause-effect mechanisms (base mechanisms), and
- the identification of the model boundaries.

Supporting activities include:

- the identification of E, and
- the identification and analysis of related DPMs and QMs.

Methods and techniques:

- Workshops,
- Brainstorming and creativity techniques (e.g. the "magnetic hexagon" technique[32] proposed by Hodgson [Hod92]),

- Knowledge elicitation techniques (exploratory and structured interviews, document elicitation)

Guidelines:

- The modelling process starts with the definition of the SDM reference Mode. If related empirical data is available, the reference behaviour can be extracted from this historical data. If no empirical data is available, the reference behaviour can be based on experience or anticipation of U (hypothetical reference behaviour). The rationale underlying the definition of a hypothetical reference mode must be documented.
- If a generic SDM shall be developed (e.g. for training purpose), it is still necessary to define a (generic) reference mode.
- The reference mode shall neither be too general (this is the danger with generic reference modes) nor too detailed (this is the danger with empirically derived reference modes).
- The time horizon of the reference mode bust be chosen sufficiently large in order to comprise sufficient dynamic variance.
- The set of base mechanisms should be as small as possible. On the other hand should the model boundaries be set such that the dynamic variance exposed by the reference mode can be generated from the model structure, i.e. without any exogenous influence.
- The purpose of the base model is to test the dynamic hypothesis. It is not used for predicting future behaviour of the modelled system.
- The structure of the base model should only comprise the base mechanisms. The inclusion of additional cause-effect relations, i.e. in order to represent a larger number of real world entities, will be done during phase 2.

Materials and tools:

- Examples of reference modes, organising concepts, and base mechanisms for illustration/explanation purpose.
- Materials for the application of moderation and creativity techniques.
- Checklist for the planning and conduct of workshops.
- Checklist Ch-B.1 [Pfa98c]: Definition of the dynamic hypotheses (reference mode, organising concepts, base mechanisms, causal diagram).
- A System Dynamics tool might by used for illustrating typical behaviour modes.

[32] A case study describing the use of the "magnetic hexagon" technique can be found in [Lan93].

ID	1.3
Name	Definition of the causal diagram
Role	D: Responsible for the definition of the causal diagram
Input	SDM Base Mechanisms, SDM Project Logfile
Output	SDM Causal Diagram, SDM Project Logfile

Entry condition:

SDM Base Mechanisms exist.

Exit condition:

SDM Causal Diagram exists.

Description:

By connecting the base mechanisms into a network of feedback loops the causal diagram is formed. The causal diagram is expected to represent the network of all cause-effect relations responsible for the (interesting or problematic) dynamics that are observed in the system under consideration.

Methods and techniques:

--

Guidelines:

It is important to provide a detailed documentation of the base mechanisms that are used, i.e. description of cause, description of effect, description of the rationale on which the assumption of a particular cause-effect relationship is base (potential sources in the literature, or data sources can be mentioned).

Materials and tools:

System Dynamics tool.

ID	1.4
Name	Review of the causal diagram (verification 1)
Role	D, E, F, M, U (optional) • D: Responsible for the detailed presentation of the causal diagram, and for keeping records of proposed changes. • E, U: Responsible for the review (inspection) of the causal diagram. • F: Responsible for organising the review meeting. • M: Responsible for the moderation of the review meeting.
Input	SDM Causal Diagram, SDM Project Logfile
Output	SDM Verification Report 1, SDM Causal Diagram, SDM Project Logfile

Entry condition:

SDM Causal Diagram exists.

Exit condition:

SDM Verification Report 1 exists and proposed corrections of SDM Causal Diagram have been made.

Description:

The causal diagram is presented to the participants by D. Then, the causal diagram is reviewed by E and U (optional) with regard to:

- Structural correspondence with reality (i.e. entities and relationships between attributes of entities)
- Correct and complete integration of base mechanisms.

Proposed changes are written down and corrections are performed by D.

Methods and techniques:

Review meeting.

Guidelines:

- First, each individual cause-effect relation contained in the causal diagram is reviewed.
- Then, the correctness of the integration of the individual cause-effect relations is reviewed.
- Finally, the completeness and minimality of the causal diagram is assessed.

Materials and tools:

System Dynamics tool.

ID	1.5
Name	Implementation of the initial SDM
Role	D, E, F, U • D: Responsible for the implementation of the initial SDM. • E: Supports D in defining the functional form of the model equations and in parameter estimation (based on empirical data or on subjective judgement). • F: Supports D, E, and U. • U: Responsible for specifying the requirements for the GUI.
Input	SDM Causal Diagram, SDM Project Logfile If available: • Process Models (DPMs or PPMs) • Measurement-based Quantitative Models (QMs)
Output	Initial SDM (Initial SDM Flow Graph, Initial SDM Equations, Initial SDM GUI), SDM Project Logfile

Entry condition:

SDM Causal Diagram exists and SE subject matter experts are available.

Exit condition:

Initial SDM exists.

Description:

The implementation of the initial SDM consists in transforming the causal diagram into a (formal) flow graph, and in specifying the corresponding model equations quantitatively.

The implementation of the initial SDM comprises the following steps:

- Identification of state variables (levels).
- Identification of rate variables, constants, and auxiliary variables.
- Transformation of cause-effect relationships contained in the SDM Causal Diagram into a flow graph using a graphical editor (flow graph editor). This requires the specification of the model equations (functional relations between flow graph variables).
- Parameter estimation (model calibration).

Supporting activity: Identification and analysis of process descriptions (DPMs), and quantitative models (QMs).

Methods and techniques:

Knowledge elicitation techniques (exploratory and structured interviews, document elicitation).

Guidelines:

The most critical tasks are the identification of state variables (levels) and the specification of the model equations (incl. parameter estimation).

For the identification of the state variable the following guidelines should be followed:

- Application of "snapshot test": The snapshot test is a mental experiment that imagines stopping time in the observed system, freezing all information and material flows instantaneously, as if one took an all encompassing photography (hence the name of the test) of the system capturing intangible and invisible characteristics as well as physical processes. The potential level variables are those that still are visible to the observer of the snapshot, i.e. because they represent tangible entities (such as people, or lines of code). Note that the snapshot test is just a rule of thumb. Constants - at least in the time frame of interest - are not considered as levels. On the other hand, non-material (and thus invisible) entities that have a quasi-material quality (e.g. effort, number of defects) should also be considered as levels.
- The number of state variable should be sufficient, i.e. the value of all other variables in the SDM can be derived from the values of the state variables (and the model constants). On the other hand, the state variables should be independent, i.e. the value of one state variable cannot be derived from the values of the other state variables (at the same point in time).

Guidelines on the identification of rates, auxiliary variables, and constants can be found in [RiP81]. Guidelines on parameter estimation for SDMs can be found in [Pet75][Pet76][Pet80].

The specification of model equations should be based on quantitative models (details will be explained in Section 8).

Materials and tools:

- Checklist Ch-B.2 [Pfa98c]: Implementation of flow graph and associated model equations.
- System Dynamics tool and tools for data analysis (statistical analysis) [Pfa97c][33].

ID	1.6
Name	Review of the initial SDM (verification 2)
Role	D: Responsible for formal verification of Initial SDM
Input	Initial SDM, SDM Project Logfile
Output	SDM Verification Report 2, Initial SDM, SDM Project Logfile

Entry condition:

Initial SDM exists.

[33] A framework for instrumenting measurement programmes with tools has been described in [KRS+00].

Exit condition:

SDM Verification Report 2 exists and proposed corrections of Initial SDM have been made (or return to previous activities).

Description:

In order to verify the technical soundness of Initial SDM, the following checks are made:

Check of the SDM Flow Graph with regard to:

- Correctness and completeness of the implementation of organising concepts and base mechanisms (i.e., boundary adequacy [RiP81]).

Check of SDM Model Equations with regard to (for details refer to [RiP81]):

- Consistency of variable dimensions.
- Correctness of the arithmetics (e.g., no division by zero, extreme conditions),
- Numerical fit (plausibility of numerical values of constants and coefficients),
- Conceptual fit (plausibility of the chosen functional forms).

Check of the robustness of the Initial SDM (sensitivity with regard to parameters and structures).

Proposed changes to the Initial SDM are documented, and necessary corrections are performed.

Methods and techniques:

Walkthrough, model checking, and statistical analysis.

Guidelines:

For robustness checks sensitivity analyses and Monte Carlo simulation can be conducted.

Materials and tools:

- Checklist Ch-B.2 [Pfa98c]: Implementation of flow graph and associated model equations.
- System Dynamics tool and tools for data analysis (statistical analysis) [Pfa97c].

ID	1.7
Name	Test of the initial SDM (validation 1)
Role	D, E, F, M, U • D: Presentation of the initial SDM and of results of simulation runs. • E: Responsible for the analysis and interpretation of simulation results. • F: Supports D, E, and U in preparing and organising the test workshops. • M: Responsible for the moderation of test workshops. • U: Responsible for the interpretation of simulation results, and evaluation of the GUI.
Input	Initial SDM, SDM Project Logfile
Output	• SDM Validation Report 1 (Initial SDM Simulation Results, Initial SDM Simulation Analysis), • Initial SDM, SDM Project Logfile

Entry condition:

SDM Verification Report 2 exists and corrections of Initial SDM have been made.

Exit condition:

SDM Validation Report 1 exists and corrections of Initial SDM have been made (or return to previous activities).

Description:

Simulation runs are conducted and analysed in order to check whether:

- the Initial SDM reproduces the SDM Reference Mode (boundary adequacy for behaviour [RiP81]),
- the model structure is appropriate for U with regard to size, simplicity/complexity, and aggregation/detail (model utility [RiP81], and
- the Initial SDM GUI is acceptable for U.

If sufficient quantitative data is available, empirical correctness can be checked with the help of statistical tests that are particularly suited to SDMs (cf. [Bar85][Bar89][Bar94]).

Methods and techniques:

Test workshops, statistical analysis.

Guidelines:

Richardson and Pugh provide a taxonomy of tests for building confidence in SDMs [RiP81]. They distuingish between tests for checking the suitability for the model purpose, tests for the consistency with reality, and tests for the contribution of a suitable and consistent model to utility and effectiveness for U. In each category, a distinction is made between tests that focus on model structure and tests that focus on model behaviour.

Main focus of activity 1.7 is on testing model consistency and model utility. With regard to consistency, the level of appropriateness depends on the model purpose. SDMs for the purpose of learning require less formal (or empirical) testing than SDMs for the purpose of planning and controlling. Thus, appropriateness of SDMs for learning can be based on face validity (with regard to both structure and behaviour), whereas appropriateness of SDMs for planning and controlling should, in addition, be based on statistical tests (based on quantitative data) that evaluate empirical correctness.

Barlas has developed a set of statistical tests for the validation of SDMs, classified into structural tests, sructure-oriented behavioural tests, and behavioural tests [Bar85][Bar89][Bar94]. An example implementation of some standard goodness of fit measures for the empirical correctness of SDMs can be found in [GrM96].

Materials and tools:

- Checklist Ch-C [Pfa98c]: Verification and validation of System Dynamics models.
- System Dynamics tool and tools for data analysis (statistical analysis) [Pfa97c].

6.4.2.3 IMMoS Phase 2: Model Enhancement

ID	2.1
Name	Enhancement of the initial SDM
Role	D, E, F, U • D: Responsible for the implementation of the enhanced SDM. • E: Supports D in defining the functional form of the model equations and in parameter estimation (based on empirical data or on subjective judgement). • F: Supports D, E, and U. • U: Responsible for deciding when the modelling activity should stop (possibly with decision support from E), and responsible for specifying additional requirements for the GUI.
Input	SDM Goal Definition, Initial SDM, SDM Causal Diagram, SDM Project Logfile If available: • Process Models (DPMs and PPMs) • Measurement-based Quantitative Models (QMs)
Output	• Enhanced SDM (Enhanced SDM Flow Graph, Enhanced SDM Equations, Enhanced SDM GUI, Enhanced SDM Causal Diagram…), • SDM Project Logfile

Entry condition:

Initial SDM has been validated and proposed corrections have been implemented.

Exit condition:

U considers the enhanced SDM to be ready for use.

Description:

Modifications, extensions, and refinements of the initial SDM are done in a way such that policies that are supposed to overcome the observed problems can be experimented with. This may include a re-calibration of model parameters.

When new/alternative base mechanisms are found and included into the initial model, the corresponding causal diagram is updated accordingly. This is done for documentation purpose.

The GUI, allowing for easy model execution and analysis of simulation runs, has to be adjusted to the model enhancements.

Supporting activity: Identification and analysis of process descriptions (DPMs and PPMs), and quantitative models (QMs).

Methods and techniques:

Knowledge elicitation techniques (exploratory and structured interviews, document elicitation).

Guidelines:

- Under no circumstances, the enhancement of the SDM should start before the initial SDM has been validated.
- In order for the enhanced SDM to not become overly complex, model equations should only be added to the initial SDM when this does yield new model behaviour patterns, which are either of interest for the problem under investigation, or create more realistic behaviour and thus let appear simulation results more convincing.
- A new model equation should represent an important relation between attributes of real world entities. New model equations should not be added based on mere intuition.
- When the SDM is in danger to become too complex, the level of detail should be reduced (and not the model scope).
- The causal diagram is an important conceptual tool for building the initial SDM. For the enhanced SDM the importance of the causal diagram is reduced to the level of documentation and communication. Any enhancement of the SDM should be directly made in the flow graph (and afterwards be documented in the causal diagram).

Materials and tools:

- Checklist Ch-B.2 [Pfa98c]: Implementation of flow graph and associated model equations.

- System Dynamics tool and tools for data analysis (statistical analysis) [Pfa97c].

ID	2.2
Name	Test of the enhanced SDM (validation 2)
Role	D, E, F, M, U • D: Presentation of the enhanced SDM and of results of simulation runs. • E: Responsible for the analysis and interpretation of simulation results. • F: Supports D, E, and U in preparing and organising the test workshops. • M: Responsible for the moderation of test workshops. • U: Responsible for the interpretation of simulation results, and evaluation of the GUI.
Input	SDM Goal Definition, Enhanced SDM, SDM Project Logfile
Output	• SDM Validation Report 2 (Enhanced SDM Simulation Results, Enhanced SDM Simulation Analysis), • Enhanced SDM, SDM Project Logfile

Entry condition:

Enhanced SDM exists.

Exit condition:

SDM Validation Report 2 exists and corrections of Enhanced SDM have been made (or return to activity 2.1).

Description:

Simulations are run to check whether the structure and behaviour of the enhanced SDM is adequate with respect to the problem statement. In addition, the usability of the SDM is evaluated (model complexity, GUI). Proposed corrections are documented and implemented.

Methods and techniques:

Test workshop.

Guidelines:

The enhanced SDM must only be used by U for the specified purpose.

Materials and tools:

- Checklist Ch-C [Pfa98c]: Verification and validation of System Dynamics models.
- System Dynamics tool and tools for data analysis (statistical analysis) [Pfa97c].

6.4.2.4 IMMoS Phase 3: Model Application

ID	3.1
Name	Application and maintenance of the SDM
Role	D, E, U
Input	Enhanced SDM, SDM Goal Definition
Output	Enhanced SDM, SDM Simulation Results

Entry condition:

SDM Validation Report 2 exists and corrections of Enhanced SDM have been made.

Exit condition:

Enhanced SDM is no longer valid or SDM Goal Definition has become obsolete.

Description:

The SDM is applied in order to find new policies that solve the problem(s) described in the problem statement. This includes experimenting with parameter values and model structures. If necessary (e.g., due to changes in the modelled reality), the Enhanced SDM is updated accordingly.

As a consequence of using the Enhanced SDM for the purpose "improvement", development processes may be altered (and documented in the form of a PPM).

If the Enhanced SDM is used for the purpose "controlling", concurrently to the application of the Enhanced SDM, a related measurement programme has to be conducted.

Methods and techniques:

Causal tracing of SDM variables[34], statistical analysis.

Guidelines:

The enhanced SDM must only be used by U for the specified purpose.

Materials and tools:

System Dynamics tool and tools for data analysis (statistical analysis).

[34] Advanced SDM tools (such as Vensim) provide structural tracing tools allowing the model user to gain a deeper understanding of the impact of cause-effect relations on simulation outcomes.

6.4.3 IMMoS Templates

The following sections provide outlines of the IMMoS templates for SDM Customer Sheet and SDM Goal Definition.

6.4.3.1 SDM Customer Sheet Template

The SDM Customer Sheet template consists of eight fields:

<field 1> Name of customer organisation
<field 2> Name of SDM Customer (C)
<field 3> Name of SDM Developer (D)
<field 4> Date
<field 5> List of documents
<field 6> List of findings/issues resulting from document elicitation
<field 7> Date and place of meeting with C
<field 8> List of findings/issues resulting from interview with C

6.4.3.2 SDM Goal Definition Template

The SDM Goal Definition template consist of ten fields:

<field 1> Name of customer organisation
<field 2> Name of SDM User (U)
<field 3> Name of SDM Developer (D)
<field 4> Date
<field 5> SDM Scope
<field 6> SDM Dynamic Focus
<field 7> SDM Purpose
<field 8> SDM Role
<field 9> SDM Environment
<field 10> Additional information provided by U, e.g. first assumptions about cause-effect relations (base mechanisms), suggestions of potential problem solutions, relevant existing models, names of subject matter experts in the customer organisation, etc.

7 Support for SDM Goal Definition

Even though a clear and adequate definition of the SDM goal seems to be essential, not much support has been provided in the SD literature.

In the IMMoS framework, support for SDM goal definition is provided through a SDM goal definition taxonomy that assists SDM developers and SDM users in defining the modelling goal. Recently, categories for classifying SE simulation models according to scope and purpose have been published [KMR99]. These categories can be used as a first input for defining a SDM goal definition taxonomy.

The main starting point, however, for defining the SDM goal definition taxonomy has been the GQM goal definition template used to define measurement goals.

7.1 GQM Goal Definition

A GQM goal is specified along five dimensions, i.e. object, quality focus, purpose, viewpoint, and environment [BCR94b][BDR96]. Using these five dimensions, an example GQM goal definition would be as follows:

> Analyse the <object>
> for the purpose of <purpose>
> with respect to <quality focus>
> from the viewpoint of <role>
> in the context of <environment>.

Examples objects are *processes* and *products*, example purposes are *understanding*, *prediction* and *evaluation*, examples for the quality focus are *effort consumption*, *time needed*, *productivity* and *quality*. The slot <viewpoint> is defined by the user role. The organisational unit in which the data collection and interpretation takes place defines the slot <environment>.

7.2 IMMoS Goal Definition

Similar to the five GQM goal definition dimensions, five dimensions are used to specify the SDM development goal in the IMMoS framework:

- Dimension 1 – Role: Who will be the user of the SDM? This dimension is identical to the slot <viewpoint> in the GQM goal definition. In the context of this thesis only managerial roles will be taken under consideration.

- Dimension 2 – Scope: What is the SDM boundary and granularity. This dimension is similar to the slot <object> in the GQM goal definition.
- Dimension 3 – Purpose: Why is the SDM developed? This dimension is identical to the slot <purpose> in the GQM goal definition.
- Dimension 4 - Dynamic Focus: What particular dynamic behaviour is in the focus of interest? This dimension is similar to the slot <quality focus> in the GQM goal definition.
- Dimension 5 – Environment: In which organisational environment is the SDM developed and applied? This dimension is identical to the slot <environment> in the GQM goal definition.

To each of the five dimensions various values can be assigned. In contrast to GQM where only one value can be assigned to each dimension, in IMMoS, dimensions 1 (role), 3 (purpose), and 4 (dynamic focus) can have more than one value assignment at the same time. This is mainly due to higher complexity of SDMs, as compared to QMs, which facilitates more complex usage scenarios involving several roles, purposes, and dynamic foci.[35] There are, however, certain dependencies between roles and scopes and roles and purposes that restrict the possibility to combine values in an arbitrary way. Details will be provided in the following sections.

It should also be mentioned at this point that the slot <environment> does not have direct influence on a SDM development activity. Similar to the GQM case, the context information is only of relevance for the packaging and storage of a SDM in the organisations experience base. The provision of context information for packaging is an essential prerequisite for reuse and organisational learning. Since these aspects are out of the scope of the research conducted for this thesis, the interested reader is referred to [Bir00] for more details.

7.3 IMMoS Goal Definition Taxonomy

Table 12 shows an extract of the IMMoS goal definition taxonomy. The taxonomy lists possible values for each of the five SDM goal definition dimensions.

In three cases, i.e. for dimensions *Role*, *Scope*, and *Purpose*, value assignments are restricted to a limited set of pre-defined possibilities that can be grouped hierarchically on two levels of refinement. The asterisk in dimension *Purpose* indicates that a related SDM goal definition requires empirical learning to take place, including the collection of measurement data.

35 An example scenario explaining the usage of a SDM with a complex goal definition can be found in [Pfa98b] (Example SDM Goal Definition -- Role: process engineer and project manager; Scope: single project; Dynamic Focus: project duration, effort consumption, product quality; Purpose: understand, plan, control, and improve; Environment: software organisation ABC).

For the dimensions *Dynamic Focus* and *Environment* no normative value assignment can be made. This is due to the large variety of possible values. For example, *Dynamic Focus* can be directed toward effort consumption, quality, cost, productivity or any other attribute related to a software development artefact, activity or resource. Also combinations of several values are feasible. This is due to the nature of SDM development, i.e. the underlying paradigm of Systems Thinking, which advocates multi-causal analyses to explain observed or anticipated dynamic behaviour.

SDM Goal Definition Dimension	Multiple Values	Refinement Level 1	Refinement Level 2
Role	Yes	Strategic Management	Line Management
			Process Management (process owners, SEPG)
			Product Management
			Human Resource Management
			Configuration Management
			Quality Management
			etc.
		Project Management	Project Management
			Project Management Office (PMO)
		SE Subject Matter Experts	*<further normative refinement not in scope of work / non-managerial roles are not in the focus of interest>*
		Trainer	*<further normative refinement not in scope of work / relevance is restricted to role Management Trainer>*
Scope	No	Organisational level	Long-term organisation development
			Long-term product evolution
			Multiple, concurrent projects
		Project level	Development project (with or without maintenance)
			Sub-project (portion of project life-cycle)
Purpose	Yes	Understanding	U-1: Analysis of current or past reality
			U-2: Transfer of knowledge
		Planning	P-1: Prediction based on current reality
			P-2: Evaluation of planning alternatives
		Controlling	C-1: Validation of planning (*)
			C-2: Benchmarking (*)
		Improving	I-1: Exploration of improvement opportunities
			I-2: Improvement of current reality (*)
Dynamic Focus	Yes	*<further normative refinement not possible>*	(cost, effort, quality, time, productivity, size, functionality, trade-off between effort consumption and quality, etc.)
Environment	No	*<further normative refinement not possible>*	(attributes characterising the organisational unit: size, product domain, location, culture, etc.)

Table 12: Extract of the IMMoS goal definition taxonomy

In the following sections, each dimension will be discussed in more detail. The size and structure of the possible value sets associated with each dimension determine the depth of discussion.

7.3.1 Role

Even though there does not exist an accepted standard set of software engineering roles, a rough distinction can be made between roles that are primarily related to strategic management, i.e. roles dealing with organisational issues, and roles that are primarily related to project management. In addition to management-oriented roles, there exist a variety of roles that relate to technical software engineering tasks, and to software engineering and managerial skill development.

Table 13 shows an example role taxonomy that was derived from a concrete software organisation [Pfa97a]. Four role clusters could be identified in this organisation. For each role cluster there is at least one role name. In two cases, for the management-related role clusters, supporting roles are listed. Then, for each role, it is indicated whether a particular role is mainly active on strategic (organisational) or project level.

Role Cluster	Role Name	Supporting Roles	L[a]
Strategic Management	Department Head		O
		Department QM	O
		Process Engineer	O/P
		Product Manager	O/P
Project Management	Project Manager		P
		Project Management Office (PMO)	P
		Project QA	P
		Feature responsible	P
Technical Software Engineering	Developer		P
	Tester/QA		P
Skill Development / Training	Trainer		O/P

a. L indicates whether the role mainly acts on strategic level (O = organisation) or on project level (P = project).

Table 13: Example Role Taxonomy

The following sub-sections briefly characterise role clusters contained in the example role taxonomy.

7.3.1.1 Strategic Management

The decision what product is going to be developed and when the next project will be started is made on organisational level. These are strategic decisions which are complemented by other long-term activities like human resource management, marketing, product evolution, process evolution, choice of tools and software architectures, etc. Typical roles on organisational level include Department Head, Department Quality Manager (QM), Process Engineer, and Product Manager. The Process Engineer is the facilitator of *Organisational Learning* having the difficult task to continuously

bridge the gap between the organisational and project perspective by collecting lessons learned from current projects, integrating them into the organisation's memory (experience base), and introducing positive experience into future projects via improved processes.

7.3.1.2 Project Management

Product development is always carried out within a project. The Project Manager is responsible for carrying out the project successfully. In larger projects the Project Manager can rely on a project staff covering roles like project related quality assurance (Project QA), or a Project Management Office (PMO).

7.3.1.3 Technical Software Engineering

In each project there is a team of subject matter experts in charge of developing the software product. Basically, this team can be divided into two groups, represented by the roles Developer (including, e.g., Analyst, Designer, Programmer, etc.) and Tester.

7.3.1.4 Skill Development / Training

Trainers are the facilitators of *Individual and Group Learning*. Similarly to the role Process Engineer, the Trainer has to bridge the gap between the organisational and project perspective.

7.3.2 Scope

The SDM Goal Definition dimension *Scope* has a finite set of five possible values. This set of values was adopted from the simulation model characterisation grid proposed by Kellner et al. [KMR99]. A dependency from the dimension *Role* can be established through refinement level 1. Strategic management related roles will be particularly interested in simulation models having their scope in the organisational level, whereas project management related roles will be more interested in models with scope in the project level.

7.3.3 Dynamic Focus

The SDM Goal Definition dimension *Dynamic Focus* specifies which aspect of the system to be modelled is of particular interest. For example, in a model representing the behaviour of software development projects, aspects like effort consumption, duration, resource allocation to tasks, amount of rework, product quality, and so forth could be of interest. Typically, these

aspects measure certain attributes of real world entities. In contrast to the state of practice in GQM modelling, in the context of System Dynamics modelling, only the dynamic behaviour of the modelled system – i.e. the change of attribute values over time – is in the focus of interest.

In SDM development it is typical that several mutually related aspects of the modelled system are in the focus of interest. For example, in project management, it does not make sense to put focus only one aspect of the "magic triangle" cost/effort – time – functionality/quality. All three aspects are so closely related that they always have to be taken under consideration jointly. Generally, due to the holistic approach of systems thinking, in System Dynamics modelling, it is not useful to assume that only one aspect be included in one SDM goal definition. This is in contrast to GQM goal definition, where exactly one value can be assigned to each goal dimension, and the resulting models that are derived from a goal definition are hierarchically structured.

7.3.4 Purpose

The main purpose of SDM development and usage is learning. In Section 1, four elements of the generic process of model-based learning were identified (cf. Figure 6): understanding, planning, controlling (or evaluation), and exploration (for improvement). System Dynamics modelling and simulation can support each of these elements of the learning process, i.e. the purpose of a SDM is to support one or more of these learning steps. Direct support can be provided to understanding, planning, and exploring. Since control requires the availability of empirical data, only a combination of System Dynamics simulation and goal-oriented measurement.

In a detailed analysis [Pfa97a][Pfa98b], it was found that the four main purpose types could be further refined (cf. Table 14). The asterisk ("*") indicates that refined purposes C-1, C-2, and I-1 cannot be fulfilled by SDM development and simulation without setting up and running in parallel a related measurement programme. The columns "SDM development" and "SDM simulation" indicate how the refined purposes relate to the SDM life cycle. For example, purpose U-1 can be addressed by only developing an SDM, whereas purpose P-1 can only be addressed by running simulations. In the following sub-sections each of the four purposes will be discussed in detail and the type of learning that is triggered by the activities related to SDM development and simulation are summarised.

Purpose	Refinement	SDM development	SDM simulation
Understanding	U-1: Analysis of current or past reality	yes	yes
	U-2: Transfer of knowledge (Training)	yes	yes
Planning	P-1: Prediction based on current reality	no	yes
	P-2: Evaluation of planning alternatives	no	yes
Controlling	C-1: Validation of planning (*)	no	yes
	C-2: Benchmarking (*)	no	yes
Improving	I-1: Exploration of improvement opportunities	no	yes
	I-2: Improvement of current reality (*)	no	yes

Table 14: Refinement of SDM Goal Definition dimension *Purpose*

7.3.4.1 Understanding

The development and application of a SDM for the purpose of understanding can include both:

- analyses of dynamic behaviour of current or past reality, and
- transfer of knowledge about dynamic behaviour of current or past reality to others (training).

7.3.4.1.1 Understanding of current or past reality (U-1)

The development of a SDM based on information about current or past reality is a source of learning because it helps understand why system states that are of interest behave in the observed way if certain start conditions and exogenous influences are in place.

As soon as a SDM exists that reproduces current or past behaviour of reality, systematic variation of model parameters (i.e., sensitivity analysis or inclusion and exclusion of model structures) can help understand the sources of dynamic behaviour. In particular, the nature of trade-off relationships between system states can be investigated. This kind of analysis can form the basis for risk analyses and exploration of improvement opportunities (cf. Section 7.3.4.2 and Section 7.3.4.4).

Understanding of current or past reality comprises two steps. Table 15 lists the steps, provides a brief characterisation of each step, and lists for each step the necessary prerequisites. Table 16 summarises the kind of learning that is induced by addressing purpose U-1.

U-1 steps	Characterisation	Prerequisite
U-1.1	Based on analysis of past or current reality R, development of the SDM. The SDM development facilitates understanding of empirical reality R.	--
U-1.2	Running simulations with the SDM and conducting analyses based on simulation results. The analyses facilitate understanding of both the empirical reality R and the SDM.	U-1.1

Table 15: Sequence of U-1 steps

Addressed Purpose	Prerequisite	Induced Learning
U-1	--	Learning by authentic modelling and simulation (L-1)

Table 16: Induced learning by addressing purpose U-1

Example uses of SDM development for purpose U-1 include:

- Identification of relevant system states to describe observed dynamic behaviour, e.g. defect flow between project phases.
- Better understanding of the causal relationships between system states, e.g. factors that influence changes in motivation of project staff over time.

Example uses of SDM simulations for purpose U-1 include:

- Analysis of dynamic error injection patterns in software development projects.
- Analysis of causes why process changes did not produce the expected improvements in project performance.
- Analysis of the impact of variation in external influences to the modelled system, i.e. as a basis for risk analyses.
- Analysis of the impact of variation in system parameters or system structures as a basis for investigating potential improvement opportunities.
- Analysis of trade-off effects between system states. For example, a SDM that simulates project behaviour can be used to investigate the mutual dependency between product quality, project duration, and effort consumption.
- Analysis of system stability. For example, a SDM that simulates project behaviour can be used to investigate how much average project duration can be shortened without triggering unwanted side-effects like extreme cut-down in product quality or sudden increase of project staff turnover.

7.3.4.1.2 Transfer of Knowledge / Training (U-2)

An existing SDM can be used to explain structure and behaviour of a real system to others. Particularly interesting is the possibility to visualise and analyse the behaviour of systems with high dynamic complexity like software development projects. Without the help of a simulation model it is very hard

to grasp dynamic complexity of a system, particularly if own experience with the empirical system is limited. But not only simulating an existing SDM, also the development of an SDM (or of parts of an SDM) in the scope of a training activity can provide new insights, particularly about causal relationships between system states, and the number and type of possible influences to the modelled system.

Hence, the transfer of knowledge (or training) with the help of a SDM can consist of two steps as shown in Table 17. The second step can be conducted based on a pre-defined model, or based on step U-2.1. Table 18 summarises the kind of learning that is induced by addressing purpose U-2.

U-2 steps	**Characterisation**	**Prerequisite**
U-2.1	Based on appropriately packaged information about past or current reality R (e.g. provided by a trainer), development of the SDM. The SDM development facilitates understanding of empirical reality R.	U-1
U-2.2	Running simulations with a SDM and conducting analyses based on simulation results. The analyses facilitate understanding of both the empirical reality R and the SDM. The used SDM can be provided as-is (i.e. by a trainer), or can be the result of step U-2.1. In the first case, the trainer will get feedback about the model from the model users (i.e. trainees).	U-1; U-2.1

Table 17: Sequence of U-2 steps

Addressed Purpose	**Prerequisite**	**Induced Learning**
U-2	--	Learning by mediated modelling and simulation (L-2)

Table 18: Induced learning by addressing purpose U-2

Example uses of SDM development and simulation for purpose U-2 include the education of computer science students and the training of software project managers.

7.3.4.2 Planning

Planning with the help of SDMs refers to the prediction of behaviour of the modelled system based on simulation runs. For the case that all starting conditions and model input are fixed, the planning task consists of executing exactly one simulation run. For the case that at least one starting condition or model input can vary, the planning task can consist in finding the optimal value assignment with regard to a defined objective function.

7.3.4.2.1 Prediction based on Current Reality (P-1)

An existing SDM that is supposed to represent a subset of current reality can be used to predict the behaviour of the modelled system by running simula-

tions. The simulation results, i.e. point estimates or behaviour patterns, can be used for planning purposes.

Table 19 lists possible sequences of steps that have to be conducted in order to address purpose P-1 properly, provides a brief characterisation of each step, and lists for each step the necessary prerequisites. Table 16 summarises the kind of learning that is induced by addressing purpose P-1.

P-1 steps	Characterisation	Prerequisite
P-1.1	Based on an existing SDM a simulation run is conducted with the required starting conditions and input data. The simulation result facilitates learning in two ways: a) There is a solution to the planning problem (P-1.1 successful) b) There is no solution to the planning problem (P-1.1 not successful) In the case that there is a solution to the planning problem its implementation in the real system can be started.	U-1
P-1.2 (optional)	Even in the case that the step P-1.1 was successful, the planning solution may or may not coincide with the expectations of the planner. In either case, comparing the simulation results with expectations triggers learning.	P-1.1
P-1.3 (optional)	Even in the case that the step P-1.1 was successful, the planning results may or may not coincide with planning data generated by other available models (not necessarily SDMs). In either case, comparing the planning results with planning results generated by other models triggers learning.	P-1.1

Table 19: Sequence of P-1 step

Addressed Purpose	Prerequisite	Induced Learning
P-1	U-1	Learning by planning (L-3)
	P-1.1	Learning by comparison of planning results with expectations (L-4)
	P-1.1	Learning by comparison of planning results with planning data generated by other models (L-5)

Table 20: Induced learning by addressing purpose P-1

Example uses of SDMs for purpose P-1 include:

- A SDM representing the behaviour of a software development project can be used to estimate certain project parameters (e.g. cumulated effort consumption, number of inspections conducted, number of defects found, increase of time pressure) at a pre-defined project milestone.
- Inversely to the previous example, a SDM representing the behaviour of software projects can be used to predict the point in time at which certain model states have reached predefined values (effort consumption, number of defects detected, etc.).

7.3.4.2.2 Evaluation of Planning Alternatives (P-2)

In the case that the start conditions and input data for a simulation run are not fixed, several planning alternatives are possible. Evaluation of these planning alternatives with regard to an objective function can be part of planning.

Table 21 lists possible sequences of steps that have to be conducted in order to address purpose P-2 properly, provides a brief characterisation of each step, and lists for each step the necessary prerequisites. Table 22 summarises the kind of learning that is induced by addressing purpose P-2.

P-2 steps	**Characterisation**	**Prerequisite**
P-2.1	Based on an existing SDM, n simulations are conducted with varying starting conditions and input data (cf. step P-1.1 in Table 19). In case that there is at least one feasible solution for the planning problem, the optimal solution is selected. The simulations facilitate learning in two ways: a) There is an optimal solution to the planning problem (P-1.1 successful) b) There is no solution to the planning problem (P-1.1 not successful) In the case that there is a solution to the planning problem its implementation in the real system can be started.	U-1; P-1.1
P-2.2 (optional)	Even in the case that an optimal planning solution could be found in step P-2.1, the simulation results may or may not coincide with the expectations of the planner. In either case, comparing the planning results with expectations triggers learning.	P-2.1
P-2.3 (optional)	Even in the case an optimal planning solution could be found in step P-2.1, the simulation results may or may not coincide with planning data generated by other available models (not necessarily SDMs). In either case, comparing the planning results with planning results generated by other models triggers learning.	P-2.1

Table 21: Sequence of P-2 steps

Addressed Purpose	**Prerequisite**	**Induced Learning**
P-2	P-1.1	Learning by evaluating planning alternatives (L-6)
	P-2.1	Learning by comparison of optimal planning results with expectations (L-4)
	P-2.1	Learning by comparison of optimal planning results with planning data generated by other models (L-5)

Table 22: Induced learning by addressing purpose P-2

Example uses of SDMs for purpose P-2 include:

- A SDM modelling software development projects can be used to estimate the optimal setting of start conditions (e.g., maximal available total effort, distribution of effort over project phases, percentage of work

products to be inspected) such that the project concludes within a certain time frame with a certain product quality.

- With the help of a SDM that models the process of allocating of software requirements to subsequent product releases, optimal allocation strategies can be calculated depending on the average speed of requirements creation (and other parameters, e.g. complexity of requirements).

7.3.4.3 Controlling

Controlling refers to the comparison of simulation results with related empirical data. A typical control situation results from the comparison of plan data with measurement data. Too large discrepancy between plan and measurement data is an indicator that either the model was not valid when the planning took place, or since the planning was made, reality has changed significantly so that the model does no longer represent reality in a proper way.

7.3.4.3.1 Validation of Planning (C-1)

In order to build trust into the validity of the SDM and the planning that was made with it, a regular comparison of the planning with the behaviour of the real system is recommended. This can be achieved by conducting a measurement programme and by comparing plan data with measurement data. Too large discrepancy between plan and measurement data is an indicator that either the model was not valid when the planning took place, or since the planning was made, reality has changed significantly so that the model is no longer valid.

Table 23 lists a sequence of possible steps that have to be conducted in order to address purpose C-1, provides a brief characterisation of each step, and lists for each step the necessary prerequisites. Table 24 summarises the kind of learning that is induced by addressing purpose C-1.

C-1 steps	Characterisation	Prerequisite
C-1.1	The comparison between measurement data and plan data can have two different outcomes: a) Plan data equals measurement data, i.e. planning was correct. b) Plan data differs from measurement data, i.e. planning was incorrect. Interpretation Outcome a) can be interpreted as follows: Interpretation 1: SDM was correct when planning was made; SDM still represents the current reality R; Outcome b) can be interpreted in three different ways: Interpretation 2: If there are no indications that the modelled system has changed significantly since the point in time when planning took place, it must be assumed that the SDM was not valid when planning took place and should be corrected. Go to step C-1.2. Interpretation 3: If there are indications that the reality has changed significantly since planning took place, the SDM should be adapted to the current reality R'. Go to step C-1.3. Interpretation 4: If there are indications that the reality has changed since planning took place but the SDM is still valid (i.e. the changes from R to R' can be reproduced by readjusting parameter values), a re-planning should be conducted. Go to step C-1.4.	P-1 or P-2
C-1.2 (optional)	If step C-1.1 has shown that the SDM has never been valid, SDM development has to be corrected (cf. step U-1.1 in Table 15).	C-1.1 (b)
C-1.3 (optional)	If step C-1.1 has shown that the SDM is no longer valid, SDM development has to be redone (cf. step U-1.1 in Table 15).	C-1.1 (b)
C-1.4 (optional)	If step C-1.1 has shown that the reality has changed since planning took place but the SDM is still valid, a re-planning should be made (cf. step P-1.1 in Table 19 or step P-2.1 in Table 21).	C-1.1 (b)

Table 23: Sequence of C-1 steps

Addressed Purpose	Prerequisite	Induced Learning
C-1	C-1.1	Learning by validating planning (L-7)

Table 24: Induced learning by addressing purpose C-1

7.3.4.3.2 Benchmarking (C-2)

Simulation results of a SDM can be used as a benchmark, it can be used as an instrument that checks whether the behaviour of the real system deviates from the simulated (prescriptive) behaviour. This purpose is typical for static quantitative models (cf. for example [BLW97] or [BEL+97]).

The sequence of possible steps to address purpose C-2 is similar to that presented in Table 23. The consequences of a discrepancy between simulated

and measured data are, however, different. In the case of purpose C-2, the goal of the simulation activity is not to check the validity of the SDM but to check whether there has been an – unintended and otherwise not observed – change in the real system. Table 25 summarises the kind of learning that is induced by addressing purpose C-2.

Addressed Purpose	Prerequisite	Induced Learning
C-2	C-1.1	Learning by benchmarking (L-8)

Table 25: Induced learning by addressing purpose C-2

A typical use of a SDM for purpose C-2 is the monitoring of process adherence during the execution of a software project. For example, a deviation of the actual number of inspections conducted, or of the number of defects found during inspection, can be an indicator that the inspection process is not followed as prescribed.

7.3.4.4 Improving

Improving the behaviour of a system requires a change in the system structure, or in parameter values. On the other hand, not any change in the system structure or parameter automatically implies improved behaviour. The success of a change can only be proven by empirical evaluation, i.e. through measurement.

Often, however, empirical evaluation is expensive and risky. By evaluating intended changes in a virtual reality possible consequences – also unexpected or even unintended – can be better understood, and thus put the decision of whether to try an empirical evaluation on safer ground. Seen from a more constructive perspective, SDMs can be used for systematically develop and evaluate improvement suggestions in a virtual (laboratory-like) setting.

7.3.4.4.1 Exploration of Improvement Opportunities (I-1)

Similar to systematic experiments in the real world, a SDM can be used to investigate whether changes in model parameters or model structure improve model behaviour with respect to SDM Goal Dimension *Dynamic Focus*. In order to do so, proposed changes of the real system are implemented in the SDM and then compared to the baseline behaviour. If several improvements are suggested the one with the highest impact can be identified. Also, the effect of combining several improvement suggestions can be analysed.[36]

36 A detailed description of how to use a SDM for exploring the impact of process changes on product quality can be found in [PfB00].

Table 26 lists possible sequences of steps that have to be conducted in order to address purpose I-1 properly, provides a brief characterisation of each step, and lists for each step the necessary prerequisites. Table 27 summarises the kind of learning that is induced by addressing purpose I-1.

I-1 steps	Characterisation	Prerequisite
I-1.1	The SDM has been changed according to an improvement proposal. Then simulations are run with the same starting conditions and input data as the baseline (produced as described in steps P-1.1 in Table 19 or P-2.1 in Table 21). If more than one improvement proposal is made, the bast alternative is chosen.	P-1; P-2
I-1.2	The simulation results resulting from the best improvement suggestion are compared to the baseline. If the results of the changed model are better than the baseline, then the related improvement suggestions are a candidate for implementation. Otherwise, the improvement suggestion is rejected.	I-1.1

Table 26: Sequence of I-1 steps

Addressed Purpose	Prerequisite	Induced Learning
I-1	I-1.2	Learning by exploration (L-9)

Table 27: Induced learning by addressing purpose I-1

A typical use of SDMs for purpose I-1 is comparison of effects of process changes (e.g. introduction of new technology) on product quality, project duration, and effort consumption.

7.3.4.4.2 Improvement of Current Reality (I-2)

The improvement of the current reality comprises two steps. First, one or more improvement suggestions have to be evaluated in the virtual reality of the SDM. Then, given that at least one improvement suggestion performs better than the baseline, implementation and evaluation of the improvement suggestions in the real world – typically in the scope of a pilot project – can be conducted. Both, the collection of data for the (empirical) baseline, as well as the collection of empirical data for evaluating the improvement suggestion requires the performance of a related measurement programme.

Table 28 lists possible sequences of steps that have to be conducted in order to address purpose I-2 properly, provides a brief characterisation of each step, and lists for each step the necessary prerequisites. Table 29 summarises the kind of learning that is induced by addressing purpose I-2.

I-2 steps	Characterisation	Prerequisite
I-2.1	The improvement suggestion that has proven to improve the simulated system behaviour is implemented in the real world.	I-1
I-2.2	The implementation of the improvement suggestion in the real world and the comparison of induced behaviour in the changed system S' to the baseline behaviour of the unchanged system S can have two outcomes: Case 1: Behaviour of system S' is better than the baseline behaviour the system S. Case 2: Behaviour of system S' is not better than the baseline behaviour. In case 1, everything is ok, the change of system S was successful, i.e. a real improvement. In case 2, before the improvement suggestion is discarded, it should first be checked whether the implementation was done correctly (cf. steps related to purpose C-2). If this is the case, obviously the SDM that was used to explore the improvement suggestion is not valid and should be corrected (cf. steps related to purpose L-1).	I-2.1

Table 28: Sequence of I-2 steps

Addressed Purpose	Prerequisite	Induced Learning
I-2	I-2.2	Learning by empirical evaluation (L-10)

Table 29: Induced learning by addressing purpose I-2

7.3.5 Environment

The SDM Goal Definition dimension Environment has exactly the same meaning as the slot <environment> in the GQM goal definition template. As in the GQM case, the context information provided in *Environment* dimension is only of relevance for the packaging and storage of a SDM in the organisations experience base. The provision of context information for packaging is an essential prerequisite for reuse and organisational learning.

Typical examples of context information include characteristics of the software organisation and projects to which the modelling and simulation activities relate, i.e. project size, product domain, location, management culture, standards in place, organisational maturity, etc.

8 Integration of SD Models with Static SE Models

In Section 1.3, the claim was made that SDMs, which are dynamic white-box and black-box models, smoothly integrate information represented by static white-box models, i.e. DPMs, and static black-box models, i.e. QMs. In order to substantiate this clam it is necessary to show how the information contained in a DPM is mapped to the white-box component of a SDM, i.e. the flow graph, and how QMs are integrated into the black-box component of SDMs, i.e. the model equations.

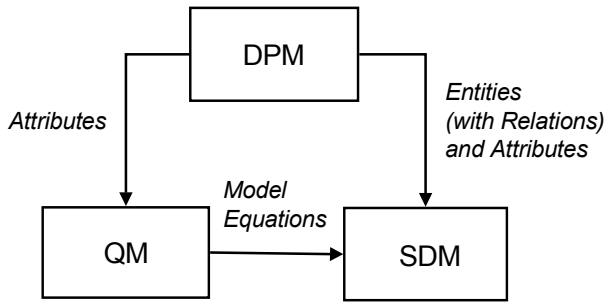


Figure 32: Relationships between static models and SDMs

Figure 32 shows which kind of information is provided from one model type to another. For example, QMs assign values to attributes of process model entities or define quantitative relationships between these attributes. It is worthwhile to note that information about the relationships between entities, which is usually expressed graphically (cf. Figure 4 on page 5), is only used for defining data collection procedures (i.e. the measurement plan) but not for defining QMs. The integration of information contained in DPMs is discussed in Section 8.1. The integration of QMs into SDMs is discussed in Section 8.2. Section 8.3 provides an example that illustrates how DPMs and QMs are integrated into a SDM.

8.1 Representation of DPM Elements in SDMs

In order to integrate the descriptive information about software processes, as captured by a DPM, in a SDM a mapping has to be defined that specifies how entities, relationships between entities, and attributes are represented in a flow graph. Table 30 defines such a mapping. It specifies how entities, i.e. processes (activities), products (artefacts), and resources (tools, roles, and actors), relationships between entities, and attributes of entities and their relationships can be represented in a SDM flow graph (cf. Figure 24 on page 50 for schematic conventions).

DPM Element		SDM Flow Graph Representation
Entity	Actor	Level
	Role	Level
	Tool	Level
	Artefact	Level
	Activity	Rate
Relationship	between Activity and Artefact	Flow of quantities
	All other relationships	Information link
Attribute	of an Entity	Level, auxiliary or constant
	of a Relationship between Entities	Level, auxiliary or constant

Table 30: Mapping of DPM elements to SDM flow graph representation

Activities are represented as rates in the SDM flow graph. All other entities are represented as levels. Relationships between activities and artefacts, i.e. relations "is consumed by" and "produces" (cf. Figure 4), are represented as flows of quantities. All other relationships are represented by information links. Attributes of both entities and relationships of entities are represented by levels, auxiliaries or constants. If the attribute is variable, the decision of whether it should be modelled as a level or an auxiliary, depends on the source of change. If it is model internal (endogenous) the attribute should be modelled as a level. If it is model external (endogenous), i.e. it is an input parameter, it should be modelled as an auxiliary.

8.2 Integration of QMs into SDMs

The possible types of QM models are typically used in a SDM as follows:

- Descriptive QMs (DQMs) are used to define start conditions of the model, i.e. initial values of levels, and to define input data (auxiliaries) and model parameters (constants).
- Predictive QMs (PQMs) are used to define implicit management decision rules (rate equations).
- Evaluation QMs (EQMs) are used to define explicit management decision rules (rate equations).

8.3 Example SDM Integrating a DPM and QMs

The following example of a simplified design inspection process will be used to illustrate the mappings of DPM elements and QMs to SDMs.

8.3.1 Descriptive Process Model

Figure 33 shows a simplified product flow of a design process with inspection, consisting of two activities, i.e. design and inspection, and three arte-

facts (products), i.e. requirements specification, design document, and inspection report with list of defects. The connection between artefact "inspection report" and activity "design" indicates that defects found during inspection have to be reworked, i.e. a re-design is triggered.

The starting and stopping of an activity is controlled by entry and exit criteria that typically are defined based on status information associated with the input and output products. For example, the entry criterion of activity "design" might be defined based on (1) the number of requirements not yet implemented, and (2) the number of defects not yet reworked. Similarly, the entry criterion of activity "inspection" might be based on (1) the completeness of the design documents and (2) the number or severity of defects found during the previous inspection. The latter criterion answers the question whether a re-inspection should be performed.

Each artefact has certain attributes. Some of them are shown in Figure 33. For example, attributes of the artefact "requirements specification" might provide information about the status of the document (e.g., available, under work, complete), the size of the documents, or the number of requirements specified in the document. Similar attributes might be associated with artefacts "design document" and "inspection report", but instead of counting the number of requirements, the number of defects injected (or contained) in the design documents, and the number of defects detected during activity "inspection" might be counted.

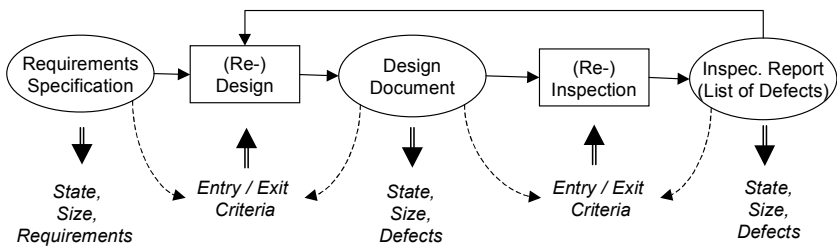


Figure 33: Simplified DPM of a design process with inspection

8.3.2 Quantitative Models

Quantitative models related to the design process with inspection, as presented in the previous sub-section, are typically used to measure or estimates attributes like "size", "number of requirements" and "number of defects", and to define decision rules, e.g., to specify whether a re-inspection has to be performed.

An example predictive model estimating the number of defects detected per inspection might have the following form:

(1) *Defects_detected=inspection_effectiveness*defects_contained_in_desing_document*

with

(2) *inspection_effectiveness=f(inspection_effort, design_doc_size)*

Equation (1) expresses the assumption that the number of defects found during an inspection depends on a portion of the total number of defects contained in the design documents. The size of the portion is defined by inspection effectiveness. Equation (2) then defines the estimator of inspection effectiveness as a function of the effort used for conducting inspections and the size of the design documents inspected (per inspection).

An example evaluation model specifying whether a re-inspection has to be conducted might be expressed in terms of a decision rule as in Equation (3):

(3) if defects_detected (per inspection) > defect_limit re-inspection ="yes"

then

re-inspection ="yes"

else

re-inspection = "no"

8.3.3 System Dynamics Model

In Section Section 8.3.1, a simplified DPM of a design process with inspection, and in Section Section 8.3.2, two of the related QMs were presented. In the following, the mapping of the DPM to a SDM flow graph, and the integration of the related QMs into the set of SDM equations is sketched.

Figure 34 shows an excerpt of the SDM flow graph. View 1 of the flow graph represents the product flow as described by the DPM in Figure 33. View 2 represents the control flow, i.e. the entry/exit criteria, defined through status attributes of the artefacts[37]. View 3 represents the defect co-flow, as specified through the related attributes of the artefacts "design document" (des_doc) and "inspection report" (des_insp_doc). The complete mapping from DPM to SDM is summarised in Table 31.

37 Note that in order to save space, in Figure 34, only a subset of the interface to the variables defined in view 2 but not the view itself is represented.

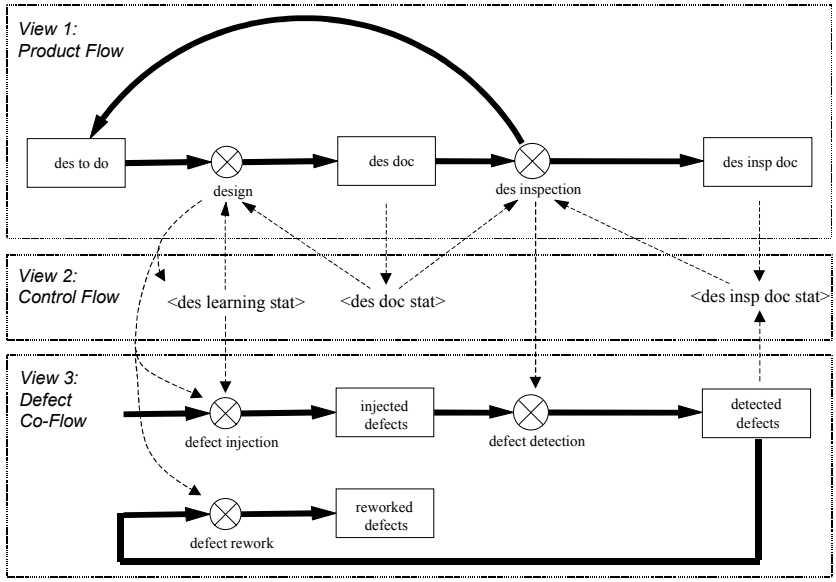


Figure 34: SDM flow graph (extract) of a design process with inspection

DPM Element			**SDM Flow Graph Representation**
Entity	Actor	n/a	--
	Role	n/a	--
	Tool	n/a	--
	Artefact	Requirements Specification	Level: des_to_do
		Design Document	Level: des_doc
		Inspection Report	Level: des_insp_doc
	Activity	(Re-)Design	Rate: design
		(Re-)Inspection	Rate: inspection
Relationship	betw. Activity and Artefact	Requirements Specification-> (Re-) Design	Flow of quantities: design_to_do -> design
		(Re-)Design -> Design Documents	Flow of quantities: design → des_doc
		Design Documents -> (Re-) Inspection	Flow of quantities: des_doc→ inspection
		(Re-)Inspection -> Inspection Report	Flow of quantities: inspection→ des_insp_doc
		Inspection Report -> Re-)Inspection	Flow of quantities: inspection→ des_to_do
	all other relation-ships	n/a	--
Attribute	of an Entity	Requirements Specification: State	Level: des_to_do
		Requirements Specification: Size	Level: des_to_do
		Design Document: State	Level: des_doc_stat
		Design Document: Size	Level: des_doc
		Design Document: Defects (contained)	Level: injected_defects - reworked_defects
		Inspection Report: State	Level: des_insp_doc_stat
		Inspection Report: Size	Level: des_insp_doc
		Inspection Report: Defects (contained)	Level: detected_defects
	of a Relationship between Entities	n/a	--

Table 31: Mapping from DPM to SDM (example)

The complete set of model equations constituting the SDM are presented below, ordered according to level equations, rate equations, and model parameters (constants).

Level equations:

a) Artefacts:

des to do= INTEG (des inspection-design, planned des doc size) -- pages
des doc = INTEG(design-des inspection,0) -- pages
des insp doc = INTEG(des inspection*des insp doc factor,0) – pages

b) Attributes:

des defects detected = INTEG(des defects detection-des defects rework,0) -- defect
des defects generated = INTEG(des defects generation,0) -- defects
des defects reworked = INTEG(des defects rework,0) -- defects
des defects undetected = INTEG(des defects generation-des defects detection,0) -- defects
des doc stat = INTEG(des doc stat change,0) -- no unit
~ status 0: non_exist
status 1: incomplete
status 2: complete
des insp doc stat = INTEG(des insp doc stat change,0) -- no unit
~ status 0: non_exist
status 1: incomplete
status 2: complete
status 3: complete_again
des learning stat = INTEG(des learning,0) -- no unit

Note: The level equations are automatically generated from the flow graph.

Rate equations:

design = IF THEN ELSE((des doc stat=1),MIN(des to do,des rate*MAX(des learning stat,1)),0) -- pages/day
des inspection = IF THEN ELSE((des insp doc stat=1):OR:((des insp doc stat=3):AND:(des doc stat=2)),des doc,0) -- pages/day
des defects generation = design*des def gen rate*(1/MAX(1,des learning stat*des learning stat)) -- defects/day
des defects detection = IF THEN ELSE((des insp doc stat=1):AND:(des doc stat=2),des def detection factor*des defects undetected,0) -- defects/day
des defects rework = IF THEN ELSE((des insp doc stat>0),MIN(des defects detected,des defect rework rate*design),0) -- defects/day
des doc def flag =IF THEN ELSE(des defects detected>des defect limit,1,0) -- no unit
des doc stat change = IF THEN ELSE((des doc stat=0):AND:(entry design flag=1),1,IF THEN ELSE ((des doc stat=1):AND: (des to do<=0),1,IF THEN ELSE((des doc stat=2):AND:(des to do>0),-1,0))) -- no unit
des defect rework rate = des def gen rate -- defects/day
des learning = design/planned des doc size -- no unit
des insp doc stat change = IF THEN ELSE((des insp doc stat=0):AND:(des doc stat=2),1,IF THEN ELSE((des insp doc stat=1):AND:(des doc stat=2),1,IF THEN ELSE((des insp doc stat=2):AND:(des doc def flag=1),1,IF THEN ELSE ((des insp doc stat=3):AND:(des doc def flag=0):AND:(des doc stat=2),-2,0)))) -- no unit

Model parameters (constants):

des rate = 10 -- pages/day
des def detection factor = 0.8 -- no unit
des def gen rate = 0.1 -- defects/page
des insp doc factor = 0.1 -- no unit
entry design flag = 1 -- no unit

Input data:

des defect limit = 9 -- defects/page
planned des doc size = 200 – pages

The following QMs were integrated into the set of SDM equations:

DQM (descriptive QM):

des rate = 10 -- pages/day
des def detection factor = 0.8 -- no unit
des def gen rate = 0.1 -- defects/page
des insp doc factor = 0.1 -- no unit

Note: In the example SDM, variable *des_def_detection_factor* replaces variable *inspection_effectiveness*. This is based on the assumption that inspection effort and size of the design documents inspected are constant (on average).

PQM (predictive QM):

des defects detection = IF THEN ELSE((des insp doc stat=1):AND:(des doc stat=2),des def detection factor*des defects undetected,0) -- defects/day

EQM (evaluation QM):

design = IF THEN ELSE((des doc stat=1),MIN(des to do,des rate*MAX(des learning stat,1)),0) -- pages/day
des inspection = IF THEN ELSE((des insp doc stat=1):OR:((des insp doc stat=3):AND:(des doc stat=2)),des doc,0) -- pages/day
des doc def flag =IF THEN ELSE(des defects detected>des defect limit,1,0) -- no unit
des doc stat change = IF THEN ELSE((des doc stat=0):AND:(entry design flag=1),1,IF THEN ELSE ((des doc stat=1):AND: (des to do<=0),1,IF THEN ELSE((des doc stat=2):AND:(des to do>0),-1,0))) -- no unit
des insp doc stat change = IF THEN ELSE((des insp doc stat=0):AND:(des doc stat=2),1,IF THEN ELSE((des insp doc stat=1):AND:(des doc stat=2),1,IF THEN ELSE((des insp doc stat=2):AND:(des doc def flag=1),1,IF THEN ELSE ((des insp doc stat=3):AND:(des doc def flag=0):AND:(des doc stat=2),-2,0)))) -- no unit

When developing the SDM, two major enhancements were made as compared to the information available in the DPM and related QMs. Firstly, the defect co-flow was explicitly modelled in parallel to the product flow of the design and inspection process. Secondly, the root cause for the dynamics that eventually determines whether a re-inspection has to be conducted has been introduced, i.e. the learning curve. In the SDM, learning is represented by the level variable des_learning_stat. The level des_learning_stat can be interpreted as an attribute of the activity des_doc, because learning takes place during the (re-)design activity.[38] The effect of learning is two-fold. Firstly, it reduces the defect injection rate during design (rate variable "des_defects_generation"). Secondly, it increases design productivity (rate variable "design"). Figure 35 summarises the causal structure that generates the dynamics of the example SDM.

38 An alternative would have been to model learning, i.e. the accumulation of knowledge, as an attribute to the role "designer". However, because there was no need to introduce an explicit representation of roles, this SDM design option was descarded.

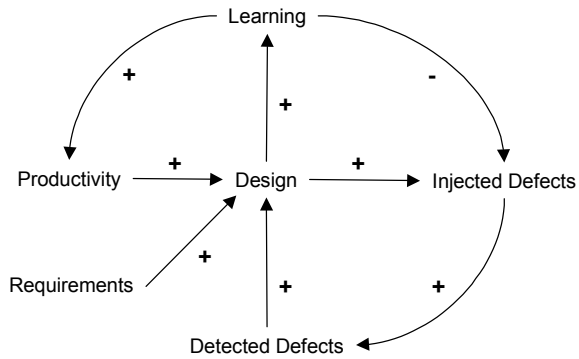


Figure 35: Causal diagram of example SDM

Figure 36 shows the results of a simulation run with the SDM. On the left-hand side, three variables related to the product flow are presented, namely the development of the design document (variable des_doc) with three rework cycles, the performance of three inspections (variable des_inspection), and the learning curve (variable des_learning_stat). On the right-hand side, three variables related to the defect co-flow are presented, namely the cumulated number of defects injected into the design document during development and rework (variable des_defects_generated), the number of detected and not yet reworked defects (variable des_defects_detected), and the cumulated number of defects reworked (variable des_defects_reworked). The impact of the cumulation of knowledge (learning curve) on both design productivity and improved work quality can be recognised in the product-flow window by the shortening of rework cycle time, and in the defect co-flow window by the reduction of defect generation during rework cycles.

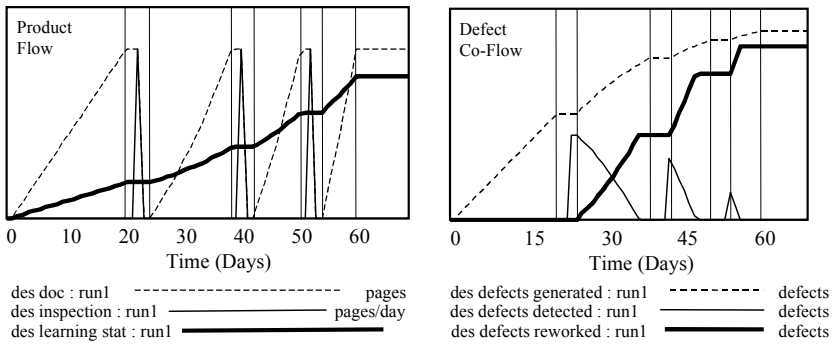


Figure 36: Simulation output of example SDM

9 Integration of SD Modelling with GQM and PM

Figure 37 gives an overview of the modelling methods that are used within the IMMoS framework, i.e. process modelling (PM), goal-oriented measurement (GQM), and System Dynamics (SD). In the following, these methods will be referred to as the IMMoS components. Figure 37 also shows the end products of the modelling methods, i.e. prescriptive process model (PPM), descriptive process model (PPM), quantitative model (QM), and System Dynamics model (SDM), as well as the relations between modelling methods and products.

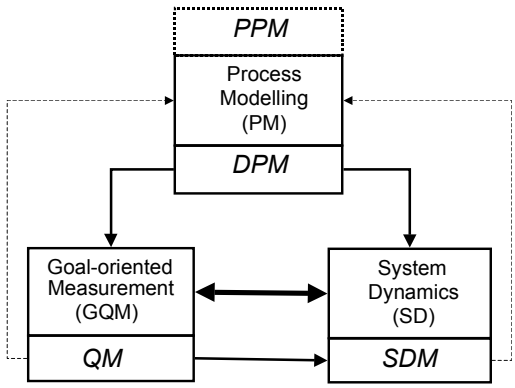


Figure 37: IMMoS components

In the following sections, the various relationships between methods of the IMMoS components, established through the mutual use of work products and end products are described in detail.[39] Special focus is put on the close relationship between System Dynamics and GQM.

9.1 Products of the IMMoS Components

To prepare for a detailed presentation of the relationships between individual IMMoS components, in this section, the most important SD, GQM, and PM products are briefly summarised.

9.1.1 SDM Development Products

Relevant work products of SD modelling and simulation are (cf. Section 6.4):

[39] Parts of this work have been published in [PfL99].

- SDM Goal Definition: specifying scope, dynamic focus, purpose, role, and environment of the SDM development project.
- SDM Reference Mode capturing the dynamic behaviour of interest.
- SDM Causal Diagram: defining the minimal network of causal relations (i.e. SDM Base Mechanisms) necessary to generate the dynamic behaviour of interest.
- Initial SDM: able to reproduce the SDM reference Mode.

Relevant end products of SD modelling and simulation are (cf. Section 6.4):

- Enhanced SDM: applied to solve the problem defined in the SDM Goal Definition.
- Analysed and interpreted simulation results: used to solve the problem defined in the SDM Goal Definition.

9.1.2 GQM Products

Relevant work products of GQM planning and measurement are (cf. [GHW95]):

- One or several measurement goals (GQM Goals): each specifying object, quality focus, purpose, viewpoint, and context of the measurement programme.
- GQM Plan: making measurement goals operational through questions, and defining the metrics and models that will help to answer these questions.
- Measurement Plan: specifying in detail all the procedures to be followed during the measurement programme.
- Measurement data.

End products of a GQM programme are the analysed and interpreted measurement data in the form of QMs. The following types of QMs exist (cf. [BDR96]):

- Descriptive QMs (DQMs), i.e. models that describe attributes of certain real-world objects quantitatively,
- Prescriptive QMs (PQMs), i.e. models that describe the relations between attributes of real-world objects and impacting factors quantitatively, or
- Evaluation QMs (EQMs), i.e. models that support decision making.

9.1.3 PM Products

The end products[40] of descriptive and prescriptive PM are descriptive process models (DPMs) and prescriptive process models (PPMs), respectively (cf. Section 1.1.2.1).

DPMs capture information about the current software development practices and organisational characteristics of the software organisation. The content of a DPM is mainly based on knowledge elicited from process experts and software development practitioners. Relevant real-world aspects are represented by entities and relations between entities. Entities are characterised through attributes. Examples of attributes are size, complexity, status, time, effort, etc. Different approaches for descriptive process modelling can be applied.

PPMs contain the same kind of information as DPMs. But the purpose of a PPM is different to that of a DPM. While DPMs are developed in order to capture current development practice, the purpose of PPMs is to specify how the development processes should be.

9.2 Relationships between SDM Development, GQM, and PM

With the information provided in the previous section, a detailed description of relationships between IMMoS components is presented below. First, a complete overview of all possible interdependencies between the three IMMoS components is illustrated, then a more detailed insight into the relations between individual products of the IMMoS components is given. Finally, a short example on the integration of SDM development with information derived from a DPM and an associated PQM is presented to demonstrate the benefits of IMMoS.

9.2.1 Overview of IMMoS Relationships

Various interdependencies between the individual modelling stages and products of SDM development, GQM and PM can be identified. An overview is sketched in Figure 38, indicating:

- Role: model developer with defined modelling goal(s).
- Modelling method: PM, GQM, SD.
- Modelling stage: descriptive and prescriptive PM, GQM planning and execution, SDM development and simulation.
- Work products of PM, GQM, and SD: measurement goals, GQM and measurement plans, SDM Reference Mode, SDM Causal Diagram (with SDM Base Mechanisms), SDM Equations (Initial SDM) with related simulation results.
- End products of PM, GQM, and SDM development: DPM, PPM, measurement data with related QM(s), SDM Equations (Enhanced SDM) with related simulation results.

[40] PM work products are not considered here. This is due to the fact that standard processes for deskriptive and descriptive PM have not yet been collected. A very general phase model (without definition of work products) can be found in [BHV97]. Refinements of this framework with lessons learned from an industrial case study were presented in [BeB00].

- Reality: software development.
- Relations (depicted as arcs) between model user, modelling stages, modelling work/end products, and reality. Arcs from products to modelling methods/stages, role or real world software development represent a "is used by" relation. Arcs from modelling methods/stages to products represent a "produces" relation. Arcs from modelling stages to modelling methods represent a "triggers" relation. Arcs from real world software development to products represent a "provides data to" relation. Arcs from real world software development to role represent a "is observed by" relation. Arcs from role to real world software development represent a "makes decision upon" relation. Arcs from role to modelling method represent a "develops model" or "uses model" relation.

When and how the products of the three modelling approaches are combined depends on the goals of the potential model user.

The PPM (grey shaded in Figure 38) is only needed to specify what an intended process change will look like. As soon as the process change is successfully implemented, the PPM automatically becomes the new DPM.

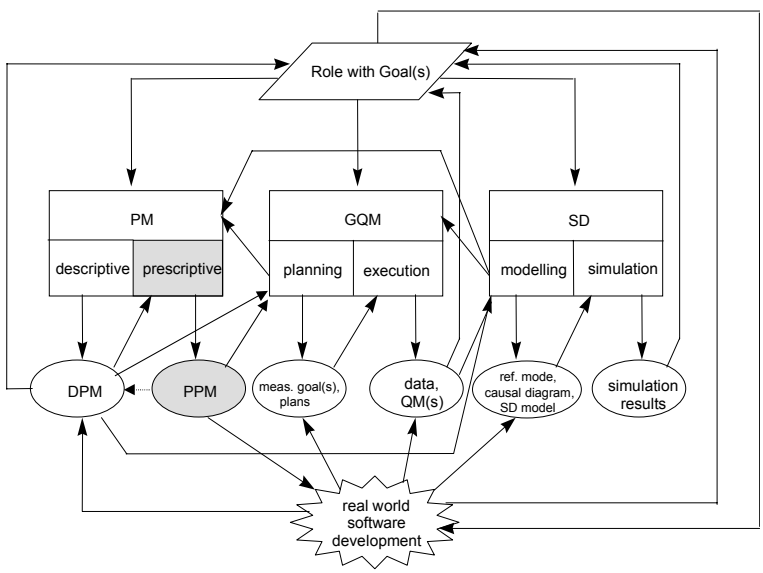


Figure 38: Interaction between IMMoS components during modelling

A complete description of the detailed relations between individual IMMoS components and their work and end products is presented in the following sub-sections.

9.2.2 Relations between GQM and PM

Figure 39 outlines the product flow between PM and GQM, only showing the end products resulting from the individual modelling activities, i.e. DPM, QM, and PPM, and their relation to the real world, which is modelled and to which the models are applied.

In order to get a qualitative understanding about the entities in the real world and their relationships, a DPM can be developed without support from any other modelling method (1). Based on information contained in the DPM, and on data collected from the real world, QMs can be developed (2). Both, development and analysis of the DPM and application of the QM to the real world may trigger proposals to change the current development process in order to improve it. The improvement suggestion will be represented in the form of a PPM, which is largely based on the existing DPM (3). In order to evaluate the PPM, the development of a QM, which uses information from both DPM and PPM, might be triggered (4). Iterations between modelling activities are depicted by dashed arcs.

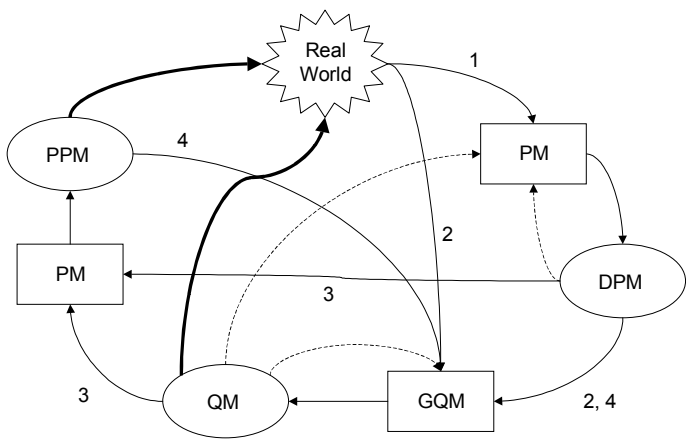


Figure 39: Relations between GQM and PM

On a more detailed level, the relations between GQM products and PM products can be summarised as follows:

Descriptive PM and GQM:

- DPM ↔ Measurement goal: Usually, the object that is subject to measurement is an entity in the DPM, and the quality focus of the measurement goal is specified by some of the entity's attributes. On the other hand, it can turn out during the specification of a measurement goal that the software processes are not yet clearly defined. In that case, a PM activity is initiated.
- DPM ↔ GQM plan: The definition of the questions and metrics contained in a GQM plan is usually done with the help of so-called abstrac-

tion sheets. Basically, an abstraction sheet is a means of acquiring, structuring, and documenting all the relevant information provided by participants in the measurement programme. An abstraction sheet contains information about the entities of the measurement object with its associated attributes representing the quality focus (as specified by the measurement goal), and information about factors that have an impact on the quality focus (so-called variation factors). In addition, hypotheses about the performance of the quality focus attributes and the way in which the variation factors influence the performance of the quality focus attributes are documented. Based on this information, for each measurement goal, a set of questions, metrics, and models can easily be defined (for details cf. [GHW95]). To a large extent, the set of entities and attributes gathered in abstraction sheets is determined by the information contained in the organisation-specific DPM. Again, it can turn out during the development of a GQM plan that the software processes are not yet defined clearly enough. In that case, a PM activity is initiated.

- DPM → Measurement plan: When, how and by whom measures that have been defined in the GQM plan are collected must be specified in accordance with the development process (as represented by the DPM). An instructive example that describes how a DPM is used for defining the measurement plan can be found in Bröckers et al. [BDT96].

GQM and prescriptive PM:

- QM → PPM: The results from the application of a QM can trigger ideas for improving the development process. The proposed process changes are represented in a PPM.
- PPM (& DPM) → QM: Evaluation of the validity of the improvement suggestions represented in the PPM might trigger the development of a QM, i.e. a EQM.

9.2.3 Relations between SDM Development and PM

The relations between SD modelling products and PM products are as follows (cf. Figure 40):

- Indirect relation via measurement (cf. Section 9.2.2).
- Direct relation through usage of qualitative process information from a DPM as input for the definition of reference modes, base mechanisms, causal diagrams, and by mapping the relevant parts of the control and product flow to the SDM flow graph (Step 2 in Figure 40)
- Again, as with QMs, the results of SDM simulation might be used for prescriptive process modelling (Step 3 in Figure 40). Note that no input from PPMs to SDM development is necessary because potential process changes are directly implemented in the SDM flow graph and evaluated through simulation.

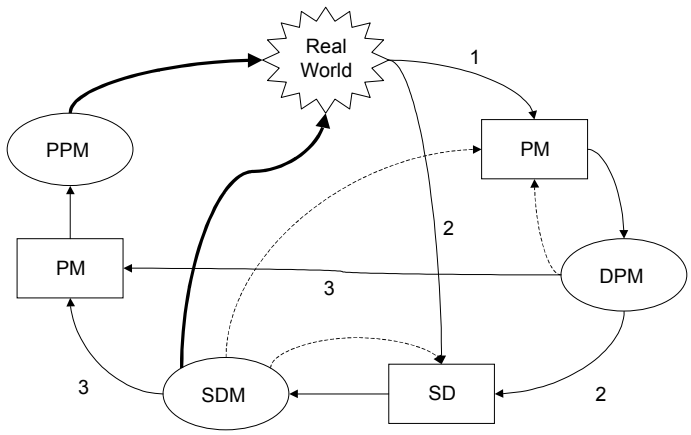


Figure 40: Relations between SDM development and PM

9.2.4 Relations between SDM Development and GQM

Figure 41 outlines the product flow between SDM development, PM and GQM, taking the viewpoint of SDM development, i.e. relations between PM and GQM without direct involvement of SDM development are not shown. Again, Figure 41 does only show the end products resulting from the individual modelling activities, i.e. DPM, QM, SDM and PPM, and their relation to the real world, which is modelled and to which the models are applied. The relation between SDM development and GQM, which is in the focus of this sub-section (cf. grey areas), is very tight and involves exchange of information at several modelling steps, involving various intermediate work products.[41]

[41] Similar to the relation between GQM and PM, where PM end products are used at several stages of the GQM process, GQM end products (i.e. QMs) are used in several stages of the SDM development process.

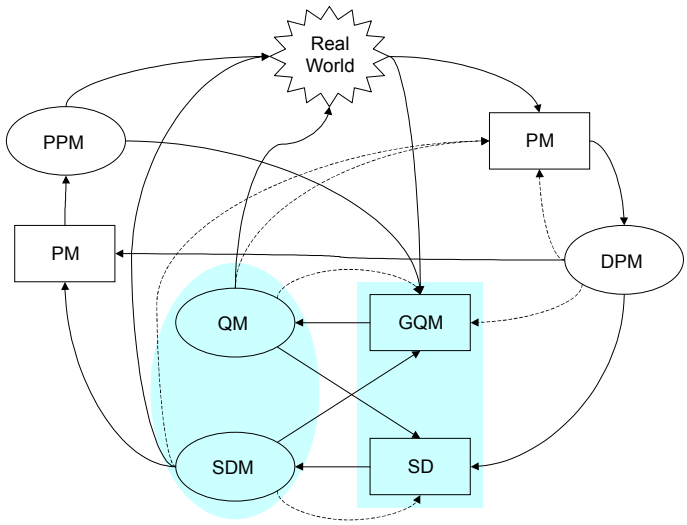


Figure 41: Relations between SDM development, GQM and PM

The relations between SD modelling products and GQM products are as follows:

- SDM Reference Mode → Measurement goal (and GQM plan): If a reference mode shall be based on empirical data and this data is not yet readily available, an adequate measurement goal has to be defined. Formally, five aspects have to be specified for a complete definition of a measurement goal: object, quality focus, purpose, viewpoint, and context. The SDM Reference Mode determines three of the five aspects. The measurement object is defined by the real-world entity under consideration, the measurement quality focus is defined by the attributes which describe the reference behaviour of the entity under consideration, the measurement purpose is defined as monitoring (i.e., quantitative characterisation of attributes over time). The information about the entities and attributes used to define the reference mode will also be the starting point for the development of the GQM plan.
- SDM Base Mechanisms / SDM Causal Diagram → Measurement goal (and GQM plan): SDM Base Mechanisms define cause-effect relationships that are supposed to generate the reference behaviour. If the identification of cause-effect relationships shall be based on empirical evidence, appropriate measurement has to be conducted. Therefore, one or more measurement goals with measurement purpose control have to be defined. Then, the GQM plans based on these measurement goals will capture the attributes describing effects in the quality focus section, and attributes describing causes in the variation factors section. Statistical correlation analysis might be applied.
- SDM Equations → Measurement goal (and GQM plan): SDM Equations define quantitative functional relationships between model variables. If the identification of functional relationships shall be based on empirical

evidence, appropriate measurement has to be conducted. Therefore, one or more measurement goals with measurement purpose control or prediction have to be defined. Then, the GQM plans based on these measurement goals will capture the dependent variable in the quality focus section, and the independent variables in the variation factors section. Statistical modelling, e.g. regression analysis, and data mining techniques might be applied.

- GQM plan → SDM Base Mechanisms / SDM Causal Diagram: When defining a GQM plan, there is a chance that during the process of knowledge acquisition with abstraction sheets new quality focus attributes and associated variation factors are identified. New pairs of dependent variables (quality focus attributes) and explanatory variables (variation factors) can be used to modify or extend the set of SDM Base Mechanisms.
- Measurement results (DQM) → SDM Reference Mode: If adequate measurement data is available, it can be used to define the reference mode through a DQM.
- Measurement results (DQM, PQM) → SDM Base Mechanisms and SDM Model Equations: Measurement results that come out of an adequately defined measurement programme can help to identify and define cause-effect relationships qualitatively (in terms of SDM Base Mechanisms) and quantitatively (in terms of SDM Equations).

Figure 42 summarises the relationships between SDM development and GQM, including the SDM user and application of the SDM for problem solution. While the bold arcs indicate the general flow of work, the plain arcs represent the relationships between SD and GQM work products. Dashed arcs indicate the use of information from DPMs.

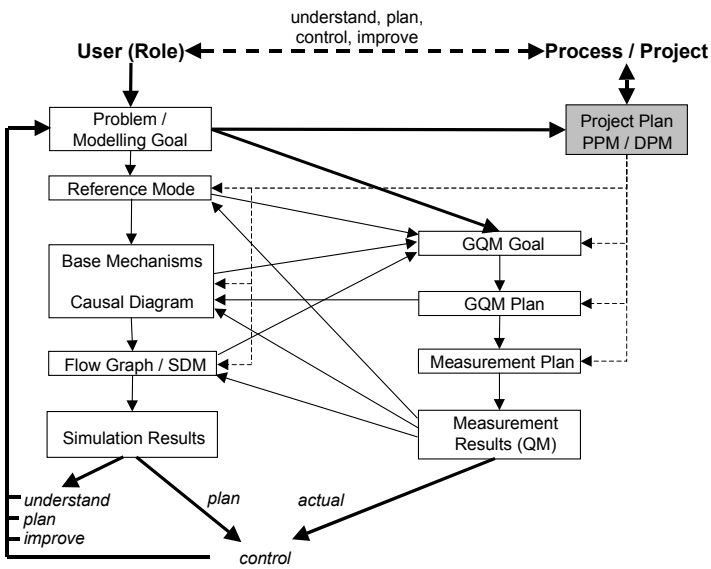


Figure 42: IMMoS application example scenario

9.3 IMMoS Application Example Scenario

To further illustrate IMMoS, in this subsection an example scenario is presented explaining how a hypothetical SD modelling and simulation activity can be linked to process modelling (PM) and goal-oriented measurement (GQM).[42] Adopting the perspective of a SD modelling project, the example scenario begins with the description of the situation at project start, then it concentrates on possible relations between SD modelling and simulation, PM, and GQM. The scenario evolves in four steps:

(1) Initialisation of the SDM development project: Motivation of the SDM development project, and description of the starting situation in terms of roles involved, problem statement, and modelling goals.

(2) Acquisition of qualitative information: SDM development triggers PM and exploits the resulting DPM. [Affected relations between IMMoS components: SD → PM → SD]

(3) Acquisition of quantitative information: SDM development triggers GQM-based measurement and exploits resulting DQMs and PQMs. [Affected relations between IMMoS components: SD → GQM → PM → GQM → SD]

(4) Application of SDM: a) SD simulation results serve for process analysis and subsequent identification of potential improvements, eventually resulting in a PPM that specifies the candidate process improvements; b) Evaluation of the implemented process changes via EQMs and use of the (re-validated) SDM for project planning and control. [Affected relations between IMMoS components: SD → PM and GQM → SD]

These four steps are worked out in more detail in the four subsections below. In each of the steps, SD, PM, and GQM-related work products and end products are used as described in Section 9.1.

9.3.1 Initialisation of SDM Development Project

Assume a Software Engineering Process Group (SEPG), consisting of process engineers and technical project leaders, is interested in achieving a better understanding of the trade-off effects between development time and software product quality. Moreover, if possible, the team would like to improve the project performance as a result of an adequate process change. If the process change is successful, the project leaders would like to have a simulation model that supports project planning and control.

In summary, the SEPG has four goals: learning (understand process/project dynamics), improving (change process to achieve better project perfor-

[42] The value of SDMs for defining measures and, in addition, for demonstrating the usefulness of establishing a measurement system in a software organisation has been explained based on the example of the PSIM model in [Pfa95].

mance), planning and controlling (of project performance). The team expects that a SD simulation model can help to support the achievement of their goals.

9.3.2 Acquisition of Qualitative Information

As a first step towards SD model building, a problem definition is provided, restricting the abstraction level of the model structure to those development activities and associated work products that play a key role for effort and time consumption, as well as for the determination of product quality. To identify the most relevant activities, artefacts, resources, roles, etc., a DPM that describes the static structure of the current development process should be consulted. The DPM either has to be developed, or - if already existing - can be extracted from the organisation's experience base.

As soon as the DPM is available, it is used to help identify the most important state variables of the SDM. Typical state variables are countable attributes of artefacts (like size, or number of defects contained), activities (like duration), relations between artefacts and activities (like number of defects generated and number of defects detected), and agents (like number), etc. A subset of these state variables will be used to define the SDM's reference mode. Additionally, the information contained in the DPM can be the starting point for discussing possible causal relationships between model variables.

Figure 43 shows an extract from the SDM's flow graph that is based on the information in the DPM and additional input from software development experts.

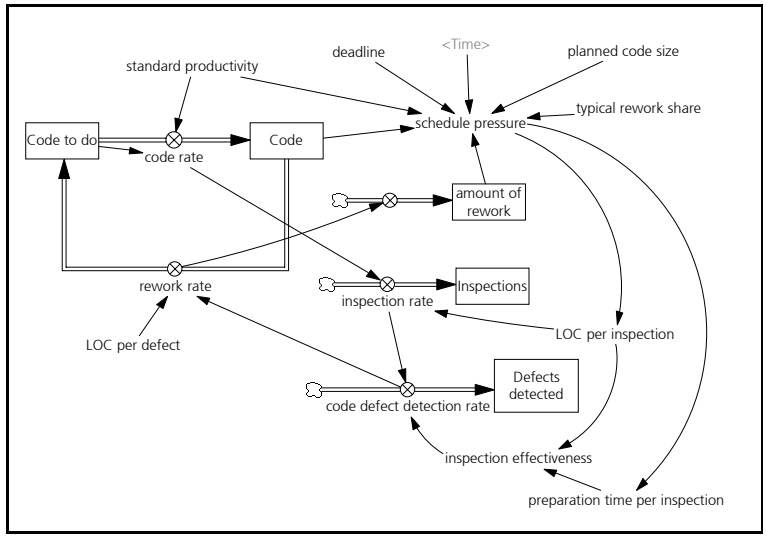


Figure 43: Extract from a view of the SDM's flow graph (implementation phase)

9.3.3 Acquisition of Quantitative Information

The reference mode is needed to sketch the most essential surface behaviour of the development process, i.e. it is a description of the dynamic behaviour of key variables that characterise the system to be modelled. A reference mode might contain, for example, the trajectories of defects detected, effort consumed, change requests, etc. over project time. The reference mode can be based on expert opinion or on empirical data. The SEPG team decides to use empirical data from typical projects. The respective data might again either be extracted from an existing experience base, or gained from a newly started and specifically tailored measurement programme. The measurement goals of such a programme are determined by the needs of SDM development. The generic specification of a measurement goal for defining the reference behaviour of a state variable is outlined in Table 32.

The complete specification of what will be measured is then worked out in the form of a GQM plan. The execution of the measurement programme is defined by the measurement plan which is derived from the GQM plan and additional information taken from the DPM and its instances (i.e. project plans). The measurement programme will result in a set of DQMs that are used to define the reference mode.

Analyse	process (or product)	[object]
with respect to	<state variable> (e.g. manpower allocation)	[quality focus]
for the purpose of	monitoring	[purpose]
from the viewpoint	SEPG team	[viewpoint]
in the context of	typical projects of the software organisation	[environment]

Table 32: Measurement goal specification template for SD Reference Mode definition

Empirical data can also play a crucial role for the identification of causal relationships. For, example, statistical analysis based on measurement data can be applied to define the SDM variable inspection effectiveness in Figure 43. The inspection effectiveness together with the number of inspections determines the number of defects that are detected during a week (variable code defect detection rate), and thus the weekly amount of rework to do (variable rework rate). The weekly number of inspections is represented by the variable inspection rate. The (average) number of defects found in a code inspection is assumed to be a function of the variables LOC per inspection and preparation time per inspection. Such a function is equivalent to a PQM. The exact definition of this PQM can be derived through regression analysis. A fully documented example of such a measurement-based analysis can be found in Briand et al. [BLW97]. The authors describe how a measurement programme was specified, and how the statistical analyses were conducted to derive a PQM on inspection effectiveness (with effectiveness being defined as the number of detected defects divided by the size of the inspected artefact). As a result of their analyses the authors showed that there is a statistically significant exponential relation between the inspected

document size, the effort spent on inspection preparation, and the resulting inspection effectiveness.

Of course, similar measurement-based analyses can be conducted for other SDM variables. Sometimes, however, it is very time and effort consuming, or even impossible to carry out an appropriate empirical analysis. In these cases, one can only rely on expert opinion. And often, due to lack of adequate data, empirical analysis cannot result in a precise definition of SDM equations, but still provide enough evidence for an association or correlation between certain model variables. This more qualitative information is still useful when developing causal diagrams.

9.3.4 Application of SDM

In the course of building the SDM, the SEPG has already gained much new insight into those aspects of the development process that influence project duration and product quality. Gaming with the simulation model allows the team to better understand the dynamic behaviour patterns and to conduct a systematic analysis of the trade-off between project duration and product quality. Hence, model building and simulation satisfy the goal of learning.

Having the SDM at hand, the analysis and interpretation of the simulation outcomes might trigger several process improvement suggestions (goal: improving). To decrease the risk of losing money with unsuccessful pilot projects, the improvement suggestions should first be investigated by implementing the intended process changes in the SDM and comparing simulation results. If these analyses confirm the improvement suggestions, this might even be an additional argument to convince management that the suggested process change should be piloted in real world software development.

Before a process change is implemented, it should be specified in a PPM, and, after implementation, it must be evaluated. This can only be done by using an EQM that compares the performance of the new process to the baseline. If the improvement is actually confirmed this will also support the validity of the SDM, and the project leaders might use it for project planning, risk management and project control (goals: planning and controlling).

Process engineers and project leaders might even use the SDM as a tool for process or project benchmarking. Comparing the process/project behaviour predicted by the SDM with control data derived from an appropriate measurement programme can do this (goal: controlling). Assuming that the SDM is sufficiently valid, a big difference between the model predictions and the measurement data indicates an undetected alteration of the real system. An example of such an undetected alteration might be a gradually increasing loss of conformance with inspection guidelines. The identification of such decay of the inspection process might result in a reinforcement of the guidelines and prevent future loss of money (i.e. during maintenance).

Part IV: Validation

10 Validation of the IMMoS Approach

In Section 4.9.2 the following research hypotheses were formulated:

- H_1: SDM development with IMMoS is at least as effective as SDM development without IMMoS.
- H_2: SDM development with IMMoS is more efficient than SDM development without IMMoS.

The concept of effectiveness is related to the suitability of a developed SDM. Suitability is measured as the degree to which a developed SDM fulfils its purpose from the point of view of the model user. The validity of hypothesis H_1 is evaluated by comparing the suitability of SDMs developed with IMMoS (innovation) to the suitability of SDMs developed without IMMoS (baseline). The expectation is that SDMs developed with IMMoS are at least as suitable as SDMs developed without IMMoS.

The concept of efficiency is related to the consumption of resources during the development of a SDM. Resource consumption is measured in terms of effort and time needed for the modelling activities. The validity of hypothesis H_2 is evaluated by comparing the relative[43] resource consumption of SDMs developed with IMMoS (innovation) to the relative resource consumption of SDMs developed without IMMoS (baseline). For the evaluation, only suitable SDMs will be compared, because it would not be of much use to show that SDM development with IMMoS is faster and cheaper than without IMMoS, if the resulting SDMs would not fulfil their purposes. In that respect, the validity of hypothesis H_1 can be seen a necessary condition for the validity of hypothesis H_2. The expectation is that the development of SDMs with IMMoS needs less time and effort than the development of SDMs without IMMoS.

The overall approach to the validation of IMMoS is summarised in Figure 44. For the evaluation of the research hypotheses, two industrial case studies and one controlled experiment were conducted. The following models were used to validate the IMMoS approach:

- PSIM: Project/Process Simulator (developed without IMMoS / baseline)
- RESIM: Requirements Engineering Simulator (developed with IMMoS / innovation)
- GENSIM: Generic Simulator (developed with IMMoS / innovation)

[43] In order to eliminate the impact of SDM size on resource consumption, duration and effort numbers were divided by SDM size.

The PSIM model was developed during a first case study conducted at Siemens in collaboration with the corporate research division. The customer organisation was a department in the telecommunication branch. Because IMMoS was not yet available during the PSIM case study, this project formed the baseline for the validation of IMMoS. Details about the PSIM project were provided in Section 4.

The RESIM model was developed during the second case study conducted at Siemens, again in collaboration with the corporate research division. In this case, the customer organisation was a division in the automotive branch. For the development of the RESIM model IMMoS was applied. Details about the RESIM project will be provided in Section 11.

The GENSIM model was developed at Fraunhofer IESE. It was used in lectures at the University of Kaiserslautern. For the development of the GENSIM model IMMoS was applied. Details about the GENSIM project will be provided in Section 12.

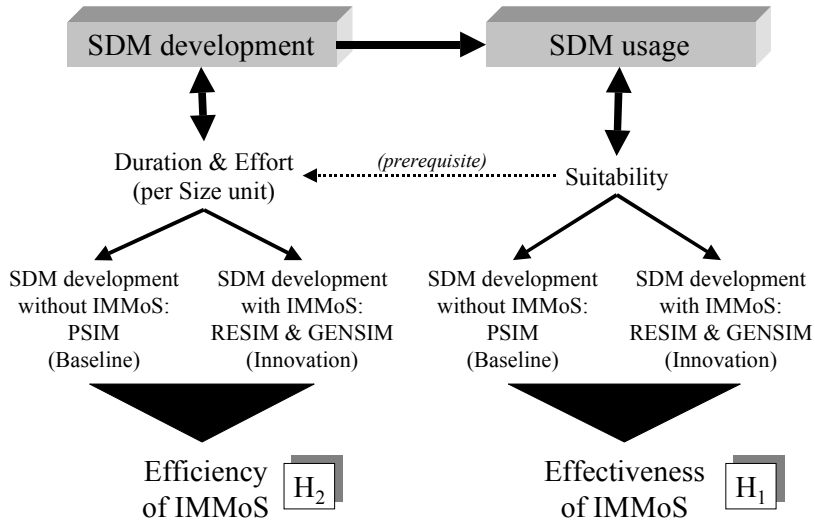


Figure 44: IMMoS validation approach

The following sub-sections sketch the approaches that were chosen to evaluate the research hypotheses and summarise the evaluation results. Details about the evaluation of hypothesis H_1 are provided in Section 13. Details about the evaluation of hypothesis H_2 are provided in Section 14.

10.1 Evaluation of Hypothesis H_1 (Effectiveness)

In order to evaluate hypothesis H_1 it is necessary to compare the effectiveness of SDM development with IMMoS (innovation) to SDM development

without IMMoS (baseline). As the formulation of hypothesis H_1 states, it is not expected that SDM development with IMMoS is generally *more* effective than SDM development without IMMoS. There is no reason to assume that a SDM developed without IMMoS cannot fulfil its purpose from the point of view of the model user.

Effectiveness of SDM development (with or without IMMoS) was evaluated in terms of suitability, i.e. the fulfilment of the SDM purpose from the point of view of the SDM user. The suitability of the SDM models PSIM and RESIM was assessed within the context of industrial case studies, based on objective data (PSIM) and subjective data (PSIM and RESIM). The suitability of GENSIM was assessed based on the results of a controlled experiment with computer science students at the University of Kaiserslautern. In the controlled experiment, the effectiveness of a training session on the topic "Software Project Management" was evaluated. A pre-test-post-test control group design was applied. The treatment of the experimental group involved simulation with GENSIM, while the treatment of the control group involved the usage of a traditional project estimation model (COCOMO [Boe81]). Four variables were used to capture student performance: (1) interest in the topic of software project management, (2) knowledge about typical behaviour patterns of software development projects, (3) understanding of simple project dynamics, (4) understanding of complex project dynamics. The suitability of GENSIM was considered sufficient if the post-test performance and the performance improvement (difference between pre-test to post-test scores) of the students in the experimental group was not worse than that of the students in the control group.

Table 33 summarises the results of the evaluation. All three SDMs were considered suitable at least for one SDM user, i.e. there is no difference between SDMs developed with IMMoS to the SDM developed without IMMoS, and thus no indication that the null hypothesis associated with hypothesis H_1 could not be rejected. Hence, hypothesis H_1 is supported.

	Baseline (SDM development without IMMoS)	Innovation (SDM development with IMMoS)	
Evaluation context	Industrial case study	Industrial case study	Controlled experiment
SDM name	PSIM	RESIM	GENSIM
Evaluation criterion	SDM suitability	SDM suitability	SDM suitability
SDM user	a) Project Manager b) Process Owner	Process Consultant (Assessor)	Computer Science Students
SDM purpose	a) Planning and Control b) Improvement	Understanding	Understanding (Training)
Type of measurement data	Subjective (a and b) and objective (only a)	Subjective	Subjective and objective
Measurement results for suitability	1. Objective data: The predictive accuracy of PSIM was greater than 90%. 2. Subjective data: a) According to the Project Manager, PSIM fulfilled its purpose sufficiently well. b) The Process Owner could not use PSIM for the intended purpose.	Subjective data: According to the SDM user, RESIM fulfilled its purpose sufficiently well.	1. Objective data: Compared to a training session with COCOMO, the knowledge of students about typical behaviour patterns of software projects increased (statistical significance). With regards to the understanding of typical software project dynamics no significant difference could be observed between experimental and control group. 2. Subjective data: Compared to a training session with COCOMO, the interest of students in the topic "Software Project Management" increased (practical significance).
Interpretation of measurement results	PSIM is suitable (at least for one SDM user)	RESIM is suitable	GENSIM is suitable

Table 33: Summary of evaluation results for hypothesis H_1

10.2 Evaluation of Hypothesis H_2 (Efficiency)

In order to evaluate hypothesis H_2 it is necessary to compare the efficiency of SDM development with IMMoS (innovation) to SDM development without application of IMMoS (baseline). The result of this comparison is only useful when in both cases, i.e. innovation and baseline, the developed SDMs were suitable.

Efficiency of SDM development (with or without IMMoS) was evaluated in terms of resource consumption, i.e. calendar time (duration) and effort needed to develop the SDM. For all three models, PSIM, RESIM, and GENSIM, resource consumption was assessed based on objective data. In order to neutralise the impact of model size, time and effort data was divided by the number of level variables[44]. The number of level variables was considered an adequate measure of SDM size.

Table 34 summarises the results of the evaluation. The SDMs that were developed with IMMoS needed considerably less time and effort for their development. This is an indication that the null hypothesis associated with hypothesis H_2 could be rejected if a sufficiently large number of cases would be available for a full statistical analysis. Hence, hypothesis H_2 is supported.

	Baseline (SDM development without IMMoS)	**Innovation (SDM development with IMMoS)**	
Evaluation context	Industrial case study	Industrial case study	Preparation of controlled experiment
SDM name	PSIM	RESIM	GENSIM
Evaluation criteria	Resource consumption for SDM development (duration and effort)	Resource consumption for SDM development (duration and effort)	Resource consumption for SDM development (duration and effort)
Type of measurement data	Objective	Objective	Objective
Measurement results for duration (divided by SDM size)	0.69 [calendar months / level]	0.18 [calendar months / level]	0.04 [calendar months / level]
Measurement results for effort (divided by SDM size)	0.58 [person months / level]	0.12 [person months / level]	0.05 [person months / level]

Table 34: Summary of evaluation results for hypothesis H_2

[44] For a definition of level variables in SDMs refer to Section 3.3.5, page 50.

11 The RESIM Project

This section presents a simulation model that was developed for Siemens Corporate Technology (Siemens CT). The purpose of this simulation modelling project was a) to demonstrate the impact of unstable software requirements on project duration and effort, and b) to analyse how much effort should be invested in stabilising software requirements in order to achieve optimal cost effectiveness [PfL00b].

11.1 Motivation and Background

The starting point for developing the simulation model was a CMM-compatible software process assessment [Völ94][MMP+98][Leb00], which Siemens CT had conducted within a Siemens Business Unit (Siemens BU). Usually, the main result of a software process assessment is a list of suggested changes to the software processes. In this case, the assessors' observations indicated that the software development activities were strongly affected by software requirement volatility. Moreover, due to the type of products developed by Siemens BU, i.e. products consisting of hardware (e.g. micro-controllers) and embedded software, the definition of software requirements was under direct control of systems engineering, and thus not totally under responsibility of the software department. During the assessment, the assessors observed that many system requirements that had already been addressed by software development were changed by the customer, or replaced by new requirements defined by systems engineering late in the project. In addition, there were many cases where system requirements that originally had been passed to software development eventually were realised by hardware, and vice versa. Based on these observations, the assessors expected that improvement suggestions that exclusively focused on software development processes (e.g., introduction of software design or code inspections) would not help stabilise software requirements. Since the software department that had ordered the process assessment primarily requested improvement suggestions that could be implemented under their responsibility, there was a need to find means that helped convince decision makers that first systems engineering had to be improved before improvements in software development could become effective. Hence the decision was made to develop a simulation model that clarified the situation, and that investigated the cost-effectiveness of improvements in systems engineering with regards to software development.

11.2 RESIM Model Development

The simulation model RESIM was developed using the System Dynamics method. The IMMoS process model guided the modelling activities.

The RESIM development process was highly iterative. 13 increments were needed to come up with a base model that was able to capture the software development behaviour mode of interest, and which contained all relevant factors governing observed project behaviour. After two additional iterations, the simulation model was ready to be used for its defined purpose.

In total, 5 persons were involved in the RESIM development (including 4 persons at Siemens CT and the SDM developer). Overall, model building and documentation consumed less than 2 person months of effort.

11.3 RESIM Design Decisions

Besides the definition of the model boundaries and model granularity, the most important design decisions were related:

a) to the typically observed behaviour patterns ("reference mode") of development projects at Siemens BU that the model should be able to reproduce through simulation, and
b) to the assumptions about the most significant cause-effect relationships ("base mechanisms") governing the observed project dynamics.

11.3.1 Reference Mode

The reference mode was defined by the dynamics of product evolution, i.e. a product is developed in three increments, and the dynamics of requirement generation, i.e. each product increment implements certain types of requirements. Both behaviour patterns can typically be observed during project performance as shown in Figure 45.

11.3.1.1 Dynamics of Product Evolution

The growth of the software product is sketched with a dashed line in Figure 45. The development of the software product is done in three subsequent, approximately equally long periods. During each period one increment is developed. The contents of the respective increments can be characterised as follows:

- Increment A: implements the base functionality (prototype)
- Increment B: implements all important requirements
- Increment C: implements all requirements (incl. customer-specific adaptations)

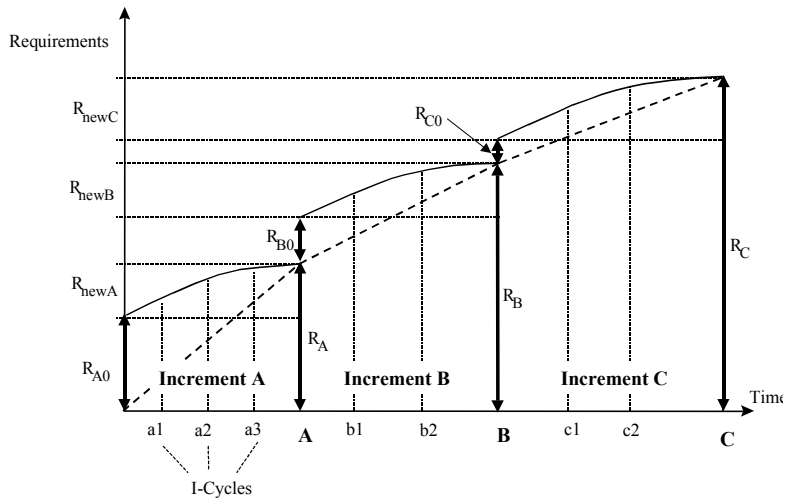


Figure 45: Typical pattern of product evolution during project performance

During the development of an increment, usually several releases that are subject to customer examination are created. The development cycles that are needed to create a release are called improvement cycles (I-Cycles).

11.3.1.2 Dynamics of Requirements Generation

At the beginning of each development period of an increment, a fixed set of requirements to start with is known (R_{A0}, R_{B0}, R_{C0}). During the development of an increment, new requirements are received from the customer (mostly as a result from the examination of releases). Typically, the number of new requirements shows a ceiling effect. Using the notation in Figure 45, the following properties can be observed:

- Number of software requirements at project start:
  R_{A0}
- Cumulated number of requirements for increment A:
  $R_A = R_{A0} + R_{newA}$
- Cumulated number of requirements for increment B:
  $R_B = R_A + R_{B0} + R_{newB}$
- Cumulated number of requirements for increment C:
  $R_C = R_B + R_{C0} + R_{newC}$
- Relationship between number of new requirements at start of an increment development:
  $R_{A0} > R_{B0} > R_{C0}$

It should be noted that only those requirements are shown in Figure 45 that actually are contained in the final product, i.e. the cumulated number of requirements does not reflect modification or replacement of already implemented requirements.

11.3.1.3 General Constraints On Project Dynamics

In order to reflect typical project behaviour properly, the model had to take under consideration two important constraints on project dynamics:

- The average productivity of the workforce, measured as the number of implemented requirements per effort unit, is constant during the development of a product increment I_j (j = A, B, C). Between increments the following relations hold: prod (I_A) > prod (I_B) > prod (I_C).
- Generally, holding the project deadline has highest priority, i.e. if the project schedule is at risk, more manpower will be added to the project.

11.3.2 Base Mechanisms

For building the SDM it was necessary to identify the most important causal relationships that are supposed to generate the typical project behaviour. Starting point for this modelling step was the assumption that the stability of software (SW) requirements definition is a measure of systems engineering (SE) quality, and that systems engineering quality can be increased, if effort is invested for related improvement actions. Based on the insights that the Siemens CT experts gained during process assessment, the following base mechanisms were identified:

- The more effort is spent on SE improvement, the better is the quality of SE: [SE effort + → SE quality +]
- The better the quality of SE is, the higher is the stability of the SW requirements: [SE quality + → stability of SW requirements +]
- The higher the stability of SW requirements is, the smaller is the number of implemented SW requirements that have to be replaced or modified: [stability of SW requirements + → replacement of implemented SW requirements -]
- The more requirements that have already been implemented are replaced by new requirements, the larger is the total number of requirements to be implemented, and thus the time needed to complete the project: [replacement of implemented SW requirements + → total number of SW requirements to implement + → SW project duration +]
- The more (excess) time is needed to complete the project, the bigger becomes time pressure: [SW project duration + → time pressure +]
- The bigger the time pressure is, the more (additional) manpower will be allocated to the project: [time pressure + → manpower +]

- The more (additional) manpower is allocated, the bigger will be the average development productivity: [manpower + → development rate (per time unit) +]
- The more requirements that have already been implemented are replaced by new requirements, the more I-cycles have to be conducted: [replacement of implemented SW requirements + → number of I-cycles +]
- The more I-cycles are conducted, the smaller is the average development productivity of the related increment: [number of I-cycles + → development rate (per time unit) -]

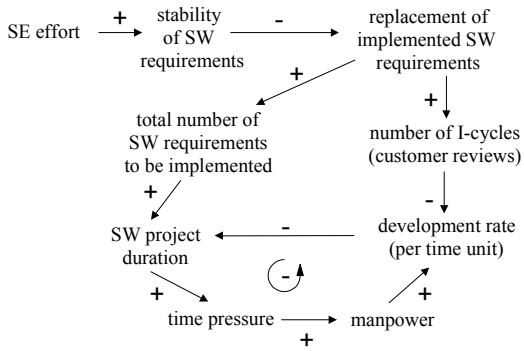


Figure 46: Causal Diagram

In order to better understand the key dynamics of the system to be modelled, these individual causal relationships can be linked together in a causal diagram (cf. Figure 46). The causal diagram clearly shows that an increase of SE effort would reduce SW project duration for two reasons. Firstly, it would reduce the overall number of SW requirements that is implemented (also counting replacements or modifications of already implemented requirements). Secondly, it would reduce the number of I-cycles, and thus increase the average development rate (per time unit). Conversely, a lack of SE effort would increase SW project duration, which – in order to keep the project deadline – could only be compensated by adding manpower. This compensation mechanism is controlled through a negative feedback loop.

11.4 RESIM Model Structure

RESIM was implemented in a modular way using the SD modelling tool Vensim 3.0 [Ven97]. The main module represents the software development with its interface to systems engineering from which the software requirements are received. Four additional modules describe certain aspects of software development in more detail, namely: workforce allocation and adjustment, effort and cost calculations, generation of new software requirements, and co-ordination of incremental software development. Figure 47 shows how the five modules are interrelated. In addition, Figure 47 indicates that module 3 (effort and cost calculation) contains the

variables that are needed for solving the issue under consideration, i.e. the policy (or independent) variable effort_for_systems_engineering, and the result (or dependent) variable total_effort (for systems engineering and software development).

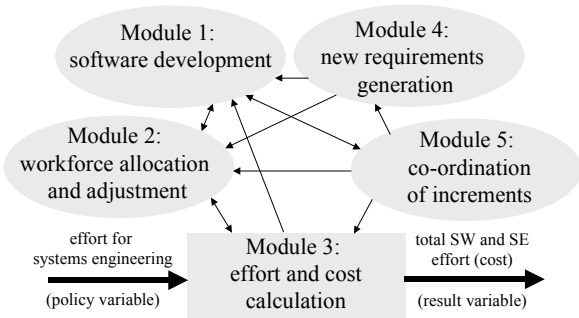


Figure 47: Modular structure of RESIM with I/O interfaces

The information flow between the five RESIM modules is summarised in Table 35. For each module relationship a characterisation of the piece of information that is exchanged is provided, together with a brief description of the information usage in the target module.

Relationship	**Information**	**Usage in target module**
Mod 1 → Mod 2	development rate	SW project planning (duration and manpower need)
Mod 1 → Mod 5	SW requirements that still have to be implemented	control of stop flag
Mod 2 → Mod 1	allocated manpower	calculation of development rate
Mod 2 → Mod 3	allocated manpower	effort and cost calculation
Mod 3 → Mod 1	weekly replace factor (represents requirements instability)	calculation of (1) development rate and (2) number of SW requirements that still have to be implemented
Mod 3 → Mod 2	weekly replace factor (represents requirements instability)	SW project planning (duration and manpower need)
Mod 4 → Mod 1	new SW requirements (at start and during development)	calculation of SW requirements that still have to be implemented
Mod 4 → Mod 2	new SW requirements (at start and during development)	SW project planning (duration and manpower need)
Mod 5 → Mod 1	synchronisation info: tells which increment is currently active	calculation of development rate
Mod 5 → Mod 2	synchronisation info: tells which increment is currently active	SW project planning (duration and manpower need)
Mod 5 → Mod 3	synchronisation info: tells which increment is currently active	effort and cost calculation
Mod 5 → Mod 4	synchronisation info: tells which increment is currently active	calculation of new SW requirements (varies among increments)

Table 35: Information flow between RESIM modules

The following sub-sections provide rough descriptions of each model module. The complete set of RESIM model equations can be found in AppendixB.

11.4.1 Module 1: Software Development

The module "software development" represents all relevant elements of the software development process and its interface to systems engineering. In systems engineering, all customer requirements are collected and analysed. Those requirements that shall be implemented in the software product are filtered out and passed over to the software development process. Generally, there are three types of software requirements: those that are known at project start, those that are newly received in addition to the existing requirements during project performance, and those that are newly received in order to replace existing (and already implemented) requirements during project performance. Replacing requirements can be received at any time during project performance. The number of replacing requirements per period (e.g. per week) is proportional to the number of requirements known at project start. The factor that determines the number of replacing requirements (variable weekly_replace_factor) depends on the quality of systems engineering, i.e. the lower the systems engineering quality the greater the proportionality factor (cf. Section 13.3).

At project end, all software requirements are implemented in the software product. The speed with which a certain number of requirements can be implemented depends on the size of the workforce and the average productivity. Since the software product is implemented in three increments, and each increment is different in nature, there is a dedicated level of average productivity assigned to each increment. Typically, the productivity is such that the development periods of the increments are equally long. The average productivity is affected by the variable weekly_replace_factor, i.e. the greater the proportionality factor, the lower is the average productivity. This dependency relationship is justified by the observation that an increase in requirement replacements increases the amount of rework (represented by an increased number of I-Cycles).

11.4.2 Module 2: Workforce Allocation and Adjustment

The module "workforce allocation and adjustment" represents all elements of the software development process relevant for allocating software developers and adjusting their number during project performance (if necessary). The initial number of software developers is calculated based on the number of initial requirements according to the typical manpower allocation pattern at Siemens BU. Workforce adjustments during project performance become necessary when continuously calculated projections of the probable project termination (which are based on the current workforce size, the current development productivity, and the number of remaining requirements to be implemented) significantly differ from the planned overall project duration.

According to nature and extent of the divergence, developers are added or taken away from the team, the adjustment being subject to realistic delay.

11.4.3 Module 3: Effort and Cost Calculations

Based on the actual project duration and the workforce allocation, the module "effort and cost calculation" calculates the overall effort used to develop the software product. In parallel, the development cost is calculated by multiplication with a cost factor.

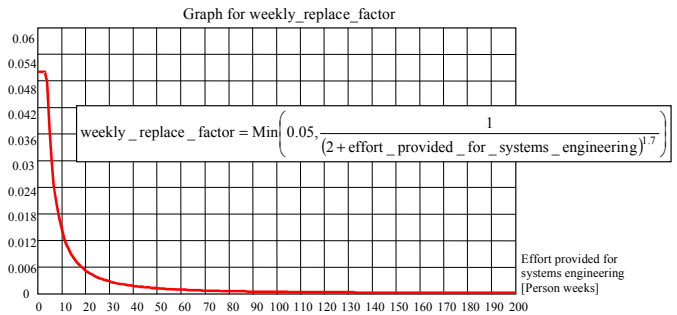


Figure 48: Relation between variable weekly_replace_factor and systems engineering effort

This module also determines the effort for conducting the systems engineering task using the variable effort_provided_for_systems_engineering as a policy parameter when running simulations. It is assumed that the quality of the systems engineering, and thus the stability of the requirements (expressed through the variable weekly_replace_factor) is a direct function of the effort invested. The assumed relationship between effort provided for systems engineering and the variable weekly_replace_factor is shown in Figure 48.

11.4.4 Module 4: New Requirements Generation

The module "new requirements generation" determines the number of new requirements received during the development of increments A, B, and C (R_{newA}, $R_{B0} + R_{newB}$, $R_{C0} + R_{newC}$). The calculations are based on typical patterns observed at Siemens BU. Figure 49 shows the behaviour of the new requirements generation rate for increment A, adjusted to an initial number of requirements of 1000 (R_{A0}).

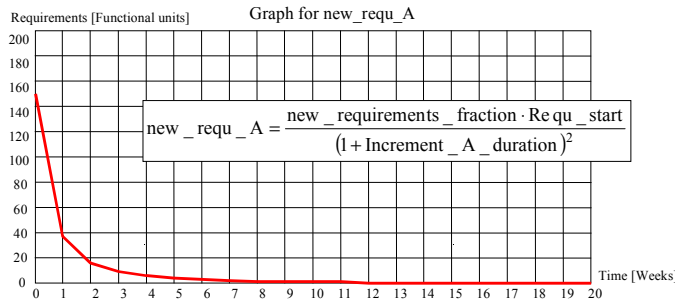


Figure 49: Relation between variable new_requ_A and time

11.4.5 Module 5: Co-ordination of Increments

The module "co-ordination of increments" is needed for synchronising the model calculations related to the development of the respective software increments.

11.5 RESIM Model Calibration and Validation

The RESIM model was calibrated based on the knowledge of Siemens CT about the behaviour patterns of typical development projects at Siemens BU (baseline). Siemens CT gained their knowledge mainly through the process assessment previously conducted within the software organisation of Siemens BU. It should be noted that most of the information about software projects at Siemens BU was qualitative of nature (with the exception of effort data). Therefore, only relations between major variables were used to calibrate the model. Based on these relations, a normalised baseline project was defined through the model constants listed in Table 36.

Model validation was mainly based on plausibility checks conducted by Siemens CT experts. It should be pointed out, however, that the most important necessary condition for model validity, i.e. the ability to reproduce the reference mode, was fulfilled. Figure 50 presents the simulated patterns of SW product growth (implemented stable requirements / variable: SW_product) and growth of SW requirements that actually are contained in the final SW product (stable requirements / variable: actual_all_SW_requirements), as generated by the SDM for the baseline situation, i.e. with 10 person-weeks of effort provided for systems engineering. Using the criterion of face validity, these patterns clearly reproduce the reference behaviour as defined in Section 11.3.1 (cf. Figure 45).

Description	Unit	Value
Number of requirements at project start (R_{A0})	functional unit	1000
New requirements fraction (needed to calculate R_{newA}, R_{newB}, R_{newC})	dimensionless	0.15
Initial requirements fraction for increment B (needed to calculate R_{B0})	dimensionless	1.8
Initial requirements fraction for increment C (needed to calculate R_{C0})	dimensionless	0.5
Target project completion time	week	100
Nominal average productivity for increment A	functional unit / person week	11
Nominal average productivity for increment B	functional unit / person week	4
Nominal average productivity for increment C	functional unit / person week	2.5
Cost per effort unit	money unit / person week	2000

Table 36: RESIM model constants used for calibration

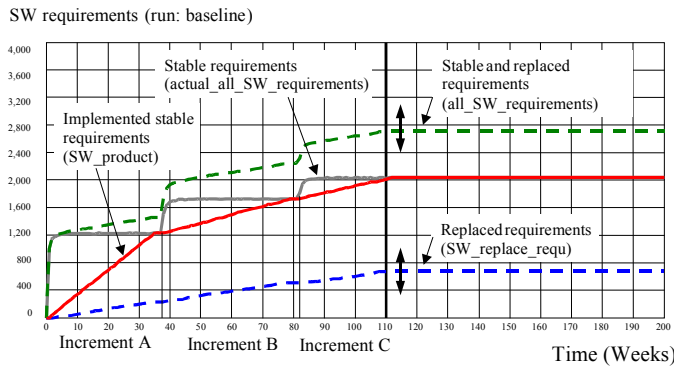


Figure 50: Reproduction of the RESIM reference mode

Note that the number of replaced requirements (variable: SW_replace_requ) and thus the total number of stable and replaced requirements (variable: all_SW_requirements) can vary largely as a consequence of variation in effort invested to improve systems engineering.

11.6 RESIM Model Application

The question that had to be answered with help of the simulation model was: "How much effort should be invested into systems engineering at Siemens BU in order to improve (software) requirements analysis and thus minimise the overall software development cost?" To answer this question, an equivalent mathematical minimisation problem was formulated:

$$\text{total\_effort} = x + \sum_{t=1}^{T} y(t) \rightarrow \min$$

with:

t: elapsed time (weeks)
T: project termination (weeks)
x: effort for systems engineering (person weeks)
y: weekly effort consumption for software development (person weeks / week)

The solution to this problem was found through variation of the policy variable x (model parameter: effort_provided_for_systems_engineering) and by using the built-in optimisation function of the simulation tool Vensim, which applies the Fletcher-Powell algorithm [FlP63]. The most important results are summarised in Table 37.

Simulation run	**n1**	**n2 (baseline)**	**n3**	**n5**	**optimal**	**n6**
Syst. Eng. effort [person weeks]	5	10	15	30	42	50
SW dev. effort [person weeks]	875	586	499	452	420	416
Total effort [person weeks]	**880**	**596**	**514**	**482**	***462***	**466**
Cost-effectiveness	-	0	0.138	0.191	0.225	0.218
AARR per week [%]	1.83	0.73	0.40	0.14	0.08	0.06

Table 37: Summary of RESIM simulation results

It turned out that an increase of the systems engineering effort share from 1.7% of the total effort (baseline situation) to 9.1% of the total effort (optimal situation) will reduce the overall cost for systems engineering and software development by more than 20% (from 596 to 462 person weeks). This effect is mainly due to the stabilisation of requirements, which is expressed in terms of the actual average requirements replacement (AARR) per week. In the optimal case, on average only 0.08% of the currently known (and yet implemented) requirements were replaced per week, adding up to a total of 29 replaced requirements during project performance.

12 The GENSIM Project

This section presents the motivational background and the concepts of a web-based training (WBT) module for student education on the topic of software project management.

12.1 Introduction and Background

There is an increasing demand for software project managers in industry. Therefore, efforts are needed to develop the management-related knowledge and skills of the current and future software workforce. In particular, university education needs to provide to their computer science and software engineering (SE) students not only technology-related skills but in addition a basic understanding of typical phenomena occurring in industrial (and even academic) software projects.

The potential of simulation models for the training of managers has long been recognised: flight-simulator-type environments (or microworlds) confront managers with realistic situations that they may encounter in practice, and allow them to develop experience without the risks incurred in the real world. Two detailed examples of training workshops based on real industrial cases using simulation can be found in [Gra+92] (other examples are mentioned in [Mil95] and [Mor88], to give a few pointers).

As regards the specific topic of software project management, experimental studies have been conducted on using simulation models representing the typical behaviour of software development projects.

Experiments carried out at the Jet Propulsion Laboratory aimed at studying the decision-making process of software managers [Lin93]. Twenty managers were asked to conduct a project simulated with the aid of the Software-Engineering Process Simulation Model (SEPS) [LAS97]; some were provided with cause-effect feedback of their actions, while the others were not. It was observed that the second group (without feedback information) tended to act in a more "fire fighting" mode than the first one; and the feedback information was most beneficial to the less experienced managers.

At Draper Laboratory, a simulation model served as the basis for an experiment involving a group of 50 experienced managers [SNV93]. The scenario of the experiment involved a 15 percent requirement change in the course of the simulated project. Few of the managers were able to adapt properly to this situation: most of them reacted by hiring new staff late on the project (a typical "fire fighting" policy), and experienced budget and schedule overruns. Worse, the managers tended to reproduce exactly the same errors

when running the scenario for the fourth or fifth time. The authors of the study conclude that the participating managers were limited by their mental model of the process, and that they were reluctant to change it.

The two experiments mentioned above show that natural one-way causal thinking can be detrimental to the success of software managers. Therefore, the aim of the training should go beyond that of facing people to realistic problems; the concern is also to make managers adopt systems thinking, and perceive the existence of (unexpected) feedback to management decisions. This sets the problem of an adequate game interface and more importantly of the workshop or training course organisation: just running a simulation model as a black box may not be sufficient to have people gain insight into the software development process, and accept to alter their mental model. Lessons learned from experiments in strategic management training [Gra+92] indicate that adequate course organisation avoids the situation observed at Draper Laboratory, where managers played the simulation game without analysing their errors, hence gaining insufficient insight into the problem.

The objective of the GENSIM project was to develop a simulation-based WBT module that helps SE students understand the complex decision-making situations in software project management, which are often characterised by trade-off effects between conflicting goals. Typical project management goals are to implement a specified functionality, to control or shorten project duration (time-to-market), to control or decrease project cost, and to control or improve the quality of the delivered end product (for specified product functionality).

The main goal of the simulation-based WBT module is to facilitate the transfer of knowledge about software project management to computer science and SE students through a scenario-driven interactive learning environment. An additional goal is to raise interest in the topic of software project management among SE students, and to make them aware of some of the difficulties associated with controlling the dynamic complexity of software projects.

The WBT module that was developed in the scope of the GENSIM project can be run using standard web-browsers (e.g. Netscape). It is composed of course material that is presented to the trainees through the internet. The core element of the WBT module is the SD simulation model GENSIM (Generic Simulator). With GENSIM typical behaviour of software development projects can be simulated.

The design of the SD model GENSIM (WBT/Simulator) and the single-learner training scenario (WBT/Scenario) that triggers the activation of the simulation component are described in the following sections (more details can be found in [Kle00], [PKR00a] and [PKR01a]).

12.2 Design of the WBT/Simulator GENSIM

The WBT/Simulator GENSIM represents in a simplified, generic waterfall-model-like fashion three phases of a typical software development project: Design, Implementation, and Test. The calibration of GENSIM was not based on a real industrial case or on exhaustive empirical research, but on the functional relationships between effort, time, and size (functionality), as suggested by the well-known COCOMO model [Boe81].

12.2.1 GENSIM Model Parameters

The most important GENSIM model parameters that can be accessed through the WBT module by the trainees are listed in Table 38. The first three columns list input parameters, the last column lists output parameters. The model parameters in the first column are needed to characterise the software development project. This is done by defining the expected job size (i.e. product size or functionality) and project complexity. These parameters are the minimal input necessary to run model simulations, thus they are qualified as mandatory. The model parameters in the second and third column can be either altered by the project manager or left unchanged to the default assignment, therefore they are qualified as optional. For example, if the project manager does not define the parameters in the second column at project start, default values are automatically provided by the model, and the simulation will be run using these. The interesting feature of the simulation model is that the project manager can change these parameters at any time during the simulation run. For example, whenever the project manager feels that more manpower is needed in order to hold the project deadline, the simulation can be interrupted and the parameter "Planned manpower" can be altered. Besides these core management parameters dealing with schedule planning, manpower allocation, alteration of the job size, or quality target setting, the WBT/Simulator GENSIM offers parameters that allow the project manager to apply QA technologies such as code or design inspections. The application of QA technologies can help the project manager influence project duration, effort consumption, and product quality. These parameters are listed in the third column of Table 38. The last column lists the main output parameters, i.e. job size at end of project, actual project completion time, total effort consumed, and actual field defect density as a measure of product quality.

Input Parameters			Output Parameters
Project Characterisation Parameters (mandatory)	Project Management Parameters (optional)	QA technology-related Parameters (optional)	
Initial job size in tasks	Job size adjustment		Job size in tasks
	Planned completion time		Project completion time
	Planned manpower		Effort
Project complexity	Manpower skill		
	Goal field defect density	Application of design and/or code inspections	Field defect density

Table 38: Key parameters of the WBT/Simulator GENSIM

12.2.2 GENSIM Model Structure

In total, the WBT/Simulator GENSIM consists of five interrelated sub-models (views):

- Production View: This view represents a typical software development lifecycle consisting of the following chain of transitions: set of requirements (planned functionality) → design documents → code → tested code. In order to limit model complexity, detection of defects during testing only causes reworking of the code and not of previous design documents. Similarly, optional quality assurance (QA) activities conducted during design and coding will only trigger reworking of documents that are developed during the respective phase.
- Quality View: In this view, the defect co-flow is modelled: defect injection (into design or code) → defect propagation (from design to code) → defect detection (in the code during testing) → defect correction (only in the code). Optionally, additional QA activities will result in defect detection and rework already during design and coding.
- Effort View: In this view, the total effort consumption for design development, code development, code testing, optional QA activities, and defect correction (rework) is calculated.
- Initial Calculations View: In this view, the nominal value of the central process parameter "productivity" is calculated using the basic COCOMO equations for estimating effort and project duration. The nominal productivity varies with assumptions about the product development mode (organic, semi-detached, embedded) and characteristics of the available project resources (e.g. developer skill). The average needed manpower is derived from the effort and duration estimates.
- Productivity, Quality & Manpower Adjustment View: In this view, project-specific process parameters, like (actual) productivity, defect generation, effectiveness of QA activities, etc., are determined based on

a) planned target values for manpower, project duration, product quality, etc., and
b) time pressure induced by unexpected rework or changes in the set of requirements.

Figure 51 shows the most important cause-effect relationships that generate the dynamic behaviour of the simulated software project. The bold arcs represent those relationships that are defined by the basic COCOMO model [Boe81], i.e. the model equations for project effort and duration. The fundamental input to the model is the planned product size or functionality (P-SIZE). Based on the planned product size, the nominal effort for the phases design, implementation, and test (N-EFF1) as well as the nominal duration (N-DUR1) are calculated. The "1" at the end of the variable names indicates that both calculations are made without consideration of potential rework due to defect detection during testing. Based on nominal effort and duration, the nominal average manpower allocation (N-AMP) is calculated. As long as no other information is received, the value of N-AMP is equal to A-AMP (actual average manpower allocation). Planned size and nominal effort are used to calculate the average productivity per person-day (PROD-per-PD). Together with the process parameter N-DEF-Inject-Rate (nominal average defect injection rate), the variables PROD-per-PD, N-DUR1, and A-AMP determine the number of undetected defects contained in the design documents (Undet-Des-DEF), and the number of undetected defects contained in the code (Undet-Code-DEF). As long as no defects are detected and corrected during design, all undetected design defects propagate into the code. During testing, a portion of the defects contained in the code is detected (Det-Code-DEF). The size of the portion depends on the nominal test-effectiveness (N-Test-Effectiveness) and the total number of defects contained in the code (Undet-Code-DEF). Defect detection causes rework effort (Code-Rework-EFF) which in turn decreases productivity (PROD-per-PD) and the number of undetected code defects (Undet-Code-DEF). As a result of the decrease in productivity, the actual duration of the project will increase (A-DUR2). The size of the rework effort depends on the phase in which the defects have been detected. For example, the amount of rework induced by corrections of defects detected during coding is assumed to be smaller than the amount of rework induced by corrections of defects found during testing (variables N-Rework-Eff-Coding and N-Rework-EFF-Test). The quality of the tested and reworked product can be expressed as defect density (DEF-Density) by simply dividing the number of remaining undetected defects in the code by the product size.

It should be noted that Figure 51 only provides an incomplete picture of the causal relationships contained in the GENSIM model. For example, the possibility to detect and correct defects during the design phase is not shown in order to improve readability of the presented graph.

The dotted lines in Figure 51 indicate causal effects that are triggered when input parameters are altered. Alteration of input parameters creates differences to nominal values and thus triggers a chain of causal reactions that

strive to bring the system back to equilibrium by re-adjusting the affected model variables. For example, if the planned average manpower allocation (P-AMP) is not equal to the N-AMP, then this unbalance has a negative effect on productivity and quality, either due to communication overhead (in case of overstaffing), or due to exhaustion of individuals (in case of understaffing).

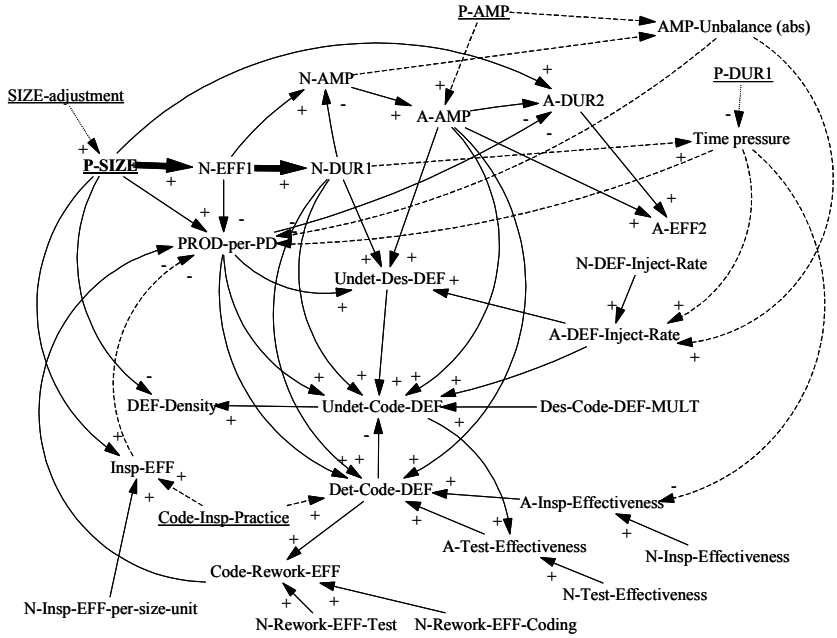


Figure 51: Extract of the causal diagram

Since Figure 51 does not show all possibilities of input parameter manipulation and the induced effects on project behaviour, a list of the most interesting effects that are induced by alteration of input parameters is summarised in Table 39. Table entries printed in Italics describe effects that can be created by the simulation model but are not included in the causal model shown in Figure 51. For some of the causal effects the exact numerical relationships can be found in Appendix C.

It should be noted that alterations of input parameters can be made at any point in time during project simulation, and that compound effects on project behaviour can be created by concurrent multiple-parameter alteration.

Description	Altered input parameters	Induced effects
Unbalanced average manpower	P-AMP > 0 and P-AMP <> N-AMP	1. Defect injection rate increases 2. Productivity per person-day decreases
Time pressure	P-DUR1 > 0 and P-DUR1 < N-DUR1	1. Defect injection rate increases 2. Productivity per person-day decreases 3. Inspection effectiveness decreases
Negative time pressure	P-DUR1 > 0 and P-DUR1 > N-DUR1	1. Defect injection rate decreases 2. Productivity per person-day decreases 3. No effect on inspection effectiveness
Application of QA activities (during coding or *design*)	Code-Insp-Practice > 0 *Des-Insp-Practice > 0*	1. Inspection effort increases 2. The number of detected defects increases (during coding or *design*)
Improved product quality	*Target defect density > 0 and* *Target defect density < DEF-Density*	*1. Productivity per person-day decreases (only during testing)* *2. Test effectiveness increases*
Manpower skill is insufficient	*Manpower skill < 1*	*1. Productivity per person-day decreases* *2. Defect injection rate increases*

Table 39: Effects of input parameter alterations

12.2.3 GENSIM Model Implementation

For the implementation of GENSIM the SD modelling tool Vensim 3.0 was used [Ven97]. The GENSIM user interface, which provides a set of instructive graphical analysis functions to the model user, was developed with Borland Delphi. A brief description of the GENSIM user interface can be found in Appendix C (a detailed description is provided in [Kle00]).

12.3 Activation of the WBT/Simulator GENSIM

The sequence of training steps of the WBT module, into which the WBT/Simulator GENSIM has been integrated, is structured according to a pre-defined scenario. The WBT/Scenario triggers the activation of the WBT/Simulator GENSIM. This section describes the WBT/Scenario structure and its main characteristics. It should be noted that in order to support the effectiveness of the WBT, and to avoid the phenomena observed at Draper Laboratory, the trainee will be properly briefed before and de-briefed after running the scenario of the WBT module [Lan95].

12.3.1 WBT/Scenario Structure

The WBT/Scenario has been structured into a sequence of scenario blocks. The current version of the WBT module contains four project management (PMT) training blocks:

- Block 1 - PMT Introduction: General introduction into the main tasks of a software project manager and the typical problems he has to solve with regards to project planning and control. This includes a brief discussion of problems caused by the so-called "magic triangle", i.e. the typical presence of unwanted trade-off effects between project effort (cost), project duration, and product quality (functionality).
- Block 2 - PMT Role Play: Illustration of common project planning problems on the basis of a case example in which the trainee takes over the role of a fictitious project manager.
- Block 3 - PMT Planning Models: Presentation of basic models that help a project manager with his planning tasks, namely a process map, and a predictive model for effort, schedule and quality. The predictive model is identical with the GENSIM model contained in the WBT/Simulator. A description of the main variables of GENSIM and their functional relationships (causal model) is provided to the trainees.
- Block 4 - PMT Application Examples: Explanation on how to apply the planning models on the basis of examples that are presented in the form of little exercises. In particular, examples that involve trade-off between two and three variables of the magic triangle are presented. During this scenario block, trainees can use a stand-alone application of GENSIM to reproduce the application examples while reading the training materials.

Figure 52 provides an overview of the WBT/Scenario structure indicating the two different ways of activating the GENSIM model, i.e. integrated into the WBT module or as a stand-alone tool.

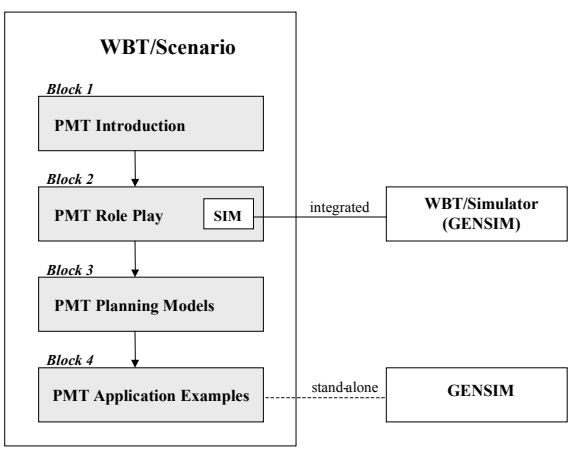


Figure 52: WBT/Scenario structure

The PMT Role Play block provides the most interesting scenario because it forces the trainee to interact with the WBT module. The goal of the scenario block is to make trainees aware of typical performance patterns of software projects, and to make them understand how knowledge about these patterns can help project managers in making the right decisions, for example with regards to project planning.

The scenario of the PMT Role Play block has been designed to help the trainee understand the complex implications of a set of empirically derived principles that typically dominate software projects conducted according to the waterfall process model. Even though the waterfall process approach is no longer state-of-the art and in most industrial organisations not even state-of-the-practice, it is expected that it is still an interesting object of study for students having little or no experience with real-world industrial software development. The set of principles used in the block scenario (cf. Table 40) was distilled from the top 10 list of software metric relationships published by Barry Boehm [Boe87].

No.	Principle
1	"Finding and fixing a software problem after delivery is 100 times more expensive than finding and fixing it during the requirements and early design phases."
2	"You can compress a software development schedule up to 25 percent of nominal, but no more."
3	"Software development and maintenance cost are primarily a function of the number of source lines of code (SLOC) in the product."
4	"Variations between people account for the biggest differences in software productivity."
5	"Software systems and products typically cost 3 times as much per SLOC as individual software programs. Software-system products (i.e. system of systems) cost 9 times as much."
6	"Walkthroughs catch 60% of the errors."

Table 40: List of principles dominating project performance

In order to make the trainee understand the implications of these principles (and their combinations), a role-play is conducted in which the trainee takes the role of a project manager who has been assigned to a new development project. Several constraints are set, i.e. the size of the product and its quality requirements, the number of software developers available, and last but not least the project deadline. The first thing to do for the project manager (in order to familiarise with the simulation model) is to check whether the project deadline is feasible under the resource and quality constraints given. Running a simulation does this check. From the simulation results, the project manager learns that the deadline is much too short. Now, the scenario provides a set of actions that the project manager can take, each action associated with one of the principles and linked to one of the model parameters listed in Table 38. Soon the project manager learns that his department head does not accept all of the proposed actions (e.g. reducing the product size or complexity). Depending on the action the project man-

ager has chosen, additional options can be taken. Eventually, the project manager finds a way to meet the planned deadline, e.g. by introducing code and design inspections (associated with Principle 6 in Table 40).

The role-play is arranged in a way that the project manager can only succeed if he/she combines actions that relate to at least two of the principles listed in Table 40. At the end of the role-play, a short discussion of the different possible solutions is provided, explaining the advantages and disadvantages of each. A full description of the design and implementation of the PMT Role Play block can be found in [Kle00].

12.3.2 WBT/Scenario Block Characteristics

In order to describe the WBT/Scenario blocks with regards to didactical aspects, they can be characterised using the following criteria:

(C.1) Duration [minutes]

(C.2) Medium [paper-based vs. web-based]

(C.3) Trainee interaction mode [active vs. passive]

(C.4) Usage of WBT/Simulator [yes / no]

The result of characterising the WBT/Scenario blocks according to these criteria is summarised in Table 41.

WBT/Scenario Block	C.1	C.2	C.3	C.4
PMT Introduction	3 min	paper / web	passive	no
PMT Role Play	15 min	web	active	yes (embedded)
PMT Planning Models	15 min	paper / web	passive	no
PMT Application Examples	12 min	paper / web	passive / active	no / yes (stand-alone)

Table 41: Characterisation of WBT/Scenario blocks

The PMT Introduction and PMT Planning Models blocks are the most conventional with regards to didactical aspects. They deliver course content to the trainee in the form of plain text, which can be presented either in paper-based or in web-based form. In both cases the interaction is quite limited since it consists basically of text reading. A web-based presentation format certainly provides more possibilities to include "catchy" features, but in either case the main driver of trainee activation, i.e. the use of the WBT/Simulator, is excluded. It should be noted that both blocks together are expected to consume only 40% (18 minutes) of the overall time needed to conduct a full training session.

The PMT Application Examples block could be performed in the same way as the PMT Introduction or PMT Planning Models block, but it is also possible to invoke the WBT/Simulator and thus trigger active participation of the trainee. It should be noted, however, that in the current version of the WBT module only stand-alone usage of the WBT/Simulator is supported, i.e. the trainee has to switch between web-browser and simulation tool. The time consumption for completing a PMT Application Examples session is estimated to be 12 minutes.

The most advanced scenario block of the WBT module with regards to didactical aspects is PMT Role Play, because it fully exploits the possibilities associated with web-based presentation. It offers an integrated access to the WBT/Simulator and requires active interaction of the trainee in order to be able to proceed and solve the project management problem imposed by the role-play. The time consumption for completing a PMT Role Play session is estimated to be about 15 minutes.

13 Effectiveness of IMMoS

The criterion for evaluating effectiveness is the suitability of the developed SDM for its purpose from the point of view of the SDM user. For RESIM and GENSIM, the SDM user (role) and the purpose were specified in the SDM Goal Definition. Since IMMoS was not applied in the PSIM project, the identification of the model user and the model purpose changed during the modelling process (cf. Section 4.6.1). Therefore, in Table 42, which summarises the SDM Goal Definitions, for PSIM the rows "Role" and "Purpose" distinguish between what actually turned out to be the goal towards the end of the modelling process (underlined text), and what was thought to become a useful goal in the future (plain text).

	SDM Goal Definition		
	Implicitly defined (without IMMoS)	**Explicitly defined (with IMMoS)**	
SDM	**PSIM**	**RESIM**	**GENSIM**
Role	Project Manager / Process Owner	Process Consultant (Assessor)	Computer Science Students
Scope	Single Project	Single Project	Single Project
Dynamic Focus	Impact of changes in process or product on project performance	Impact of software requirements volatility on software development productivity	Trade-off effects between product size, development time, effort consumption, and product quality
Purpose	Planning / Control / Improvement	Understanding	Understanding (Training)
Environment	Siemens (Telecommunication / HICOM) – 1994 / 1995	Siemens (Automotive / Micro Controllers) – 1998	University of Kaiserslautern – 1999

Table 42: Summary of SDM Goal Definitions for PSIM, RESIM and GENSIM

In the following sections, more details are provided on how suitability was measured for the developed SDMs. Even though the suitability of PSIM is not required for evaluating the effectiveness of IMMoS, it is a prerequisite for evaluating the efficiency of IMMoS.

13.1 Suitability of PSIM

At the beginning of the modelling process of PSIM, the goal was to provide a simulation tool to the process owner that could be used for two purposes. First, the SDM should demonstrate that the recently defined concurrent software development process is more productive than the old one. Second, the

SDM should provide a microworld representing the actual software development processes in the organisation. This microworld should help the process owner to explore improvement opportunities. During the modelling process, it was decided that the SDM should also be used by project managers to plan and control the projects under their responsibility.

13.1.1 Role: Process Owner

From the point of view of the process owner, the purpose of the SDM, i.e. "improvement", could not be fulfilled satisfactorily. The main reason for this was the unrealistic assumption of the process owner that the new concurrent development process was actually implemented in the projects for which the SDM should help with planning and controlling. The fact that the projects for which the SDM should be used by the project manager, was actually still using the old development process, required that the also the SDM represented the old process. Otherwise, the model would not have been valid for the planning and controlling task. Since the modelling effort was only sufficient to describe the development processes in place in order to satisfy the needs of the project manager, it was not possible to use simulation to demonstrate the higher productivity of the new concurrent development process.

Even though PSIM did not fulfil the original purpose of the process owner, it was still useful. It uncovered that the process owner was not aware of the fact that the new (prescriptive) process model had not yet been implemented and was not used in current practice by the responsible project manager and the developer team. So, even though unintended, PSIM actually fulfilled the purpose "understanding" from the point of view of the process owner.

13.1.2 Role: Project Manager

The suitability of PSIM for the purpose "planning and controlling" from the point of view of the role project manager was evaluated by using subjective and objective data.

(1) Objective data: In order to evaluate the predictive power of PSIM, the accuracy of simulation results was measured by comparing the plan data generated by PSIM to the actual data. Since product size and available manpower was fixed, main focus was put on schedule, i.e. completion of milestones, and quality, i.e. defects found during inspections and test. These comparisons at showed good results. The model predictions anticipated the actual project behaviour with a maximal deviation of less than 10%. This number was much better than the real planning (without PSIM), which deviated from the actual data by more than 30%. It should be noted, however, that the real planning was based on the assumption

that the new concurrent development process was used, which was not the case, as has been stated earlier.

(2) Subjective data: The judgement of the usefulness of PSIM by the project manager was very positive. Besides the fact that the planning accuracy seemed to be good, the project manager stated that he considered PSIM to be a useful "sparring partner" for decision-making, and – in the long run – as a promising tool for process analysis.

13.1.3 Suitability of PSIM with IMMoS

The discussion of the suitability of PSIM in the previous sections has shown that PSIM basically fulfilled its purpose for the relevant user roles. Obviously, this could be achieved without using the IMMoS approach. However, the discussion has also shown that the definition of the modelling goal without IMMoS was quite time consuming. It seems to be plausible, although it cannot be proven, that with IMMoS, the PSIM development would have been much more goal-oriented from the beginning yielding exactly the same definition of user roles and purposes, but with less iterations, more explicitly, and thus more efficiently.

13.2 Suitability of RESIM

The purpose of RESIM was to help the model user, i.e. a software process assessor at Siemens CT, in his consulting activities at a Siemens Business Unit (Siemens BU). The goal of the model user was to transfer his understanding of how requirements volatility impacts development productivity to the line management and process owners at Siemens BU. The evaluation of whether this goal was achieved could only be based on subjective judgement of the model user.

Based on the simulations, it was possible to demonstrate that software requirements volatility is extremely effort consuming for the software development organisation and that investments in systems engineering in order to stabilise requirements definition would well pay off. According to the model user, the results of the simulation experiments had provided a twofold advantage. Firstly a deeper understanding of the procedures for capturing and changing requirements grew up in the assessment team while discussing about real life and its representation through the model. Secondly the quantitative evaluation of the present situation and of the effect of possible changes was convincing for the Siemens BU. The model results helped a lot to broaden the view of the requirements process within software development and to start an improvement program across all the roles and organisations participating in this process.

Note that all results produced by the simulation model are based on qualitatively formulated assumptions underlying the model structure. Without thor-

ough review of the model structure by experts of Siemens BU, and without a calibration of the model parameters and model functions to empirical data, the model cannot be used for precise point estimates in the sense of a predictive model. However, according to the model user, having such a simulation model at hand makes it quite easy to visualise the critical project behaviour and to discuss the assumptions about the cause-effect relationships that are supposed to be responsible for the generated behaviour. In that sense, the model user at Siemens CT felt that building RESIM was a useful exercise, and that similar models can be helpful in future process improvement projects with Siemens BUs.

13.3 Suitability of GENSIM

The suitability of the model GENSIM was evaluated based on the analysis of subjective and objective data gained from a controlled experiment that compared the effectiveness of GENSIM with that of a standard software project planning model [PKR01b]. Note that GENSIM was not evaluated stand-alone but in the context of its application as a simulation component integrated into a simulation-based training module (cf. Section 12 for details).

The main objective of developing and applying the simulation-based training module was to facilitate effective learning about certain topics of software project management to computer science students. This was done by providing a scenario-driven interactive single-learner environment that can be accessed through the internet by using a standard web-browser. An additional goal was to raise interest in the topic of software project management among computer science students, and to make them aware of some of the difficulties associated with controlling the dynamic complexity of software projects.

The training module used in the study is composed of course material on project planning and control. The arrangement and presentation of the course material is defined by the single-learner training scenario. The core element of the training module is a set of interrelated project management (i.e. planning) models, represented by a simulation model that was created by using the System Dynamics (SD) simulation modelling method [For71][RiP81]. This model – GENSIM – simulates typical behaviour of software development projects.

In order to investigate the effectiveness of computer-based training in the field of software project management using a SD simulation model, a controlled experiment applying a pre-test-post-test control group design was conducted. The subjects who were willing to participate in the experiment had to pass two tests, one before the training session (pre-test) and one after the training session (post-test). The effectiveness of the training was then evaluated by comparing within-subject post-test to pre-test scores, and by comparing the scores between subjects in the experimental group, i.e. those who used the SDM, and subjects in the control group, i.e. those who

used a conventional project planning model instead of the SDM. In the study, the well-known COCOMO model [Boe81] was used by the control group since this model is quite comprehensive and can be considered as state-of-the-practice in many industrial software organisations.

The following dimensions were used to characterise "effectiveness" of the training session:

1. Interest in software project management issues.
2. Knowledge about typical behaviour patterns of software development projects.
3. Understanding of "simple" project dynamics.
4. Understanding of "complex" project dynamics.

In the study, these dimensions were represented by dependent variables (Dep.1 to Dep.4).

13.3.1 Hypotheses

Standard significance testing was used to analyse the effectiveness of the training session. Two null hypotheses together with their associated alternative hypotheses were stated.

The first null hypothesis was stated as:

$H_{0,1}$: There is no difference between scores before (pre-test) and after (post-test) the training session.

The second null hypothesis was stated as:

$H_{0,2}$: There is no difference in effectiveness between the experimental group (using the SDM) and the control group (using the COCOMO model).

The alternative hypotheses, i.e., what was expected to occur, were then stated as:

1. H_1 (relates to $H_{0,1}$) – "Post-test versus pre-test scores": The average performance of all subjects (experimental group and control group) during post-test is better than during pre-test.
2. H_2 (relates to $H_{0,2}$) – "Performance improvement": The average performance improvement of the experimental group is better than the average performance improvement of the control group.
3. H_3 (relates to $H_{0,2}$) – "Post-test performance": The average post-test scores of the experimental group are better than the average post-test scores of the control group.

H_1, H_2 and H_3 apply to all dependent variables (Dep.1 to Dep.4). Note that it is not expected that both alternative hypotheses of $H_{0,2}$ will occur simultaneously. This reflects on the fact that occurrence of alternative H_2 is less likely when pre-test scores of the experimental group are significantly higher than those of the control group. Similarly, alternative hypothesis H_3 is less likely to occur when pre-test scores of the control group are significantly higher than those of the experimental group.

13.3.2 Subjects

The participants of the study were computer science students at the University of Kaiserslautern, Germany, who were enrolled in the advanced software engineering class lasting one semester. While the course was running, subjects were asked if they would be interested in participating in an experiment related to software project management issues that would involve a simulation model. The subjects knew that they would have to participate in a self-learning training session, that they would have to pass a test, and that the test scores would be analysed to evaluate the training session. Twelve students expressed their interest in participation.

As the German system allows students to take different classes at different times during their studies, information on their personal background with regard to experience in software development and software project management was captured before passing the pre-test.

13.3.3 Treatments

The training sessions of both groups, experimental and control, was structured by training scenarios, consisting of a sequence of scenario blocks. The generic scenario structure is composed of the four scenario blocks described in Section 12.3.1.

13.3.3.1 Treatment of the Experimental Group.

The experimental group passed all scenario blocks. The SD model GENSIM was used as the predictive model in scenario blocks 3 and 4. In addition, the SD model GENSIM was integrated into the interactive role-play offered by scenario block 2.

13.3.3.2 Treatment of the Control Group.

The control group passed only scenario blocks 1, 3, and 4. The predictive model used in scenario blocks 3 and 4 was the intermediate COCOMO model. A detailed description of the COCOMO model can be found in [Boe81].

13.3.4 Experimental Design

For evaluating the effectiveness of a training session using SDM simulation, a pre-test-post-test control group design was applied. This design involves random assignment of subjects to an experimental group and a control group. Both groups have to pass a pre-test and a post-test. The pre-test measures the performance of the two groups before the treatment, and the post-test measures the performance of the two groups after the treatment. By studying the differences between the post-test and pre-test scores of the experimental group and the control group, conclusions can be drawn with respect to the effect of the treatment (i.e. the independent variable of the experiment) on the dependent variable(s) under study.

13.3.5 Experimental Variables

During the experiment, data for three types of variables are collected. Table 43 lists all experimental variables, including one independent variable, four dependent variables, and three variables that represent potentially disturbing factors.

Independent Variable	
Ind.1	Type of treatment
Dependent Variables	
Dep.1	Interest in software project management issues (“Interest”)
Dep.2	Knowledge about typical behaviour patterns of software development projects (“Knowledge”)
Dep.3	Understanding of “simple” project dynamics (“Understand simple”)
Dep.4	Understanding of “complex” project dynamics (“Understand complex”)
Disturbing Factors	
DiF.1	Personal background
DiF.2	Time consumption / time need
DiF.3	Session evaluation (personal perception)

Table 43: Experimental variables

13.3.5.1 Independent Variables.

The independent variable Ind.1 (type of treatment) can have two values, either T_A, which is applied to the experimental group, or T_B, which is applied to the control group. The difference between T_A and T_B is basically determined by two factors. The first factor is the training scenario according to which the course material is presented. The second factor is the planning model that is used to support software project management decision-making. Table 44 briefly summarises the differences between the treatment of the experimental group and the treatment of the control group, indicating

the duration of the scenario blocks applied, and providing information on the nature of the used planning models.

	Treatment T_A	Treatment T_B
Scenario	Block 1 – 3 min	Block 1 – 3 min
	Block 2 – 15 min	n/a
	Block 3 – 15 min	Block 3 – 30 min
	Block 4 – 12 min	Block 4 – 12 min
PMT model	SDM: • behavioural (white box) • point estimates (black box)	COCOMO model: • point estimates (black box)

Table 44: Differences between treatments

With regard to the scenario, the main difference consists in the application of scenario block PMT Role Play for treatment T_A. As a consequence of performing the scenario block PMT Role Play, interaction of the trainee with the training module will be high whereas treatment T_B will only trigger low interaction of the trainee with the training module. The application of scenario block 2 also has implications on the medium through which the course material is presented. Treatment T_B may choose between paper-based or web-based, whereas treatment T_A at least requires web-based presentation of scenario block PMT Role Play. It should also be noted that for treatment T_B the presentation style of the training materials can always be similar to a textbook presentation even if the materials are provided web-based. With regard to the model that is used during the training session, treatment T_B exclusively relies on a black-box model providing point estimates, such as COCOMO. In contrast to this, by using a SD simulation model, treatment T_A is based on a white-box model that in addition to point estimates facilitates insights into behavioural aspects of software projects.

13.3.5.2 Dependent Variables

The dependent variables Dep.1, Dep.2, Dep.3, and Dep.4 are determined by analysing data collected through questionnaires that all subjects have to fill in, the first time during the pre-test, and the second time during the post-test. Each questionnaire consists of 5 to 7 questions where answers have to be provided on a uniform scale. The value of each dependent variable will then be equal to the average score derived from the related questionnaire.

The contents of the questionnaires are as follows:

- Dep.1 (“Interest”): Questions about personal interest in learning more about software project management.
- Dep.2 (“Knowledge”): Questions about typical performance patterns of software projects. These questions are based on some of the empirical

findings and lessons learned summarised in Barry Boehm's top 10 list of software metric relations [Boe87].

- Dep.3 ("Understand simple"): Questions on project planning problems that require simple application of the provided PMT models, addressing trade-off effects between no more than two model variables.
- Dep.4 ("Understand complex"): Questions on project planning problems addressing trade-off effects between more than two variables, and questions on planning problems that may require re-planning due to alterations of project constraints (e.g. reduced manpower availability, shortened schedule, or changed requirements) during project performance.

13.3.5.3 Disturbing Factors

The values of the three potentially disturbing factors DiF.1, DiF.2, and DiF.3 are also derived from questionnaires that all subjects have to fill in. The questionnaire for DiF.1 will be filled in before the pre-test, the questionnaires for DiF.2 and DiF.3 will be filled in after the post-test.

The contents of the questionnaires are as follows:

- DiF.1: Questions about personal characteristics (age, gender), university education (year, major, minor), practical software development experience, software project management literature background, and preferred learning style.
- DiF.2: Questions on actual time consumption per scenario block, and on perceived time need.
- DiF.3: Questions on personal judgement of the training session (subjective session evaluation).

13.3.6 Experimental Procedure

The experiment was conducted following the plan presented in Table 45. After a short introduction during which the purpose of the experiment and general organisational issues were explained, data on the background characteristics (variable DiF.1) was collected with the help of a questionnaire. Then the pre-test was conducted and data on all dependent variables (Dep.1 through Dep.4) was collected, again using questionnaires. Following the pre-test, a brief introduction into organisational issues related to the treatments was given. After that, the subjects were randomly assigned to either the experimental or control group. Then each group underwent its specific treatment. After having concluded their treatments, both groups passed the post-test using the same set of questionnaires as during the pre-test, thus providing data on the dependent variables for the second time. Finally, the subjects got the chance to evaluate the training session by filling in another questionnaire, providing data on variables DiF.2 and DiF.3. The time frames

reserved for passing a certain step of the schedule was identical for the experimental and control groups.

The experiment was performed on two days following the schedule presented in Table 45. On the first day, the steps "Introduction to experiment", "Background characteristics", and "Pre-test" were conducted, consuming a total of 40 minutes. On the second day, the steps "Introduction to treatments", "Random assignment of students to groups", "Treatment", "Post-test", and "Time need & subjective session evaluation" were conducted, consuming a total of 90 minutes. Of the 12 students that agreed to participate in the experiment, 9 students participated in both pre-test and post-test. 5 subjects were assigned randomly to the experimental group (A), and 4 students to the control group (B).

Introduction to experiment	5 min
Background characteristics	5 min
Pre-test	
• Interest	3 min
• Knowledge about empirical patterns	5 min
• Understanding of simple project dynamics	10 min
• Understanding of complex project dynamics	12 min
Introduction to treatments	5 min
Random assignment of subjects to groups	5 min
Treatment	45 min
Post-test	
• Interest	3 min
• Knowledge about empirical patterns	5 min
• Understanding of simple project dynamics	10 min
• Understanding of complex project dynamics	12 min
Time need & subjective session evaluation	5 min
Total	130 min

Table 45: Schedule of experiment

13.3.7 Data Collection Procedure

The raw data for dependent variables Dep.1 to Dep.4 were collected during pre-test and post-test with the help of questionnaires.

The values for variable Dep.1 ("Interest") are average scores derived from five questions on the student's opinion about the importance of software project management issues (i) during university education and (ii) during performance of industrial software development projects, applying a five-point Likert-type scale [Lik32]. Each answer in the questionnaire is mapped to the value range R = [0, 1] assuming equidistant distances between possible answers, i.e. "fully disagree" is encoded as "0", "disagree" as "0.25", "undecided" as "0.5", "agree" as "0.75", and "fully agree" as "1". All

questions were formulated in a way that positive attitude towards project management education and application of project management techniques in projects must be expressed by ticking the fields "agree" or "fully agree".

The values for variables Dep.2 ("Knowledge"), Dep.3 ("Understand simple"), and Dep.4 ("Understand complex") are average scores derived from five (for Dep.2), seven (for Dep.3), and six (for Dep.4) questions in multiple-choice style. The answers to these questions were evaluated according to their correctness, thus having a binary scale with correct answers encoded as "1", and incorrect answers encoded as "0".

The raw data for disturbing factors DiF.1 to DiF.3 were collected before pre-test (DiF.1) and after post-test (DiF.2 and DiF.3).

In order to determine the values of factor DiF.1 ("Personal background") information on gender, age, number of terms studied, subjects studied (major and minor), personal experience with software development, and number of books read about software project management was collected. In order to simplify the analysis, the actual values for factor DiF.1 eventually used for the statistical analysis were exclusively based on the normalised average scores derived from six questions on the student's personal experience with software development. Each of these questions could be answered with "yes" (encoded as "1") or "no" (encoded as "0"). "Yes" indicated that a certain type of experience was present, and "no", that it was not present. Therefore, simple adding of the scores per answer gives a measure of experience with a maximal score of 6 and a minimal score of 0. Dividing by the number of questions provides a normalised value range, i.e. range R = [0, 1].

The values for factor DiF.2 are normalised average scores reflecting the "time need" for reading and understanding of the scenario blocks 1, 3, and 4, for familiarisation with the supporting tools, and for filling in the post-test questionnaire. For group A, the variable DiF.2' includes also scores related to scenario block 2. If a subject wants to express that more than the available time was needed related to a certain task, then "yes" (encoded as "1") should be marked, otherwise "no" (encoded as "0"). Adding the scores and dividing them by the number of tasks once again provides a normalised value range R = [0, 1], with "1" indicating time need for all tasks and "0" indicating the absence of time need.

The values for factor DiF.3 ("Session evaluation") are based on subjective measures reflecting the quality of the treatment. Again, for group A, the variable DiF.3' includes scores related to scenario block 2. The subjective perception of the treatment quality was evaluated with regard to four dimensions ("useful" versus "useless", "absorbing" versus "boring", "easy" versus "difficult", and "clear" versus "confusing") using five-point Likert-type scales, e.g. "extremely boring", "boring", "undecided", "absorbing", "extremely absorbing". Similar to variable Dep.1, possible answers were encoded as "0", "0.25", "0.5", "0.75", and "1" depending on whether

the subjective judgement was very negative, negative, undecided, positive, or very positive. By taking the average of the values for all four questions the values for disturbing factors could be mapped to range R = [0, 1].

13.3.8 Data Analysis Procedure

The conceptual model underlying the proposed statistical analysis is summarised in Figure 53. It is inspired by the work of Jac Vennix who conducted a similar experiment [Ven90], assuming that there are two separate effects on the dependent variables. On the one hand the effect of the independent variable, and on the other the effect of additional potentially disturbing factors.

In a first step, the statistical analysis applies a t-test to investigate the effect of the independent variable on the dependent variables. For testing hypothesis H_1, a one-way paired t-test can be used, because the data collected for this hypothesis is within-subjects, i.e. post-test scores are compared to pre-test scores of subjects within the same group [She97]. For testing hypotheses H_2 and H_3, repeated measures analysis could not be applied, thus the appropriate test was a one-sided t-test for independent samples, or, equivalently, a single factor, one-way ANOVA [She97].

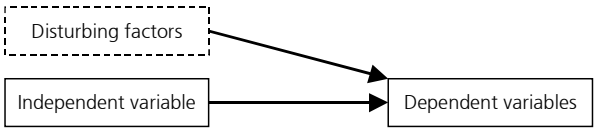


Figure 53: Relation between experimental variables

In addition to that, for testing hypotheses H_2 and H_3, analysis of covariance (ANCOVA) could be applied to improve the precision of the statistical analysis by removing potential bias due to disturbing factors [WiA78].

To conduct the analysis, a level of significance, i.e., the α-level, has to be specified. Several factors have to be considered when setting α. First, the implications of committing a Type I error, i.e., incorrectly rejecting the true null hypothesis, have to be determined. In the context of this study, it would mean the cost of building and using simulation models in student education without achieving any beneficial effect as compared to using conventional planning models. Second, the goals of the study have to be taken into account. This can be discussed from two perspectives [BBD+97]:

- From a practical perspective, which is asking for a judgement on whether it is likely that using SD simulation models for training has a better learning effect than using traditional planning models.
- From a scientific perspective, which is trying to identify cause-effect relationships between the type of the used planning model (with associated

training scenarios) and the learning effect, with a high level of confidence.

As previously stated, the empirical work presented in this paper should be considered as exploratory research whose goal is twofold: First, potentially interesting and practically significant trends shall be identified in order to focus future studies. Second, initial insights into what might be the consequences of using SD simulation models for student education shall be gained. Therefore, not a too stringent α level should be adopted, since this might result in overlooking potential areas of further investigation. In the study presented, $\alpha = 0.1$ was used. This can be seen as a compromise between a more practical perspective, and a strictly scientific perspective.

Another factor affecting the analysis procedure is the small sample sizes, which are likely to have an adverse effect on the power of the applied statistical methods, i.e. the chance that if an effect exists it will be found. The power of a statistical test is dependent on three different components: significance level α, the size of the effect being investigated, and the number of subjects. Low power will have to be considered when interpreting non-significant results. In these cases practical significance needs to be considered. Practical significance occurs when the effect being investigated impacts upon the dependent variables in a manner that can be considered practically meaningful. To determine if this is the case, the observed effect size (γ) detected for each dependent variable and for each hypothesis has to be calculated. Effect size is expressed as the difference between the means of the two samples divided by the root mean square of the variances of the two samples [She97]. For this exploratory study, effects where $\gamma \geq 0.5$ are considered to be of practical significance. This decision was made on the basis of the effect size indices proposed by Cohen [Coh88].

13.3.9 Experimental Results

Data was collected for nine subjects during pre-test and post-test. Therefore, 18 data points were available for each dependent variable and each disturbing factor – ten data points provided by the experimental group (A), and eight data points provided by the control group (B).

Table 46 and Table 47 show the raw data collected during pre-test and post-test together with the calculated values for mean, median, and standard deviation.

Table 48 shows the differences between post-test and pre-test scores together with the calculated values for mean, median, and standard deviation.

Dependent variables: pre-test scores				
Group A	**Dep.1**	**Dep.2**	**Dep.3**	**Dep.4**
Student #1	0.75	1	0.29	0.33
Student #2	0.9	0.4	0	0
Student #3	0.5	0.6	0.71	0.5
Student #4	0.8	0.2	0.29	0.67
Student #5	0.5	0.6	0.29	0.33
$Mean_{pre\text{-}test}$	**0.69**	**0.56**	**0.31**	**0.37**
$Median_{pre\text{-}test}$	**0.75**	**0.6**	**0.29**	**0.33**
$Stdev_{pre\text{-}test}$	**0.18**	**0.30**	**0.26**	**0.25**
Group B	**Dep.1**	**Dep.2**	**Dep.3**	**Dep.4**
Student #6	1	0.2	0.14	0.67
Student #7	0.8	0.8	0.29	0.17
Student #8	0.75	0.6	0.43	0.33
Student #9	0.7	0.4	0.86	0.17
$Mean_{pre\text{-}test}$	**0.81**	**0.5**	**0.43**	**0.33**
$Median_{pre\text{-}test}$	**0.78**	**0.5**	**0.36**	**0.25**
$Stdev_{pre\text{-}test}$	**0.13**	**0.26**	**0.31**	**0.24**

Table 46: Pre-test scores

Dependent variables: post-test scores				
Group A	**Dep.1**	**Dep.2**	**Dep.3**	**Dep.4**
Student #1	0.85	0.8	0.43	0.33
Student #2	1	1	0.71	0
Student #3	0.5	1	0.71	0.33
Student #4	0.85	0.8	0.71	0.83
Student #5	0.75	0.6	0.71	0.67
$Mean_{post\text{-}test}$	**0.79**	**0.84**	**0.66**	**0.43**
$Median_{post\text{-}test}$	**0.85**	**0.80**	**0.71**	**0.33**
$Stdev_{pre\text{-}test}$	**0.19**	**0.17**	**0.13**	**0.32**
Group B	**Dep.1**	**Dep.2**	**Dep.3**	**Dep.4**
Student #6	1	0.6	0.86	0.83
Student #7	0.85	0.6	0.86	0.33
Student #8	0.75	0.4	0.86	0
Student #9	0.55	0.8	0.71	0.67
$Mean_{post\text{-}test}$	**0.79**	**0.6**	**0.82**	**0.46**
$Median_{post\text{-}test}$	**0.80**	**0.60**	**0.86**	**0.50**
$Stdev_{post\text{-}test}$	**0.19**	**0.16**	**0.07**	**0.37**

Table 47: Post-test scores

Dependent variables: difference scores				
Group A	**Dep.1**	**Dep.2**	**Dep.3**	**Dep.4**
Student #1	0.10	-0.20	0.14	0
Student #2	0.10	0.60	0.71	0
Student #3	0	0.40	0	-0.17
Student #4	0.05	0.60	0.43	0.17
Student #5	0.25	0	0.43	0.33
$Mean_{difference}$	**0.10**	**0.28**	**0.34**	**0.07**
$Median_{difference}$	**0.10**	**0.40**	**0.43**	**0.00**
$Stdev_{difference}$	**0.09**	**0.36**	**0.28**	**0.19**
Group B	**Dep.1**	**Dep.2**	**Dep.3**	**Dep.4**
Student #6	0	0.40	0.71	0.17
Student #7	0.05	-0.20	0.57	0.17
Student #8	0	-0.20	0.43	-0.33
Student #9	-0.15	0.40	-0.14	0.50
$Mean_{difference}$	**-0.03**	**0.10**	**0.39**	**0.13**
$Median_{difference}$	**0.00**	**0.10**	**0.50**	**0.17**
$Stdev_{difference}$	**0.09**	**0.35**	**0.38**	**0.34**

Table 48: Difference scores

Disturbing factors					
Group A	**DiF.1**	**DiF.2**	**DiF.2'**	**DiF.3**	**DiF.3'**
Student #1	0.4	0.8	0.67	0.44	0.52
Student #2	0.6	0.4	0.33	0.56	0.46
Student #3	0.4	0	0	0.38	0.54
Student #4	0.8	0	0	0.38	0.48
Student #5	0.2	1	1	0.31	0.5
$Mean_{dif}$	**0.48**	**0.44**	**0.4**	**0.41**	**0.5**
$Median_{dif}$	**0.4**	**0.4**	**0.33**	**0.38**	**0.5**
$Stdev_{dif}$	**0.23**	**0.46**	**0.43**	**0.09**	**0.03**
Group B	**DiF.1**	**DiF.2**		**DiF.3**	
Student #6	0.8	0.2		0.69	
Student #7	0.4	0.2		0.56	
Student #8	0.4	0.6		0.69	
Student #9	0.8	0.4		0.69	
$Mean_{dif}$	**0.6**	**0.35**		**0.66**	
$Median_{dif}$	**0.6**	**0.3**		**0.69**	
$Stdev_{dif}$	**0.23**	**0.19**		**0.06**	

Table 49: Disturbing factors

Table 49 shows the data collected for the disturbing factors together with the calculated values for mean, median, and standard deviation. As can be seen, students in the control group (B) on average had more experience with software development (DiF.1) than students in the experimental group (A). In addition, students in the control group expressed less need of additional time (DiF.2) for conducting the treatment and passing the tests than students in the experimental group. Finally, students in the control group on average perceived their treatment easier, clearer, more absorbing, and more useful (DiF.3) than the students in the experimental group.

13.3.9.1 Anomalies in the Data Set

Even though, assignment of students to groups A and B was done purely random, the average values for pre-test scores and disturbing factors were not evenly distributed. For example, the subjects in the control group (B) were more experienced with regard to actively developing software than the subjects in the experimental group (A). As will be discussed in Section 13.3.9.3 and Section 13.3.9.4 it was not possible in all cases to neutralise this potential bias by applying ANCOVA.

Another anomaly in the data set is the fact that five out of nine subjects performed worse during post-test than during pre-test with regard to at least one variable. This is particularly interesting for variables Dep.2, Dep.3, and Dep.4, because they relate questions for which the correct answers were provided through the training materials.

Particularly for variable Dep.4 ("Understanding of complex project dynamics") the large variance in the post-test scores of both groups is surprising. No real clue has yet been found to this phenomenon, neither from analysing the comments made by subjects in the debriefing questionnaire, nor from talking to some of the subjects after the experiment.

13.3.9.2 Hypothesis H_1

Table 50 and Table 51 show for each group (A and B) separately the results of testing the directional alternative hypothesis H_1: $Mean_{post\text{-}test} > Mean_{pre\text{-}test}$ using a one-tailed t-test for dependent samples. Column one represents the dependent variable, column two the size of the effect detected (γ), column three the degrees of freedom (df), column four the t-value, column five the critical value for $\alpha = 0.10$ which the t-value has to exceed to be significant, and column six provides the associated p value.

Group A					
Variable	γ	**df**	**t-value**	**Crit. $t_{0.90}$**	**p-value**
Dep.1	1.07	4	2.39	1.53	0.04
Dep.2	0.77	4	1.72	1.53	0.08
Dep.3	1.23	4	2.75	1.53	0.03
Dep.4	0.35	4	0.78	1.53	0.24

Table 50: Group A results for "post-test" vs. "pre-test"

By examining columns four and five of Table 50 one can see that significant results were achieved for dependent variables Dep.1, Dep.2, and Dep.3. Therefore, for group A, H_1 can be accepted for variables Dep.1 to Dep.3 but not for variable Dep.4. It is worth noting though that the value for dependent variable Dep.4 supports the direction of the hypothesis, however without showing an effect size of practical significance.

For replication purpose, using the same experimental design, a minimum number of 56 subjects is required to have a reasonable chance of achieving statistical significance with regard to variable Dep.4, i.e. one where the power of the test is approximately 0.8 (for $\alpha = 0.1$, and the calculated effect size γ). Recalling the fact that several subjects in group A obviously did not have enough time to read and understand those training materials in scenario block 3 that are particularly related to "complex project dynamics", improvement of the experimental procedure is required before repeating the experiment.

Table 51 shows that for group B (control group) H_1 can only be accepted for variable Dep.3 ("Understanding of simple project dynamics"). The post-test scores for Dep.3 are significantly larger than the pre-test scores. In contrast, the value for variable Dep.1 does not even support the direction of the alternative hypothesis. For replication purpose, given the medium effect size of variables Dep.2 and Dep.4, a minimum number of approximately 100, respectively 56 subjects will be required to provide the test with a power of 0.8 (with $\alpha = 0.1$).

Group B					
Variable	γ	**df**	**t-value**	**Crit. $t_{0.90}$**	**p-value**
Dep.1	**	3	-0.58	1.64	**
Dep.2	0.29	3	0.58	1.64	0.30
Dep.3	1.05	3	2.09	1.64	0.06
Dep.4	0.37	3	0.73	1.64	0.26

Table 51: Group B results for "post-test" vs. "pre-test"

13.3.9.3 Hypothesis H_2

Table 52 shows the results of testing the directional alternative hypothesis H_2: $Mean_{difference(A)} > Mean_{difference(B)}$ using a one-tailed t-test for independent samples.

Group A versus B					
Variable	**γ**	**df**	**t-value**	**Crit. $t_{0.90}$**	**p-value**
Dep.1	1.38	7	2.06	1.42	0.04
Dep.2	0.51	7	0.75	1.42	0.24
Dep.3	0.16	7	-0.23	1.42	**
Dep.4	0.23	7	-0.33	1.42	**

Table 52: Results for "performance improvement"

As can be seen, for significance level $\alpha = 0.1$, the performance improvement, i.e. the score difference between post-test and pre-test, of variable Dep.1 ("Interest") is significantly larger for the experimental group (A) as compared to the control group (B), and thus alternative hypothesis H_2 can be accepted. It can also be noted that the value of variable Dep.2 ("Knowledge of empirical patterns") supports the direction of the hypothesis showing a medium to large effect size. Again, for future replication of the experiment with a power of approximately 0.8, at least 36 subjects per group are needed for the given effect size and $\alpha = 0.1$.

Note that the values of variables Dep.3 and Dep.4 do not even support the direction of the alternative hypothesis.

Group A versus B							
Variable	**df**	**F**	**Crit. $F_{0.90}$**	**p-value**	**Covariate**	**Corr. means A ------ B**	
Dep.1	7	4.23	3.59	0.08	-	*0.10*	*-0.03*
Dep.1	6	3.16	3.78	0.13	DiF.1	0.09	-0.01
Dep.2	7	0.57	3.59	0.48	-	*0.28*	*0.10*
Dep.2	6	5.86	3.78	0.05	DiF.1	0.35	0.01
Dep.2	6	1.44	3.78	0.28	DiF.2	0.31	0.07

Table 53: ANCOVA results for "performance improvement"

In order to check whether the test results are robust with respect to disturbing factors, an analysis of covariance (ANCOVA) was conducted. Table 53 shows only results for which the implicit assumptions underlying ANCOVA were fulfilled, i.e. homogeneity of regression coefficients, and difference from zero of the coefficient of the disturbing variable. The assumptions were fulfilled only for three cases. In the first case, neutralising the impact of disturbing variable DiF.1 ("Personal background") on variable Dep.1 slightly

decreased the strength of the treatment effect (with reduced effect size γ = 0.67). In the second case, however, neutralisation of the impact of DiF.1 on variable Dep.2 strongly increased the effect of the treatment, i.e. the null hypotheses can be rejected at a significant level (with p = 0.05) and thus the alternative hypotheses can be accepted (with γ = 0.91). When neutralising the impact of the disturbing factor DiF.2 ("Time need") on variable Dep.2, again an improvement of the p-level can be observed, however to a smaller extent (with γ = 0.44).

13.3.9.4 Hypothesis H_3

Table 54 shows the results of testing the directional alternative hypothesis H_3: $Mean_{post\text{-}test(A)} > Mean_{post\text{-}test(B)}$ using a one-tailed t-test for independent samples.

For significance level α = 0.1, the post-test scores of variable Dep.2 ("Knowledge of empirical patterns") is significantly larger for the experimental group (A) as compared to the control group (B), and thus alternative hypothesis H_3 can be accepted. It can also be noted that the value of variable Dep.1 ("Interest") supports the direction of the hypothesis, however, only with a very small effect size. In this case, in a future replication of the experiment, for the given effect size and significance level α = 0.1, a power of approximately 0.8 could only be reached with more than 1000 subjects per group.

Group A versus B					
Variable	γ	**df**	**t-value**	**Crit. $t_{0.90}$**	**p-value**
Dep.1	0.01	7	0.02	1.42	0.49
Dep.2	1.45	7	2.16	1.42	0.03
Dep.3	1.53	7	-2.28	1.42	**
Dep.4	0.07	7	-0.11	1.42	**

Table 54: Results for "post-test performance"

The values for variables Dep.3 ("Understand simple") and Dep.4 ("Understand complex") do not even support the direction of the hypothesis.

Group A versus B							
Variable	**df**	**F**	**Crit. $F_{0.90}$**	**p-value**	**Covariate**	**Corr. means A ------ B**	
Dep.1	7	0.00	3.59	0.98	-	*0.79*	*0.79*
Dep.1	6	3.11	3.78	0.13	Dep.1 (pre)	0.84	0.72
Dep.4	7	0.01	3.59	0.92	-	*0.43*	*0.46*
Dep.4	6	0.08	3.78	0.78	Dep.4 (pre)	0.42	0.48

Table 55: ANCOVA results for "post-test performance"

As with hypothesis H_2, in order to check whether the test results are robust with respect to disturbing factors, an analysis of covariance (ANCOVA) was conducted. Again, Table 55 shows only results for which the implicit assumptions underlying ANCOVA were fulfilled. This happened only in two cases. In the first case, neutralising the impact of pre-test scores on variable Dep.1, considerably increased the strength of the treatment effect ($p = 0.13$), with an effect size of $\gamma = 0.79$ now reaching the level of practical significance. In the second case, however, neutralisation of the impact of pre-test scores on variable Dep.4 only marginally increased the effect of the treatment ($\gamma = 0.10$). Note that attempts to neutralise potential bias induced by disturbing factors DiF.1 to DiF.3 was unfeasible in all cases, since there was no significant difference from zero of the respective coefficient in the ANCOVA model.

13.3.9.5 Analysis Summary

The results of the statistical analysis can be grouped according to the degree of evidence in supporting the hypotheses. Three categories have been defined: strong support, i.e. the data shows statistical significance (at α level 0.10), weak support, i.e. the data shows practical significance ($\gamma \geq 0.5$), and no support, i.e. the data has neither statistical nor practical significance.

- Strong Support: Statistical and practical significance was obtained for variables Dep.1 (only group A), Dep.2 (only group A), and Dep.3 (groups A and B) in support of H_1 – "post-test scores versus pre-test scores". After elimination of disturbing factors, statistical and practical significance in support of hypotheses H_2 – "performance improvement" – and H_3 – "post-test performance" – was only obtained for variable Dep.2 ("Knowledge of empirical patterns").
- Weak Support: Practical significance was obtained for variable Dep.1 ("Interest") in support of hypotheses H_2 – "performance improvement" – and H_3 – "post-test performance". For hypothesis H_2, impact on variable Dep.1 is even statistically significant, if only the results of the one-tailed t-test (cf. Table 52) are considered.
- No Support: No practical significance was found for variable Dep.4 ("Understanding of complex project dynamics") in all three hypotheses H_1, H_2, and H_3, and for variable Dep.3 ("Understanding of simple project dynamics") in hypotheses H_2 and H_3.

13.3.10 Threats to Validity

This section discusses the various threats to validity of the study.

13.3.10.1 Construct validity

Construct validity is the degree to which the variables used in the study accurately measure the concepts they purport to measure. The following issues associated with construct validity have been identified:

1. The mere application of a SDM might not adequately capture the specific advantages of SDMs over conventional planning models, since it has often been claimed that model building – and not the application of an existing model – is the main benefit of SD simulation modelling [Lan95].
2. Interest in a topic and evaluation of a training session are difficult concepts that have to be captured with subjective measurement instruments. In order to keep this threat to validity as small as possible, in the study, the instruments for measuring variables Dep.1 and DiF.3 were derived from measurement instruments that had been successfully applied in a similar study [Ven90].
3. There are indications that the distinction between "simple dynamics" and "complex dynamics", as it was made for measuring variables Dep.3 and Dep.4 (cf. Section 13.3.5.2), was too simplistic.
4. It is difficult to avoid "unfair" comparison between SDMs and COCOMO, because there are obviously features coming with SDMs that per definition cannot be offered by COCOMO (e.g. simulation of parameter changes over time / on-the-fly modification of model assumptions, etc.).

13.3.10.2 Internal Validity

Internal validity is the degree to which conclusions can be drawn about the causal effect of the independent variable on the dependent variables. Potential threats include selection effects, non-random subject loss, instrumentation effect, and maturation effect.

1. A selection effect was tried to be avoided by random assignment of subjects. In addition, existing differences in ability between groups were captured by collecting pre-test scores and by measuring the level of experience of subjects through variable DiF.1. Any potential bias induced by differences in pre-test scores and experience were tried to be neutralised by using ANCOVA.
2. Non-random drop-out of subjects has been avoided by the experimental design, i.e. assignment of groups only on the second day of the experi-

ment, i.e. directly before the treatment, and not before the pre-test already on the first day of the experiment.

3. The fact that the treatments of group A and B were different in the number of scenario blocks involved and, as a consequence, in the time available to perform each scenario block, may have induced an instrumentation effect. The post-mortem evaluation of the experiment with regard to time need indicated that most subjects of group A did not have enough time to execute both scenario blocks 2 and 3. In addition, – even though this was tried to be avoided by careful design – the planning models used in both treatments might slightly differ in scope and handling.
4. A maturation effect could have been caused if subjects had been informed on the first day of the experiment, i.e. when passing the pre-test, that at the end of the experiment they will pass a post-test with exactly the same questions. Since this information was not given to the subjects, all materials were re-collected after the pre-test, and the treatment with post-test took place 48 hours after the pre-test, it can be assumed that a maturation effect did not occur.

13.3.10.3 External Validity

External validity is the degree to which the results of the research can be generalised to the population under study and other research settings. Two possible threats have been identified: subject representativeness and materials:

1. The subjects participating in the experiment were all computer science students at an advanced level. It can be expected that the results of the study are to some degree representative for this class of subjects. Any generalisation of the results with regard to education of novice students, or even with regard to training of software professionals should be done with caution.
2. Even when the training sessions are applied to students, adequate size and complexity of the applied materials might vary depending on previous knowledge about SDM development and COCOMO.

In any case, the point should be emphasised that the presented research at its current stage is exploratory of nature and just the first step of a series of experiments, which – after modification of the treatments and stepwise inclusion of subjects with different backgrounds – might yield more generalisable results in the future.

13.3.10.4 Reliability

Reliability is the degree to which the results of a measurement reflect the true score of the intended concept, e.g. "Interest in software project management issues". Reliability would be low if measurement results, e.g. the

response to an item in a questionnaire, mainly reflect some esoteric, random error that is due to differences between subjects in how they read and understand a particular question in the questionnaire.

With regard to variable Dep.1 ("Interest") sufficient reliability can be assumed because the questionnaire used is composed of questions that have been used successfully in a previous experiment [Ven90]. The questionnaire is presented in the Appendix. The measures of variables Dep.2 to Dep.4 were collected by objective measurement, i.e. each question has exactly one correct answer. Therefore, reliability of the related measurement results can be assumed.

13.3.11 Summary and Discussion of Results

This study investigated the effect of using the model GENSIM in software project management education of computer science and SE students. The treatment focused on problems of project planning and control. The performance of the students was analysed with regard to four dimensions, i.e., interest in the topic of project management (Dep.1), knowledge of typical project behaviour patterns (Dep.2), understanding of simple project dynamics (Dep.3), and understanding of complex project dynamics (Dep.4). This was done by comparing the test results of students who passed a training session using the GENSIM model to the test results of students who passed a training session using the COCOMO model. Even though the statistical results must be interpreted with caution, due to small number of subjects involved and several threats to construct and internal validity, the findings of the analyses showed several interesting trends.

First, the treatment involving GENSIM had a positive impact on the change of scores from pre-test to post-test for all four dependent variables. The effect was statistically significant for Dep.1 to Dep.3. For Dep.4 the power of the test seemed to be too low to be able to detect the effect at the set significance level $\alpha = 0.1$.

Second, the treatment involving the SDM achieved practical significance on performance improvement and post-test performance for variable Dep.1, and even statistical significance for variable Dep.2.

Though positive, the second result might be related to problems with internal validity of the experiment, i.e. the inclusion of the role-play (scenario block 2) exclusively for the experimental group (A). Inclusion of the role-play, on the other hand, imposed additional time pressure on the subjects in the experimental group, which might have resulted in low scores for questions related to dependent variables Dep.3 and – particularly – Dep.4.

In order to avoid speculations about the positive or negative effects of the threats to internal validity, the experimental design, and the treatments involved in the experiment need improvement.

With regard to enhancing the treatments, two issues are of relevance. First, more time has to be allowed particularly for executing scenario blocks 2 (PMT Role Play) and 3 (PMT Planning Models), and for the familiarisation with the simulation tool. Second, the experimental treatment, as it is now, does not yet fully exploits all potentially available features of a learning tool that SDM usage and model building can offer. This relates to the fact that SDMs not only make causal relationships explicit and allow for variation of the strength of the relationships, but also offer means to change the structure of these relationships and make the effects of such changes on project performance visible through simulation.

A closer look at the nature of the applied treatments also proposes an improved experimental design for future replications. The inclusion of a role-play (scenario block 2) in the experimental treatment (T_A) had two consequences. As mentioned before, the role-play provided explicit information on observed empirical patterns in software development projects to the subjects in group A, which were not given in such an explicit form to subjects in group B. This might explain why subjects in group A had clearly better scores for variable Dep.2 than subjects in group B. On the other hand, because they did not have to perform a role-play, subjects in group B had more time than subjects in group A (30 min instead of 15 min) to read and understand the information provided in scenario block 3 (PMT Planning Models). This might explain why subjects in group A did not have better scores for variables Dep.3 and Dep.4. A 2x2 factorial design (either crossed or nested) could help to better distinguish between the effects of having or not having a role-play and the effects induced by the nature of the PMT planning model.

14 Efficiency of IMMoS

The evaluation of the efficiency of IMMoS was based on a comparison between SDM development with IMMoS and SDM development without IMMoS (baseline). For the comparison, objective measurement data was fed into appropriately designed evaluation models. As a general prerequisite, it had to be assumed that all SDMs used in the analysis were suitable (cf. related discussion in Section 10). The fulfilment of this assumption has been shown in Section 13.

The PSIM project was used as to generate the baseline data. The RESIM and GENSIM projects were used to generate the data to be compared to the baseline.

14.1 GQM Plan for IMMoS Efficiency Evaluation

The definition and analysis of the models to evaluate the efficiency of IMMoS followed a GQM process.

14.1.1 Definition of Measurement Goal

The measurmeent goal was defined as follows:

Analyse *IMMoS*
with respect to *Efficiency*
for the purpose of *Evaluation*
from the point of view of the *SDM Developer*
in the context of *Software Organisations.*

14.1.2 Definition of Models

A GQM abstraction sheet was used to identify the metrics and the associated models needed to achieve the measurement goal.

14.1.2.1 Quality Factors

Three quality factors were identified:

- E: Effort for SDM building [person months]
- D: Duration of SDM building [calendar months]
- S: SDM size [number of levels]

Based on the quality factors, the following models were defined:

- Model E_{rel} = E / S
- Model D_{rel} = D / S

Note: SDM size was measured by counting the number of levels, and not the number of model equations. This was done because the number of model equations can always be reduced to the number of levels, and it is only a matter of stile how many equations are spent on rate equations, and equations to determine auxiliary variables. It is not clear, however, whether the chosen size measure would also be an appropriate approximation of model complexity. Probably, model complexity depends not only on the number of levels but also on the type of relationship between levels.

14.1.2.2 Variation Factor

One variation factor was identified:

- A: Application of IMMoS [yes / no]

Note: Other possible variation factors, e.g. experience and skills of SDM builder, availability of subject matter experts, have not been considered. It is expected, however, that the application of IMMoS tends to neutralise variation in SDM development skills and experience. Lacking availability of subject matter experts only impacts duration, not effort.

14.1.2.3 Baseline Hypotheses

The total duration of the PSIM project was 18 calendar months, from April 1994 to September 1995. The total project effort was about 15 person-months. This includes 12 person-months of effort for the SDM developer and about 3 person-months for the customer organisation (customer management, SE subject matter experts, and SDM user).

- E(PSIM) = 15 person months
- D(PSIM) = 18 calendar months
- S(PSIM) = 26 levels

The related models yield:

- E_{rel}(PSIM) = 0.58 person months / level
- D_{rel}(PSIM) = 0.69 calendar months / level

14.1.2.4 Impact Hypotheses

Two impact hypotheses were defined:

- H_E: if A = yes then $E_{rel} \downarrow$
- H_D: if A = yes then $D_{rel} \downarrow$

14.2 Evaluation of Impact on Duration

Table 56 summarises the evaluation results with regards to the impact of using IMMoS on the relative duration for SDM building. It presents for the SDMs PSIM (baseline), RESIM, and GENSIM:

- the required raw data (S and D), and
- the calculated relative duration (D_{rel}).

		PSIM (baseline)	RESIM	GENSIM
Raw data	Duration D [calendar months]	18	3	1.5
	Size S [number of levels]	26	17	40
Relative Duration	D_{rel} [calendar months / level]	0.69	0.18	0.04

Table 56: Evaluation of IMMoS impact on duration of SDM building

As the data shows, both RESIM and GENSIM were developed much faster than PSIM and thus H_D is supported. Even though one could argue that the development of PSIM took particularly long due to lack of experience of the SDM developer, and temporary unavailability of SE subject matter experts, the difference between the baseline and RESIM or GENSIM quite large. Even if the development time needed for PSIM were reduced by 50%, it would still be much larger than the development time needed for RESIM or GENSIM.

Regarding GENSIM it should be noted that the development was particularly fast because no communication of the SDM developer with SE subject matter experts or prospective SDM users, i.e. the computer science students, were required.

14.3 Evaluation of Impact on Effort

Table 57 summarises the evaluation results with regards to the impact of using IMMoS on the relative effort for SDM building. It presents for the SDMs PSIM (basleine), RESIM, and GENSIM:

- the required raw data, and
- the calculated relative effort.

		PSIM (baseline)	RESIM	GENSIM
Raw data	Effort E [person months]	15	2	2
	Size S [number of levels]	26	17	40
Relative Effort	E_{rel} [person months / level]	0.58	0.12	0.05

Table 57: Evaluation of IMMoS impact on effort consumption for SDM building

As the data shows, the development of both RESIM and GENSIM needed considerably less effort than the development of PSIM and thus H_E is supported. Again, it could be argued that there were learning effects on part of the SDM developer, which is not accounted for when just looking at the raw data provided. But again, even if the effort needed to develop PSIM were reduced by 50%, it would still be much larger than the effort needed to develop RESIM or GENSIM.

14.4 Potential Impact of IMMoS on PSIM

There are no indications that the development of PSIM would have taken longer or would have needed more effort if the IMMoS approach had been followed. On the contrary, it is probable that the goal-orientation of IMMoS would have avoided many of the iterations during the modelling process that were caused by re-defining the model purpose and user.

15 Summary and Outlook

The focus of this thesis is on simulation-based learning to support both strategic and project management in software organisations.

Simulation models are valuable tools for managers because they help them understand the effects of new technologies and policies on the performance of software development processes. Based on simulations, managers can explore and analyse potential improvements before implementation and empirical evaluation of the related process changes in a pilot project. In addition, quantitative simulation models can be used to support planning and control tasks.

This section summarises the results and contributions of the thesis (Section 15.1) and outlines future work (Section 15.2).

15.1 Results and Contributions

The main achievement of the research conducted lies in the design, application and validation of a framework for Integrated Measurement, Modelling, and Simulation (IMMoS).

The novelty of the IMMoS framework is twofold. Firstly, it enhances existing guidance for SDM development by adding a component that enforces goal-orientation, and by providing a refined process model with detailed descriptions of activities, products, and roles involved in SD modelling and simulation. Secondly, it describes how to integrate SD modelling with established static black-box and white-box modelling methods, i.e. goal-oriented measurement and descriptive process modelling.

This research provided results on theoretical level (methodology design), practical level (methodology application and implementation), and empirical level (methodology validation). The results can be summarised as follows:

1. Development of the IMMoS framework (theoretical work): Based on experience with applying the System Dynamics modelling method in an industrial software organisation (PSIM project), a framework for Integrated Measurement, Modelling and Simulation (IMMoS) has been developed. The IMMoS framework consists of four elements. The first element (Process Guidance) improves existing process guidance for SDM development by providing a detailed process model with precise definitions of roles, responsibilities, activities, and work products. The second element (Goal-Orientation) provides support for SDM goal definition by offering a goal definition template to the SDM modeller that specifies

role (i.e. SDM user), scope, purpose, dynamic focus, and environment of the SDM to be developed. In the current System Dynamics modelling method, this kind of guidance was lacking. The third element (Integration of Models) describes how existing knowledge of a software organisation, which is typically represented and stored in the form of static models (i.e. process, product and quality models) can be reused and integrated with SDMs. The fourth element (Integration of Methods) describes how the modelling activities conducted during the development of a SDM relate to process modelling and goal-oriented measurement. Particular focus has been put on the integration of SDM development with GQM, which is an established and widely applied method for goal-oriented measurement in software industry.

2. Action research, application, and implementation of the IMMoS framework (practical work): In order to explore potentialities and usefulness of System Dynamics simulation modelling in industrial software organisations, the PSIM model was developed. After the IMMoS framework had been developed (cf. item 1), it was applied in the development of two additional SDMs, i.e. RESIM and GENSIM. The RESIM model was developed in an industrial environment. The GENSIM model was developed in a research environment and integrated into a web-based training module for computer science students. In order to be able to offer a hypertext version of IMMoS, the IMMoS process model was implemented with the SPEARMINT tool. Based on this implementation, a web-based Electronic Process Guide for IMMoS (IMMoS-EPG) can be generated automatically.
3. Validation of the IMMoS framework (empirical work): The validation of the IMMoS framework was based on measurement data from two industrial cases studies (PSIM project and RESIM project) and one controlled experiment (involving the GENSIM model). The *effectiveness* of the IMMoS framework has been evaluated by comparing the suitability of PSIM, which was developed without IMMoS (baseline), to the suitability of RESIM and GENSIM, which were both developed with IMMoS. The *efficiency* of the IMMoS framework has been evaluated by comparing the resource consumption (time and effort) during development of PSIM, to the resource consumption during development of RESIM and GENSIM. Empirical evidence supports that SDM development with IMMoS is more efficient and at least as effective as SDM development without IMMoS.

With the IMMoS framework the field of software engineering has been enriched by a method that facilitates simulation-based learning for software managers in an effective and efficient way. The key contributions of IMMoS are the following:

1. Methodology: The IMMoS framework combines state-of-the-art measurement and modelling approaches in the field of software engineering with the System Dynamics simulation modelling method. In particular, the integration of goal-oriented SDM development with GQM modelling can be considered an enhancement of the existing GQM method towards "Dynamic GQM". With regards to systematic software process

improvement, the IMMoS framework serves as a vehicle for smoothly extending the yet comprehensive QIP/EF approach – which is based on empirical, experience-based learning – towards simulation-based learning. This will help reduce the risk of failure when introducing software engineering technologies or process changes in industrial environments.

2. Management: By providing guidance on building and using quantitative simulation models as a source for learning, the IMMoS framework supports managers in software organisations to better cope with the dynamic complexity of software processes and projects, and thus improving their decision-making capability.

15.2 Limitations and Future Work

As it often happens in industry-driven research, the results presented in this thesis can only mark a milestone within a work in progress. In the following, a summary of the potentialities of System Dynamics in the field of software engineering, and a brief discussion of the limitations of the conducted research work will be provided. By reflecting the limitations, and by relating the achievements of the thesis to the general potentialities of System Dynamics in software engineering, an outlook to future work will be given.

The use of SDMs offers several interesting perspectives for strategic and project management in software organisations. Five possible directions of simulation-based learning have been identified and discussed in the literature (cf. [WaP94] for a more detailed presentation):

1. Research towards a theory of software management: SDMs are both a formal statement of knowledge about the modelled reality and a source for generating new knowledge by conducting virtual experiments. Systematic experimentation with SDMs can be used to support theory-building in software engineering. Due to their high explanatory power, SDMs are particularly well-suited for studies that focus on software management issues (strategic level and project level).

2. Support for strategic decision-making: Policy investigation is the vocation of System Dynamics. A SDM can be used as an electronic laboratory where hypotheses about observed problems can be tested, and corrective policies can be experimented before they are implemented on the real system. Experience from applications in other fields than software engineering indicates that significant benefits can be drawn from introducing the use of simulation in the meetings of policy makers and from performing studies of systemic problems, which eventually result in new policy recommendations (i.e. process changes). Both applications require that System Dynamics has gained acceptance by top management. Past experience of consultants using the System Dynamics method has shown that obtaining credibility is a critical point. Possible ways to achieve this in the field of software engineering are the integration of the System Dynamics method into accepted state-of-the-art SPI approaches (e.g.

QIP/EF), and the introduction of SDM-based simulation as a standard technique for visualisation and decision support into managerial training.

3. Learning from past projects: Analysing a completed project is a common means for organisations to learn from past experience, and to improve their software development process. System dynamics can facilitate post-mortem analysis: by properly calibrating a model, it becomes possible to replay the project, diagnose management errors that arose, and investigate policies that would have supplied better results. To avoid having the organisation reproduce – and amplify – its past errors, Abdel-Hamid recommends finding optimal values of past projects by simulation, and recording these values for future estimation, instead of recording real values that reflect inefficient policies [Abd93a]. Note, however, that this implies that the simulation results supplied by the model can be relied upon, which requires a high degree of quantitative accuracy.

4. Support for project planning and control: In the early System Dynamics literature related to software project management, strongly divergent opinions were expressed upon the question whether SDMs can be used to plan and control on-going software projects. Aranda et al. wrote [AFO93]: "microworlds are appropriately used for training rather than for operational decision, because such models describe general rather than precise behaviours". Opposed to this opinion, Lin et al. [LeL91] and Abdel-Hamid [Abd93a] mentioned the monitoring of software projects as a beneficial use of SDMs. The question refers to the problem of the predictive accuracy of SDMs. Some authors argue that numerical precision is an irrelevant goal, due to the noise inherently present in systems involving the human factor; others claim being able to obtain accuracy within a range of 10 percent [Wei80][46]. Based on the research done in the scope of this thesis, two complementory ways to achieve adequate model accuracy can be identified. The first one is to combine SDM development with state-of-the-practice static quantitative modelling, i.e. goal-oriented measurement and modelling (GQM). The second one is to modularise SDMs such that new models can be constructed by largely reusing less complex SDM modules that have already been validated sufficiently well in previous applications.

5. Management education and training: Flight-simulator-type environments (microworlds) can be used to confront managers with realistic situations that they may encounter in practice, and allow them to develop experience without the risks incurred in the real world. The "fun" aspect of role-playing and microworld exploration makes the learning process attractive. The potential of simulation models for the training of managers in other domains than software engineering has long been recognised. It can be expected that SDM-based learning environments can play an important role in software management training, too.

[46] A similar degree of predictive accuracy was achieved with the PSIM model (cf. Section 4.6.6).

By discussing to what extent the achievements gained with IMMoS support the favourable development of the potentialities of System Dynamics simulation in software organisations, existing limitations of the IMMoS framework and thus room for future research can be identified.

This thesis focuses on application-oriented research. Therefore, the development of the IMMoS framework mainly relates to items 2-5 in the list of System Dynamics potentialities. In particular, the alignment of IMMoS with the QIP/EF paradigm and the integration of SDM development with process modelling and goal-oriented measurement are a step forward towards developing support for strategic decision-making, learning from past projects, and support for project planning and control in software organisations. Much work, however, has still to be done with respect to modularisation and reuse of SDMs (cf. item 4). IMMoS does not provide any guidance in this regard, e.g. design principles for developing elementary base models that can be plugged together and easily adapted to different environments. A starting point for enhancing the IMMoS Process Model could be the work of Tvedt who did some intial research in this direction [Tve95].

Another methodological limitation of IMMoS is the yet insufficient guidance on how to combine empirical learning (i.e. conducting pilot projects in industrial environments) with simulation-based learning from the perspective of cost-effectiveness. This issue should be addressed in a follow-up industrial research project that is specifically dedicated to cost-benefit analysis of simulation-based SPI. The main objective of this project would be the clarification of the circumstances under which the costs for SDM development and running simulations are smaller than the cost of failure associated with conducting pilot projects without prior simulation-based analysis of the intended process changes.

In addition to methodological enhancements of IMMoS, there is also need to continue with empirical validation, either as part of the research project sketched in the previous paragraph, or in independent applications of IMMoS. Even though first empirical evidence supports the effectiveness and efficiency of the IMMoS framework, more experiments and industrial case studies are needed to underpin the results.

A very promising new area of research is currently emerging in the field of management education and training. The integration of simulation technology into web-based learning environments could be a successful pathway to increased effectiveness of work-based individual and collaborative learning in software organisations, and thus help mitigate the huge lack of software engineers and managers in an ever faster changing world.

References

[Abd90] Abdel-Hamid TK, "Investigating the Cost/Schedule Trade-Off in Software Development", *IEEE Software*, pp. 97-105, Jan. 1990.

[Abd93a] Abdel-Hamid TK, "Adapting, Correcting and Perfecting Software Estimates: a Maintenance Metaphor", *IEEE Computer*, pp. 20-29, March 1993.

[Abd93b] Abdel-Hamid TK, "Thinking in Circles", *American Programmer*, pp. 3-9, May 1993.

[AbM91] Abdel-Hamid TK, Madnick SE, "Software Projects Dynamics – an Integrated Approach", Prentice-Hall, 1991.

[ASR93] Abdel-Hamid TK, Sengupta K, Ronan D, "Software Project Control: An Experimental Investigation of Judgement with Fallible Information", *IEEE Trans. on Software Engineering*, pp. 603-612, Vol. 19, No. 6, June 93.

[AAF+01] Acuña ST, de Antonio A, Ferré X, López M, Maté L, "The Software Process: Modelling, Evaluation and Improvement", to appear in: *Handbook of Software Engineering and Knowledge Engineering*, World Scientific Publishing, 2001.

[ACM90] Ambriola V, Ciancarini P, Montangero C, "Software Process Enactment in Oikos", *Proceedings of the 4th ACM SIGSOFT Symposium on Software Development Environments*, Irvine, Dec. 1990.

[AFO93] Aranda RR, Fiddaman T, Oliva R, "Quality Microworlds: modeling the impact of quality initiatives over the software product life cycle", American Programmer, May 1993, pp. 52-61.

[Arg83] Argyris C, "Action Science and Intervention", *The Journal of Applied Behavioral Science*, Vol. 19, No. 2, pp. 115-140, 1983.

[Arg85] Argyris C, *Strategy, Change, and Defence Routines*, Pitman, 1985.

[ArS78] Argyris C, Schön D, *Organizational Learning: A Theory of Action Perspective*, Addison-Wesley, 1978.

[ABG+92] Armenise P, Bandinelli S, Ghezzi C, Morzenti A, "Software Process Representation Languages: Survey and Assessment", *Proceedings of the Fourth International Conference on Software Engineering and Knowledge Engineering (SEKE)*, Capri (Italy), June 1992.

[ArK94] Armitage JW, Kellner MI: "A Conceptual Schema for Process Definitions and Models". *Proceedings 3rd Int'l Conference on the Software Process (ICSP)*, IEEE Computer Soc., pp. 153-165, 1994.

[Axe76] Axelrod R, *The Structure of Decision: The Cognitive Maps of Political Elites*, Princeton, Princeton University Press, 1976.

[BFG92] Bandinelli S, Fuggetta A, Ghezzi C, "Software Processes as Real Time Systems: A case study using High-Level Petri nets", *Proceedings of the International Phoenix conference on Computers and Communications*, Arizona, April 1992.

[BFL+95] Bandinelli S, Fuggetta A, Lavazza L, Loi M, Picco GP: "Modeling and Improving an Industrial Software Process". *IEEE Trans. on Software Engineering*, Vol. 21, No. 5, pp. 440-453, May 1995.

[BFL+92] Barbieri A, Fuggetta A, Lavazza L, Tagliavini M, "DynaMan: a Tool to Improve Software Process Management through Dynamic Simulation", *Proceedings of Fifth International Workshop on Computer-Aided Software Engineering (CAiSE)*, Montreal, 6-10 July 1992.

[BaK91] Barghuti N, Kaiser G, "Scaling up Rule-Based Software Development Environments", *Proceedings of the 3rd European Software Engineering Conference (ESEC)*, Milan, Italy, Oct. 1991.

[Bar85] Barlas Y, *Validation of System Dynamics Models with a Sequential Procedure Involving Multiple Quantitative Methods*, PhD Thesis, Georgia Institute of Technology, UMI Dissertation Services, 1985.

[Bar89] Barlas Y, "Multiple Tests for Validation of System Dynamics Type of Simulation Models", *European Journal of Operational Research* 42, pp. 59-87, 1989.

[Bar94] Barlas Y, "Model Validation in System Dynamics", *Proceedings of the Int'l System Dynamics Conference*, Stirling, Scotland, Methodological Issues Vol., pp. 1-10, 1994.

[Bas89] Basili VR, "Software Development: A Paradigm for the Future", *Proceedings 13th Annual International Computer Software & Applications Conference (COMPSAC)*, Keynote Address, Orlando, FL, September 1989.

[Bas92] Basili VR, "The Experimental Paradigm in Software Engineering", in: Rombach HD, Basili VR (eds.), *Experimental Software Engineering Issues: Critical Assessment and Future Directions*, LNCS 706, Springer-Verlag, pp. 3-12, 1992.

[Bas93] Basili VR, "Applying the Goal/Question/Metric Paradigm in the Experience Factory", *Proceedings of the 10th Annual CSR Workshop*, Oct. 1993.

[BaC95] Basili VR, Caldiera G, "Improve Software Quality by Reusing Knowledge and Experience", *Sloan Management Review*, Fall 1995.

[BCR94a] Basili VR, Caldiera G, Rombach HD, "Experience Factory", in: Marciniak JJ (ed.), *Encyclopedia of Software Engineering*, Vol. 1, pp. 469-476, John Wiley & Sons, 1994.

[BCR94b] Basili VR, Caldiera G, Rombach HD, "Goal Question Metric Paradigm", in: Marciniak JJ (ed.), E*ncyclopedia of Software Engineering*, Vol. 1, pp. 528-532, John Wiley & Sons, 1994.

[BaW84] Basili VR, Weiss DM, "A Methodology for Collecting Valid Software Engineering Data", *IEEE Transactions on Software Engineering*, Vol. SE-10, No. 6, pp. 728 – 738, 1984.

[BHV97] Becker U, Hamann D, Verlage M, "Descriptive Modeling of Software Processes", *Proceedings of the 3rd Conference on Software Process Improvement*, Barcelona, Spain, December 1997.

[BeW97] Becker U, Webby R, *Towards a Comprehensive Schema Integrating Software Process Modeling and Software Measurement*, Technical Report IESE-021.97/E, Fraunhofer IESE, Kaiserslautern, 1997.

[BeB00] Becker-Kornstaedt U, Belau W, "Descriptive Process Modeling in an Industrial Environment: Experience and Guidelines", *Proceedings of the 7th European Workshop Software Process Technology (EWSPT)*, pp. 176-189, 2000.

[BHK+99] Becker-Kornstaedt U, Hamann D, Kempkens R, Rösch P, Verlage M, Webby R, Zettel J, "Support for the Process Engineer: The Spearmint Approach to Software Process Definition and Process Guidance", in: Jarke M, Oberweis A (eds.), *CaiSE'99*, LNCS 1626, Springer-Verlag, pp. 119-133, 1999.

[BEM91] Belkhatir N, Estublier J, Melo W, "Adele 2. An Approach to Software Development coordination", *Proceeding of the First European Workshop on Software Process Modeling*, Milan, Italy, May 1991.

[BDK+99] Bicego A, Dierks PP, Kuvaja P, Pfahl D, "Product Focused Process Improvement. Experiences of Applying the PROFES Improvement Methodology at DRÄGER", *Proceedings of the European Software Day at EUROMICRO*, Wien: Österreichische Computer Gesellschaft, 1999.

[Bir00] Birk A, *A Knowledge Management Infrastructure for Systematic Improvement in Software Engineering*, PhD Thesis, University of Kaiserslautern, 2000.

[BDH+98] Birk A, Derks P, Hamann D, Hirvensalo J, Oivo M, Rodenbach E, van Solingen R, Taramaa J, "Applications of measurement in product-focused process improvement: A comparative industrial case study", *Proceedings of the 5th International Symposium on Software Metrics (METRICS)*, Bethesda, MD, 20 - 21 Nov. 1998, IEEE Computer Society, 1998.

[BJK+98] Birk A, Järvinen J, Komi-Sirviö S, Kuvaja P, Oivo M, Pfahl D, "PROFES - A product driven process improvement methodology", *Proceedings of the European Conference on Software Process Improvement (SPI)*, 1-4 Dec. 1998, Monaco: John Herriot, 1998.

[BiS98] Birk A, Surmann D, "A seeded experience base on knowledge elicitation techniques", *Technical Report* IESE-061.98/E, Fraunhofer IESE, Kaiserslautern, 1998.

[Boe81] Boehm BW, *Software Engineering Economics*, Prentice Hall, 1981.

[Boe87] Boehm BW, "Industrial software metrics top 10 list", *IEEE Software*, pp. 84-85, September 1987.

[BAB+00] Boehm BW, Abts C, Brown WA, Chulani S, Clark BK, Horowitz E, Madachy R, Reifer DJ, Steece B, *Software Cost Estimation with COCOMO II*, Upper Saddle River: Prentice Hall PTR, 2000.

[Bos92] Bossel H, *Modellbildung und Simulation*, Vieweg, Braunschweig/Wiesbaden, 1992.

[BDR96] Briand LC, Differding CM, Rombach HD, "Practical Guidelines for Measurement-Based Process Improvement", *Software Process Improvement and Practice* 2 (4), pp. 253-280, 1996.

[BBD+97] Briand LC, Bunse C, Daly JW, Differding C, "An Experimental Comparison of the Maintainability of Object-Oriented and Structured Design Documents", *Empirical Software Engineering*, 2(3), pp. 291-312, 1997.

[BEL+97] Briand LC, El Emam K, Laitenberger O, Fussbroich T, *Using Simulation to Build Inspection Efficieny Benchmarks for Development Projects*, Technical Report IESE-041.97/E, Fraunhofer IESE, Kaiserslautern, 1997.

[BEF+98] Briand LC, El Emam K, Freimut B, Laitenberger O, *A Comprehensive Evaluation of Capture-Recapture Models for Estimating Software Defect Content,* Technical Report IESE-068.98/E, Fraunhofer IESE, Kaiserslautern, 1998.

[BEW99] Briand LC, El Emam K, Wieczorek I, "Explaining the Cost of European Space and Military Projects", *Proceedings of the 21st International Conference on Software Engineering (ICSE)*, Los Angeles, IEEE Computer Society Press, pp. 303-312, 1999.

[BLW97] Briand LC, Laitenberger O, Wieczorek I, "Building Resource and Management Models for Software Inspections", *Technical Report* ISERN-97-06, International Software Engineering Network, March 1997.

[Brö95] Bröckers A, "Process-based software risk assessment", *Proceedings of the 4th European Workshop on Software Process Technology*, Lecture Notes in Computer Science, No. 913 (W. Schäfer, ed.), Springer Press, pp. 9-29, 1995.

[Brö97] Bröckers A, *Modellbasierte Analyse von Softwar-Projektrisiken* (in German), PhD Thesis, University of Kaiserslautern, Shaker Verlag, 1997.

[BDT96] Bröckers A, Differding C, Threin G, "The role of software process modeling in planning industrial measurment programs", *Proceedings of the 3rd International Software Metrics Symposium (METRICS)*, Berlin, IEEE Computer Society Press, 1996.

[BLR+92] Bröckers A, Lott C, Rombach HD, Verlage M, "MVP-L Language Report", *Internal Report* 229/92, University of Kaiserslautern, AG Software Engineering, Dec. 1992.

[CaS99] Cartwright M, Shepperd M, "On building dynamic models of maintenance behaviour", in: Kusters R, Cowderoy A, Heemstra F, van Veenendaal E.(eds.), Project Control for Software Quality, Shaker Publishing, 1999.

[Chi93] Chichakly KJ, "The Bifocal Vantage Point: Managing Software Projects from a Systems Thinking Perspective", *American Programmer*, pp. 18-25, May 1993.

[Chr99] Christie AM, "Simulation: An Enabling Technology in Software Engineering", *CROSSTALK – The Journal of Defense Software Engineering*, pp. 2-7, April 1999.

[ChS00] Christie AM, Staley MJ, "Organizational and Social Simulation of a Requirements Development Process", Software Process Improvement and Practice 5, pp. 103-110, 2000

[Coh88] Cohen J, *Statistical Power Analysis for the Behavioral Sciences*, Academic Press, 2nd edition, 1988.

[CoM93] Cooper KG, Mullen T, "Swords and Ploughshares: the Rework Cycles of Defence and Commercial Software Development Projects", *American Programmer* 6 (5), 1993, pp. 41-51.

[Coy96] Coyle RG, *System Dynamics Modelling – A Practical Approach*, Chapman & Hall, 1996.

[CKO92] Curtis B, Kellner MI, Over J, "Process Modeling", *Communications of the ACM*, Vol. 35, No. 9, Sept. 1992.

[DeG94] Deiters W, Gruhn V, "The FunSoft Net Approach to Software Process Management", *International Journal of Software Engineering and Knowledge Engineering* 4 (2), pp. 229-256, 1994.

[Die92] Diehl EW, *MicroWorld Creator User's Guide*, MicroWorlds, Inc., Cambridge, MA, 1992.

[Die93] Diehl EW, "The analytical Lens Strategy-Support Software to Enhance Executive Dialog and Debate", *American Programmer*, pp. 26-32, May 1993.

[Die94] Diehl EW, "Managerial Microworlds as Learning Support Tools", in: Morecroft JDW, Sterman JD (eds.), *Modeling for Learning Organisations*, pp. 327-337, Productivity Press, Portland, 1994.

[Dör80] Dörner D, "On the Difficulties People have in Dealing with Complexity", *Simulations and Games* 11 (1), pp. 87-106, 1980.

[DrL99] Drappa A, Ludewig J, "Quantitative modeling for the interactive simulation of software projects", *Journal of Systems and Software* 46, pp. 113-122, 1999.

[Dyn91] *Professional DYNAMO Plus Reference Manual*, Pugh-Roberts Associates, Cambridge, 1991.

[EJS91] W. Emmerich, G. Junkermann, W. Schäfer, "MERLIN: knowledge-based process modeling", *Proceedings of the First European Workshop on Software Process Modeling (EWSP)*, Milan, Italy, May 1991.

[Fer93] Fernström C, "Process Weaver: Adding Process Support to UNIX", *Proceedings of the 2nd International Conference on the Software Process (ICSP)*, Berlin, 1993.

[FeP97] Fenton NE, Pfleeger SL, *Software Metrics: A Rigorous and Practical Approach*, International Thomson Computer Press, 2nd edition, 1997.

[Fey78] Fey WR, "An Industrial Dynamics Case Study", in *Managerial Applications of System Dynamics*, Roberts EB (ed.), MIT Press, pp. 117-138, 1978.

[FKN94] Finkelstein A, Kramer J, Nuseibeh B, *Software Process Modelling and Technology*, Research Studies Press, 1994.

[FlP63] Fletcher R, Powell MJD, "A rapidly convergent descent method for minimization", *Computing* 6, pp. 163-168, 1963.

[For61] Forrester JW, *Industrial Dynamics*, Productivity Press, Cambridge, 1961.

[For71] Forrester JW, *Principles of Systems*, Productivity Press, Cambridge, 1971.

[FuW96] Fuggetta A, Wolf A., *Software Process*, Chapter 1, John Wiley & Sons, 1996.

[Gra80] Graham AK, "Parameter Estimation in System Dynamics Modeling", in: Randers J (ed.), *Elements of the system dynamics method*, Productivity Press, Cambridge, pp. 143-161, 1980.

[Gra+92] Graham AK et al., "Model-supported case studies for management education", *European Journal of Operational Research* 59, pp. 151-166, 1992.

[GHW95] Gresse C, Hoisl B, Wüst J, "A Process Model for Planning GQM-based Measurement". *Technical Report*, STTI-95-04-E, Software Technology Transfer Initiative (STTI), University of Kaiserslauter, Oct. 1995.

[GrM96] Grcic B, Munitic A, "System Dynamics Approach to Validation", *Proceedings of the 1996 Int'l System Dynamics Conference*, Cambridge, Massachusettes, pp. 186-189, July 1996.

[HJO+98] Hamann D, Järvinen J, Oivo M, Pfahl D, "Experience with explicit modelling of relationships between process and product quality", *Proceedings of the European Conference on Software Process Improvement (SPI)*, 1-4 Dec. 1998, Monaco: John Herriot, 1998.

[HPJ+99] Hamann D, Pfahl D, Järvinen J, van Solingen R, "The Role of GQM in the PROFES Improvement Methodology", *Proceedings of the 3rd Conference on Quality Engineering in Software Technology (CONQUEST)*, Nürnberg, 1999.

[Ham80] Hamilton MS, "Estimating Lengths and Orders of Delays in System Dynamics Models", in: Randers J (ed.), *Elements of the system dynamics method*, Productivity Press, Cambridge, pp. 162-183, 1980.

[Har+88] Harel D, et al., "STATEMATE: A working environment for the development of complex reactive systems", *Proceedings of 10th International Conference on Software Engineering (ICSE)*, Singapur, 1988.

[HaP98] Harel D, Politi M, *Modelling Reactive Systems with Statecharts: The Statemate Approach*, McGraw-Hill, 1998.

[HeH00] Henderson H, Howard Y, "Simulating a Process Strategy for Large Scale Software Development using System Dynamics", *Software Process Improvement and Practice* 5, pp. 121-131, 2000.

[Hod92] Hodgson AM, "Hexagons for Systems Thinking", *European Journal of Operational Research* 59, no. 1, pp. 220-230, 1992.

[HuL88] Huff KE, Lesser VR, "A plan-based intelligent assistant that supports the software development process", *ACM SIGSOFT Software Engineering Notes* 13, No. 5, pp. 97-106, November 1988.

[Hog87] Hogart R, *Judgement and Choice*, Chichester, Wiley, 1987.

[HKL87] Huckfeldt RR, Kohfeld CW, Likens TW, *Dynamic Modeling. An Introduction*, Thousand Oaks, Sage Publications, 1982.

[Hum89] Humphrey W, *Managing the Software Process*, Addison Wesley, 1989.

[HuK89] Humphrey W, Kellner MI, "Software Process Modeling: Principles of Entity Process Models", *Proceedings of the 11th International Conference on Software Engineering (ICSE)*, pp. 331-342, 1989.

[IEEE91] IEEE Standard 1074-1991, *IEEE Standard for Developing Software Life Cycle Processes*, 1991.

[ISO95] ISO/IEC Standard 12207-1995, *ISO/IEC International Standard: Information Technology. Software Life Cycle Processes*, 1995.

[ISO98] ISO/IEC TR 15504, *Information Technology - Software Process Assessment - Parts 1-9. Technical Report Type 2*, International Organisation for Standardisation (ed.), Case Postale 56, CH-1211 Geneva, Switzerland, 1998.

[JBB94] Jeffery DR, Basili VR, Berry M, "Establishing Successful Measurement for Software Quality Improvement", *Proceedings of the TCSAUS IFIP Int'l Working Conference*, pp. 339-350, North-Holland, Amsterdam, 1994.

[JeB93] Jeffery DR, Berry M, "A Framework for Evaluation and Prediction of Metrics Program Success", *Proceedings of the 1st Int'l Software Metrics Symp. (METRICS)*, IEEE Computer Society Press, Los Alamitos, CA, pp. 28-39, 1993.

[KaM94] Kaposi AA, Myers M, *Systems, Models and Measures*, Springer-Verlag, London, 1994.

[Jon91] Jones C, *Applied Software Measurement – Assuring Productivity and Quality*, McGraw-Hill, Inc., 1991.

[Kel88] Kellner MI, "Modeling the Software Maintenance Process: Analytic Summary Models", *Proceedings of the Conference on Software Maintenance*, Phoenix, AZ, October 24-27, IEEE Computer Society, pp. 279-283, 1988.

[KeH89] Kellner MI, Hansen GA, "Software Process Modeling: A Case Study", *Proceedings of the 22nd Annual Hawaii International Conference on System Sciences*, Vol. II - Software Track, pp. 175-188, 1989.

[KMR99] Kellner MI, Madachy RJ, Raffo DM, "Software process simulation modeling: Why? What? How?", *Journal of Systems and Software* 46, pp. 91-105, 1999.

[KRS+00] Kempkens R, Rösch P, Scott L, Zettel J, "Instrumenting Measurement Programs with Tools", *Proceedings of the PROFES Conference*, Oulu, Finland, pp. 353-375, June 2000.

[Kle00] Klemm M, *Design and Implementation of a Scenario for Simulation-based Learning in the Domain of Software-Engineering*, Project Thesis, University of Kaiserslautern, 2000.

[KLN+92] Klingler CD, Lott CM, Nevasier M, Rombach HD, Marmor-Squires A, "A Case Study In Process Representation Using MVP-L", *Proceedings of IEEE COMPASS*, 1992.

[Lan93] Lane DC, "The road not taken: observing a process of issue selection and model conceptualization", *System Dynamics Review* 9, no. 3, pp. 239-264, Fall 1993.

[Lan95] Lane DC, "On a Resurgence of Management Simulation Games*", Journal of the Operational Research Society* 46, pp. 604-625, 1995.

[LSO+98] van Latum F, van Solingen R, Oivo M, Hoisl B, Rombach HD, Ruhe G, "Adopting GQM-Based Measurement in an Industrial Environment", *IEEE Software* January-February, pp.78–86, 1998.

[Leb00] Lebsanft K, "Das Siemens Process Assessment", in: Heinrich LJ, Häntschel I (eds.), *Evaluation und Evaluationsforschung in der Wirtschaftsinformatik*, Oldenbourg Verlag, 175-188, 2000.

[LeR99] Lehman MM, Ramil JF, "The impact of feedback in the global software process", Journal of Systems and Software 46(2/3), 1999, pp. 123-134.

[LeL91] Levary RR, Lin CY, "Modelling the Software Development Process Using an Expert Simulation System Having Fuzzy Logic", *Software – Practice and Experience*, pp. 133-148, Feb. 1991.

[LiC91] Liu C, Conradi R, "Process Modeling Paradigms: An Evaluation", *Proceeding of the First European Workshop on Software Process Modeling*, Milan, Italy, pp. 39-52, May 1991.

[Lik32] Likert R, "A technique for the measurmeent of attitude", *Archives of Psychology*, 22(140), 1932.

[Lin89] Lin CY, "Computer-Aided Software Development Process Design", *IEEE Trans. on Software Engineering*, pp. 1025-1037, Sept. 1989.

[Lin93] Lin CY, "Walking on Battlefields: Tools for Strategic Software Management", *American Programmer*, pp. 33-40, May 1993.

[LAS97] Lin CY, Abdel-Hamid TK, Sherif JS, "Software-Engineering Process Simulation Model (SEPS)", *Journal of Systems and Software* 38, pp. 263-277, 1997

[Lon93] Lonchamp J, "A structured conceptual and terminological framework for software process engineering", *Proceedings of the Second International Conference on Software Process*, pp. 41-53, February 1993.

[Lud+92] Ludewig J et al., "SESAM – Simulating Software Projects", *Proceedings of the Software Engineering and Knowledge Engineering (SEKE) Conference*, Capri, Italy, 1992.

[MaS80] Mass NJ, Senge PM: "Alternative Tests for Selecting Model Variables", in: Randers J (ed.), *Elements of the System Dynamics Method*, Productivity Press, Cambridge, pp. 205-225, 1980.

[Mad94] Madachy RJ, *A software project dynamics model for process, cost, schedule, and risk assessment*, PhD Thesis, University of Southern California, Los Angeles, 1994.

[Mad96] Madachy RJ, "System Dynamics Modeling of an Inspection-Based Process", *Proceedings of the 18th International Conference on Software Engineering (ICSE)*, Berlin, Germany, IEEE Computer Society Press, March 1996.

[MaT00] Madachy R, Tarbet D, "Case Studies in Software Process Modeling with System Dynamics", *Software Process Improvement and Practice* 5, pp. 133-146, 2000.

[MHH+94] Madhavji NH, Höltje D, Won-Kook H, Bruckhaus T, "Elicit: A method for eliciting process models", *Proceedings of the Third International Conference on the Software Process*, IEEE Computer Society Press, pp. 111-122, October 1994.

[McC95] McChesney IR, "Toward a classification scheme for software process modelling approaches", *Information and Software Technology* 37, No. 7, pp. 363-374, 1995.

[MMR+72] Meadows DH, Meadows DL, Randers J, Behrens WW, *The Limits to Growth – a Report for the Club of Rome's Project on the Predicament of Mankind*, Universe Books, New York, 1972.

[MMR92] Meadows DH, Meadows DL, Randers J, *Beyond the Limits*, Chelsea Green Publishing, Post Mills, VT, 1992.

[MMP+98] Mehner T, Messer T, Paul P, Paulisch F, Schless P, and Völker A, "Siemens Process Assessment and Improvement Approaches: Experiences and Benefits", *Proceedings of the 22nd Computer Software and Applications Conference (COMPSAC)*, Vienna, 1998.

[Mil95] Milling P, "Managementsimulation im Prozeß des Organisationalen Lernens" [Organisational Learning and its Support by Management Simulators], *Zeitschrift für Betriebswirtschaft* Ergänzungsheft 3/95: Lernende Unternehmen, pp. 93-112, March 1995. (Also available at URL http://iswww.bwl.uni-mannheim.de)

[MSF92] Möhring M, Strotmann V, Flache A, *MIMOSE - Einführung in die Modellierung, Sprachbeschreibung*, Koblenz, April 1992.

[Mor85] Morecroft JDW, "Rationality in the analysis of behavioral simulation models", *Management Science*, 31/7, pp. 900-916, 1985.

[Mor88] Morecroft JDW, "System dynamics and microworlds for policymakers", *European Journal of Operational Research* 35, pp. 301-320, 1988.

[NoT95] Nonaka I, Takeuchi H, *The knowledge-creating company*, New York, Oxford University Press, 1995.

[OfJ97] Offen RJ, Jeffery DR, "Establishing Software Measurement Programs", *IEEE Software*, pp. 45-53, March/April 1997.

[OZG91] F. Oquendo, J. Zucker, P. Griffiths, "The MASP approach to Software Process Description, Instantiation and Enaction", *Proceedings of the First European Workshop on Software Process Modeling*, Milan, Italy, May 1991.

[OyK88] Oyeleye OO, Kramer MA, "Qualitative Simulation of Chemical Process Systems: Steady-State Analysis", *AIChE Journal*, 34(9), p. 1441, 1988.

[PCC+93] Paulk MC, Curtis B, Chrissis MB, Weber CV, *Capability Maturity Model for Software Version 1.1*, Software Engineering Institute, Technical Report CMU/SEI-93-TR24, 1993.

[Pet75] Peterson DW, *Hypothesis, Estimation, and Validation of Dynamic Social Models*, PhD Thesis, Department of Electrical Engineering, MIT, Cambidge, 1975.

[Pet76] Peterson DW, "Parameter Estimation for System Dynamics Models*", Proceedings of the Summer Computer Simulation Conference*, Washington, 1976.

[Pet80] Peterson DW, "Statistical Tools for System Dynamics", in: Randers J (ed.), *Elements of the system dynamics method*, Productivity Press, Cambridge, pp. 226-245, 1980.

[Pfa94a] Pfahl D, *Modelling and Simulation of Projects and Processes - An Overview for Software and Engineering Practitioners*, Technical Report, No. Z0940669, Siemens AG, Munich, August 1994.

[Pfa94b] Pfahl D, *PSIM Bedienungsanleitung* (English: PSIM User Guide), Siemens AG, 1994.

[Pfa95] Pfahl D, "Software Quality Measurement Based on a Quantitative Project Simulation Model", *Proceeding of the European Software Cost Modelling Conference (ESCOM)*, Rolduc, The Netherlands, 15-17 May, 1995.

[Pfa97a] Pfahl D, *Description of Scenarios for the Usage of Goal-oriented Measurement, Modelling, and Simulation (MMS)*, Technical Report IESE-025.97/E, Fraunhofer IESE, Kaiserslauter, July 1997.

[Pfa97b] Pfahl D, *Prerequisites for the Application of Goal-oriented Measurement, Modelling, and Simulation (MMS)*, Technical Report IESE-031.97/E, Fraunhofer IESE, Kaiserslautern, September 1997.

[Pfa97c] Pfahl D, *A Selection of Appropriate Tools for Goal-oriented Measurement, Modelling, and Simulation (MMS)*, Technical Report IESE-032.97/E, Fraunhofer IESE, Kaiserslautern, September 1997.

[Pfa98a] Pfahl D, *Ein Vorgehensmodell zur Erstellung von System Dynamics-Modellen* (English: A Process Model for Developing System Dynamics Models), Technical Report IESE-004.98/D, Fraunhofer IESE, Kaiserslautern, February 1998.

[Pfa98b] Pfahl D, *Einsatzmöglichkeiten von System Dynamics Modellen* (English: Use Cases of System Dynamics Models), Technical Report IESE-009.98/D, Fraunhofer IESE, Kaiserslautern, January 1998.

[Pfa98c] Pfahl D, *Ein Vorgehensmodell zur Erstellung von System Dynamics-Modellen – Hilfsmittel* (English: A Process Model for Developing System Dynamics Models – Supporting Materials), Technical Report IESE-033.98/D, Fraunhofer IESE, Kaiserslautern, June 1998.

[PfB00] Pfahl D, Birk A, "Using Simulation to Visualise and Analyse Product-Process Dependencies in Software Development Projects*", Proceedings of the PROFES-2000 Conference*, Oulu, Finland, pp. 88-102, June 2000.

[PfK95] Pfahl D, Kammerer R, *Optimierung von Projekten und Prozessen in der Software-Entwicklung mit PROSYD* (English: Optimisation of Projects and Processes with PROSYD), Technical Report, No. Z0950040, Siemens AG, Munich, March 1995.

[PKR00a] Pfahl D, Klemm M, Ruhe G, "Using System Dynamics Simulation Models for Software Project Management Education and Training", *Proceedings of the 3rd Process Simulation Modelling Workshop* (ProSim-2000), London, United Kingdom, 12-14 June, 2000.

[PKR01a] Pfahl D, Klemm M, Ruhe G, "A CBT Module with Integrated Simulation Component for Software Project Management Education and Training", to appear in *Journal of Systems and Software*, October 2001.

[PKR01b] Pfahl D, Koval N, Ruhe G, "An Experiment for Evaluating the Effectiveness of Using a System Dynamics Simulation Model in Software Project Management Education", *Proceedings of the 7th International Software Metrics Symposium* (METRICS-2001), London, pp. 97-109, April 2001.

[PfL99] Pfahl D, Lebsanft K, "Integration of System Dynamics Modelling with Descriptive Process Modelling and Goal-oriented Measurement", *Journal of Systems and Software* 46, No. 2/3, pp. 135-150, 1999.

[PfL00a] Pfahl D, Lebsanft K, *Simulation of Software Development Systems – Taking an Integrated View on processes, Products, and People when Analyzing Software Projects*, Tutorial presented at the ESCOM-SCOPE-2000 Conference, Munich, Germany, 17 April, 2000.

[PfL00b] Pfahl D, Lebsanft K, "Using Simulation to Analyse the Impact of Software Requirement Volatility on Project Performance", *Information and Software Technology* 42, No. 14, pp. 1001-1008, 2000.

[PfL00c] Pfahl D, Lebsanft K, "Knowledge Acquisition and Process Guidance for Building System Dynamics Simulation Models: An Experience Report from Software Industry", *International Journal of Software Engineering and Knowledge Engineering* 10, No. 4, pp. 487-510, 2000.

[Pop68] Popper KR, *The logic of scientific discovery*, Hutchinson, London, 1968.

[PMB99] Powell A, Mander K, Brown D, "Strategies for lifecycle concurrency and iteration: A system dynamics approach", *Journal of Systems and Software* 46(2/3), 1999, pp. 151-162.

[PRO00] PROFES Consortium, *The PROFES User Manual*, Fraunhofer IRB, Stuttgart, Germany, 2000.

[Put78] Putnam LH, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem", *IEEE Trans. on Software Engineering*, pp. 345-361, July 1978.

[Ran73] Randers J, *Conceptualizing Dynamic Models of Social Systems: Lessons from a Study of Social Change*, PhD Thesis, Massachusetts Institute of Technology, 1973.

[Ran80a] Randers J (Ed.), *Elements of the system dynamics method*, Productivity Press, Cambridge, 1980.

[Ran80b] Randers J, "Guidelines for Model Conceptualization", in *Elements of the system dynamics method*, J. Randers (Ed.), Productivity Press, Cambridge, pp. 117-139, 1980.

[Ric90] Richardson GP, *Feedback Thought in Social Science and Systems Theory*, University of Pennsylvania Press, 1990.

[RiP81] Richardson GP, Pugh GL, *Introduction to System Dynamics Modeling and Dynamo*, MIT Press, Cambridge, 1981.

[Rob64] Roberts EB, "Research and Development Policy Making", *Technology Review* 66, no. 8, pp. 3-7, June 1964. Also reprinted in *Managerial Applications of System Dynamics*, Roberts (Ed.), MIT Press, Cambridge, pp. 283-292, 1978.

[Rob74] Roberts EB, "A Simple Model of R & D Project Dynamics", *R&D Management*, vol. 5, no. 1, October 1974. Also reprinted in *Managerial Applications of System Dynamics*, Roberts EB (Ed.), MIT Press, Cambridge, pp. 293-314, 1978.

[Rob78a] Roberts EB (Ed.), *Managerial Applications of System Dynamics*, MIT Press, Cambridge, 1978.

[Rob78b] Roberts EB, "Systems Dynamics – An Introduction", in: *Managerial Applications of System Dynamics*, Roberts EB (Ed.), MIT Press, Cambridge, pp. 3-36, 1978.

[RoW96] Rodrigues AG, Williams TM, "System Dynamics in Software Project Management: towards the development of a formal integrated framework", Research Paper, No. 96/5, University of Strathclyde, Glasgow, Scotland, 1996. (This paper was presented at the International System Dynamics Conference, Boston, MIT, July 1996.)

[RCH+00] Roehling ST, Collofello JS, Hermann BG, Smith-Daniels DE, "System Dynamics Modeling Applied to Software Outsourcing Decision Support", *Software Process Improvement and Practice* 5, pp. 169-182, 2000.

[RoV95] Rombach HD, Verlage M, "Directions in Software Process Research*", Advances in Computers* 41, pp. 1-63, 1995.

[Rug94] Ruge M, "A Graph-Theoretic Approach to Qualitative Simulation: Computer-Aided Technology Assessment Software", *Internal Report*, Dept. ZFE BT SE 5 TA, Siemens AG, May 1994.

[Ruh96] Ruhe G, "Qualitative Analysis of Software Engineering Data Using Rough Sets", *Proceedings of the Fourth International Workshop on Rough Sets, Fuzzy Sets, and Machine Discovery*, Tokyo, pp. 292-299, Nov. 1996.

[Rus98] Rus I, *Modelling the impact on project cost and schedule of software reliability engineering strategies*, PhD Thesis, Arizona State University, Tempe, 1996.

[RuC99] Rus I, Collofello J, Lakey P, "Software process simulation for reliability management", Journal of Systems and Software 46(2/3), 1999, pp. 173-182.

[Sch93] Schneider K, "Object-Oriented Simulation of the Software Development Process in SESAM", *Proceedings of the Object-Oriented Simulation Conference* (OOS'93), San Diego, Jan. 1993.

[Sch92] Schön D, "The Theory of Inquiry: Dewey's Legacy to Education", *Curriculum Inquiry* 22(2), pp. 119-139, 1992.

[Sen90] Senge PM, *The Fifth Discipline – the Art & Practice of the Learning Organization*, New York: Doubleday, 1990.

[She97] Sheskin DJ, *Handbook of Parametric and Nonparametric Statistical Procedures*, CRC Press, Boca Raton, 1997.

[Sim79] Simon HA, "Rational Decision Making in Business Organizations", *American Economic Review* 69, pp. 493-513, 1979.

[Sim82] Simon HA, *Models of Bounded Rationality*, Cambridge, MIT Press, 1982.

[SNV93] Smith BJ, Nguyen N, Vidale RF, "Death of a Software Manager: How to Avoid Career Suicide through Dynamic Software Process Modeling", *American Programmer*, pp. 10-17, May 1993.

[vSB99] van Solingen R, Berghout E, *The Goal/Question/Metric method: A practical guide for quality improvement of software development*, McGraw-Hill Publishers, 1999.

[Spe80] Spector P, "Ratings of Equal and unequal Response Choice Intervals", *The Journal of Social Psychology* 112, pp. 115-119, 1980.

[Ste90] *Stella II User's Guide*, High Performance Systems Inc, Hanover, 1990.

[Ste85] Sterman JD, "The Growth of Knowledge: Testing a Theory of Scientific Revolutions with a Formal Model", Technological Forecasting and Social Change 28(2), pp. 93-122, 1985.

[Ste94] Sterman JD, "Learning in and about Complex Systems", *System Dynamics Review*, 10 (2-3), pp. 291-330, 1994.

[Tan80] Tank-Nielsen C, "Sensitivity Analysis in System Dynamics", in *Elements of the system dynamics method*, J. Randers (Ed.), Productivity Press, Cambridge, pp. 187-204, 1980.

[TvC95] Tvedt JD, Collofello JS, "Evaluating the Effectiveness of Process Improvements on Development Cycle Time via System Dynamics Modeling". Proceedings of the Computer Science and Application Conference (COMPSAC), 1995, pp. 318-325.

[Tve96] Tvedt JD, *An Extensible Model for Evaluating the Impact of Process Improvements on Software Development Cycle Time*, PhD Thesis, Arizona State University, Tempe, 1996.

[VaD87] Vapenikova O, Dangerfield B, *DYSMAP2 User Manual*, University of Salford, 1987.

[Ven90] Vennix JAM, *Mental Models and Computer Models – design and evaluation of a computer-based learning environment for policy-making*, PhD Thesis, University of Nijmegen, 1990.

[Ver98] Verlage M, *Ein Ansatz zur Modellierung großer Software-Entwicklungsprozesse durch Integration unabhängig erfaßter rollenspezifischer Sichten* (in German), PhD Thesis, University of Kaiserslautern, Shaker verlag, 1998.

[Ven97] *Ventana Simulation Environment* (Vensim®) - Reference Manual, Version 3.0, Ventana Systems, Inc., 1997.

[Vis94] G. Visaggio, "Process Improvement Through Data Reuse", *IEEE Software*, pp. 76-85, July 1994.

[Völ94] Völker A, "Software Process Assessments at Siemens as a Basis for Process Improvement in Industry", *Proceedings of the ISCN*, Dublin, Ireland, 1994.

[WaP94] Waeselynck H, Pfahl D, "System Dynamics Applied to the Modelling of Software Projects", *Software Concepts and Tools* 15, No. 4, pp. 162-176, 1994.

[Wei80] Weil HB, "The Evolution of an Approach for Achieving Implemented Results from System Dynamics Projects", in *Elements of the system dynamics method*, J. Randers (Ed.), Productivity Press, Cambridge, pp. 271-291, 1980.

[Wie96] Wiegand M, *Prozesse Organisationalen Lernens*, Gabler, Wiesbaden, 1996.

[WiA78] Wildt AR, Ahtola OT, *Analysis of Covariance*, Sage University Paper Series on Quantitative Applications in the Social Sciences, series no. 07-012, Sage Publications, Newbury Park, 1978.

[Wol90] Wolstenholme EF, *System Enquiry: A System Dynamics Modelling Approach*, John Wiley & Sons, Chichester, 1990.

[Yin94] Yin RK, *Case Study Research, Design and Methods*, 2nd ed. Newbury Park, Sage Publications, 1994.

[YuM94] Yu ESK, Mylopoulos J, "Understanding "why" in software process modelling, analysis, and design", *Proceedings of the 16th International Conference on Software Engineering, IEEE Computer Society*, pp. 1-10, May 1994.

Appendix A: SD Model PSIM

PSIM Functionality

Basically, the PSIM simulation system provides two functions:

- Running a simulation.
- Analysis of simulation results.

A simulation can be run in interactive or batch mode. In interactive mode, the model user can stop the simulation after each individual simulation step and change a pre-defined set of parameters (exogenous model variables).

Running a Simulation

Before a simulation run can be started, an output file must be defined in which all subsequently generated simulation results are stored for potential future analysis.

To support running a simulation, PSIM provides a specifically tailored user interface, the so-called simulation cockpit (Figure 54). The simulation cockpit consists of three sectors for input parameters (variable project parameters), simulation control parameters, and output parameters.

A. Variable project parameters include:

- Planned work product size (for each phase separately), e.g. number of document pages, number of lines of code, number of test cases, etc.,
- Planned project duration (with milestones),
- Number of concurrent projects to which a team member is involved (on average),
- Number of available inspection rooms,
- Change request (time and size),
- Number of features,
- Estimated feature complexity,
- Number of test machines,
- Workforce (number of developers and testers), etc.

B. Simulation control parameters include:

- Selection of batch mode or interactive mode,
- Simulation step size (for interactive mode only),
- Switch to analysis of simulation results,

- Abort of simulation, etc.

C. Output parameters:

- Total project duration.
- For selected model variables (e.g. work product size, number of defects generated, time pressure, etc.) there is a specific pre-defined output graph for each project phase. Control buttons allow switching between different phases.

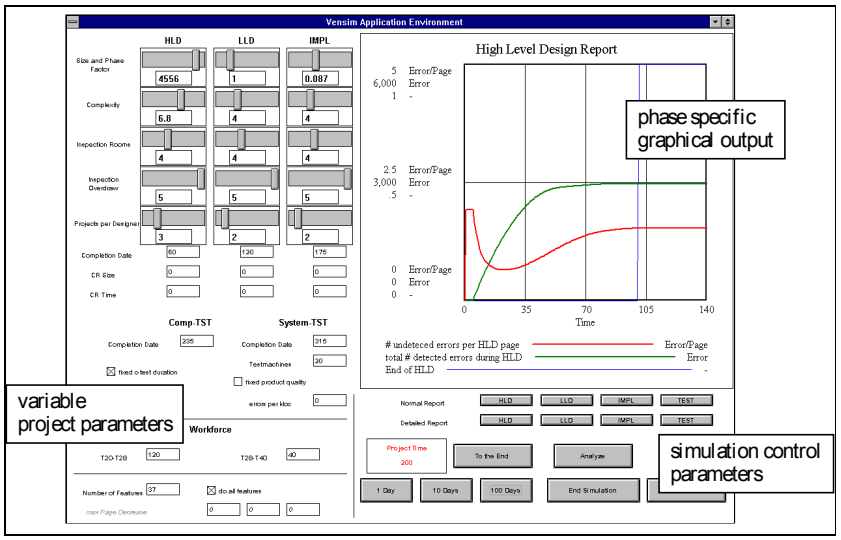


Figure 54: The PSIM simulation cockpit

Analysis of Simulation Results

A detailed analysis of simulation results is often necessary to better understand the specific behaviour of certain output parameters. Especially, when the results of a simulation do not coincide with the expectations of a model user, it is helpful to learn more about the complex interactions of the cause-effect relationships that forced a certain simulation output to be generated.

For the analysis of simulation results PSIM provides several analysis screens. Figure 55 provides an example of such a screen, showing the screen description, a list of choices for jumping into a different analysis screen, a graph with the causes tree of the selected model variable, and a graph that displays the behaviour of the selected variable.

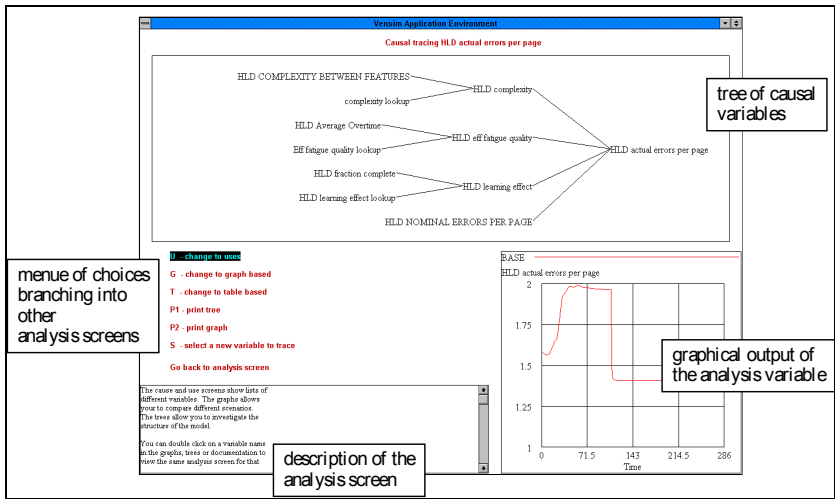


Figure 55: An example PSIM analysis screen

In total, the PSIM analysis screens provide the following functions:

- Selection of simulation runs (output files). If more than one simulation run is selected, graphical output data is displayed in different colours for better comparison of results.
- Selection of the model variable to be analysed in detail. Each individual variable contained in the model can be selected without any restriction.
- Representation of the behaviour of the selected variable as a graph or in a table (time series data).
- Representation of structural model information (causes tree and effects tree of the selected variable).
- Representations of the behaviour of all variables with direct effect on the selected variable (strip graph or table).
- Identification of structural difference between selected simulation runs.

Example PSIM Applications

The potential applications of PSIM were already mentioned in Section Section 4.2, namely project planning, project control, and process improvement. For the purpose of illustration, in the following sub-sections for each of these applications a fictitious example is sketched.

Project Planning

Under the assumption that the SDM contained in PSIM is a valid predictive model, a project manager might use PSIM for the purpose of project planning in the following way. For a set of given project parameters (e.g., man-

power, estimated product size, number of features, estimated feature complexity, etc.) he predicts the duration of the individual project phases and the overall product quality.

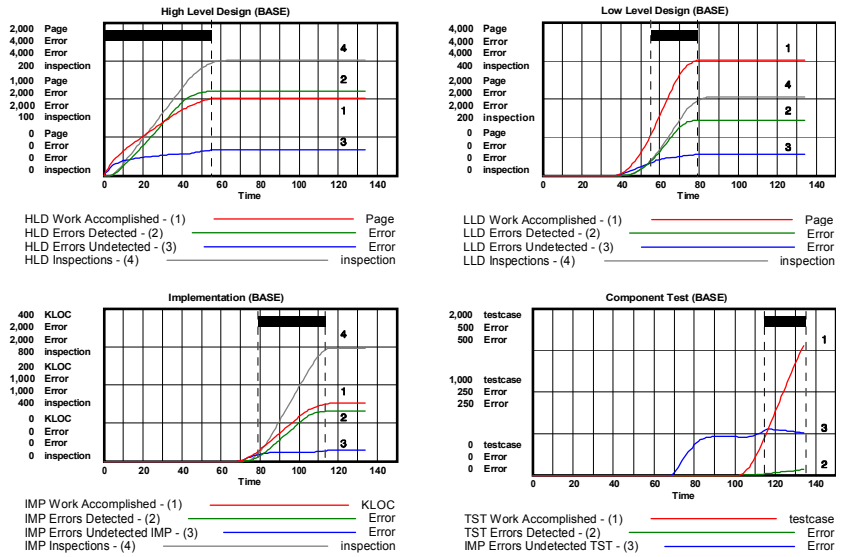


Figure 56: PSIM simulation result for project planning

Figure 56 shows the output of a simulation run generated from a project specific set of input parameters. For each of the four phases HLD, LLD, IMP and TST, product quality and other variables are displayed in individual graphs. The phase duration is highlighted with a black bar. Product quality is expressed in terms of the total number of undetected defects[47] (model variable: "Errors undetected") contained in the set of artefacts produced within a specific development phase, i.e. the set of high-level design documents, the set of low-level design documents, and the set of code modules.

Project Control

A possible application of PSIM for project control is the following: Assume that during the conduct of a project an unexpected change request increases the amount of work by 10% (expressed in terms of additional design documents). How should the project manager react to avoid time overrun and/or quality loss?

47 Please note that throughout this and the following sub-sections the terms "defect" and "error" are used as synonyms.

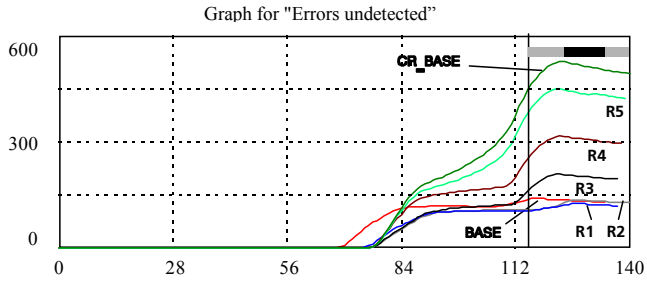


Figure 57: PSIM simulation result for project control

The curves in Figure 57 show the behaviour of the product quality indicator "Errors undetected" during the implementation and component test phases, as well as the overall project duration (assuming that the project stops at the end of the component test phase). The simulation run BASE indicates the behaviour without occurrence of a change request. The simulation run CR_BASE shows what happens if a change request occurs in phase HLD and the project manager does not change any of the parameters he is controlling. Now assume that the project manager, within certain limits, can vary two control parameters: workforce (manpower allocation), and the schedule planning for intermediate project milestones. The simulation runs R1 to R5 show the project performance for the case that both control parameters are altered simultaneously.

Simulation Run	Input (controlled by management)		Output (at end of phase TST / project end)			
	Project		Project	Product		
	Workforce [persons]	Milestone Planning [calendar weeks]	Duration [calendar weeks]	Quality (absolute) [undetected errors]	Size [KLOC]	Quality (relative) [undetected errors / KLOC]
BASE	40	(45, 80, 115, 134)	134	127	151	0.84
CR_BASE	40	(45, 80, 115, 134)	140	492	171	2.88
R1	48	*(60, 87, 125,* 134)	137	118	171	0.69
R2	45	(60, 86, 125, 137)	140	126	171	0.74
R3	45	*(60, 86, 117,* 134)	137	193	171	1.13
R4	42	(60, 80, 116, 135)	138	293	171	1.71
R5	41	*(60, 80,* 115, *136)*	139	421	171	2.46

Table 58: PSIM simulation result for project control (point estimates)

Table 58 provides the point estimates of all simulation runs. For the runs R1 to R5, changes in workforce allocation and milestone planning are printed in italics. The growth of product size (expressed in 1000 lines of source code), which is a direct consequence of the change request, as well as the overall project duration, and the absolute and relative product quality are clearly vis-

ible in this fictitious example. The project manager can use these and further simulation results to find an optimal policy. If the model variables representing project duration and product quality are appropriately combined in a weighted utility function, he can even use the optimisation feature of the SD modelling tool Vensim [Ven97] to perform an automated search for the best value assignment of the control parameters.

Process Improvement

PSIM can also be used to evaluate candidate process improvement strategies. Assume that an analysis of available data and discussions with project team members have revealed that formal inspections are performed sloppily under conditions of high time pressure, with the consequence of poor product quality. In this situation, project management suggests the implementation of a mechanism that enforces the correct conduct of formal inspections (e.g. by installing an adequate reward mechanism). However, before the intended improvement is implemented, management would like to test the suggested process change through simulation. The adherence to inspection guidelines can be controlled through model variables representing the number of inspections, and the average size of design documents or source code per inspection. If not all documents or code modules are inspected, or the inspections are done violating the recommended size constraints, then the inspections are considered as being conducted in an irregular manner (Process 1). If all documents and code modules are inspected, and the recommended size constraints are respected, then the inspection process is considered as being conducted correctly (Process 2).

Simulation Run	Input (at start of phase HLD / project start)			Output (at end of phase TST / project end)			
	Workforce [persons]	Planned project duration [days]	Planned product quality [undetected errors / KLOC]	Actual project duration [days]	Actual product quality [undetected errors / KLOC]	Time overrun compared to Baseline	Increase of product quality compared to Baseline
Process 1 / Baseline	fixed	340	-	363	9.2	-	-
Process 2	fixed	340	-	400	5.8	10 %	40 %
Process 1	fixed	340	5.8	473	5.8	30 %	40 %

Table 59: PSIM simulation result for process improvement

Table 59 shows the simulation results of a fictitious example, achieved with a differently calibrated model than in the previous subsections. The first row provides the original project performance of the unchanged process (Process 1 / Baseline). The project duration is 340 days, and the relative product quality is 9.2 undetected errors per KLOC. It can clearly be observed from the

second row in the table that the changed process (Process 2) yields an improved product quality of 5.8 undetected errors per KLOC. In addition, the simulation results indicate a time overrun of about 10% compared to the baseline. This delay, however, is small compared to the increase in product quality of more than 40%. If Process 1 had to achieve the same product quality as Process 2, a time overrun of almost 30% compared to the baseline would occur due to an extremely extended testing phase.

Appendix B: SD Model RESIM

Model Equations

Policy Variable

```
effort provided for systems engineering = GAME( 10 )
```

Levels

```
all SW requirements = INTEG( receive requ , 0)
Customer new requirements A = INTEG( new requ A , 0)
Customer new requirements B = INTEG( new requ B , 0)
Customer new requirements C = INTEG( new requ C , 0)
Effort in person weeks = INTEG( effort accumulation rate , 0)
Manpower SW = INTEG( mp adjustment , initial manpower )
Incr A duration = INTEG( A active , 0)
Incr A status = INTEG( - switch status A , 1)
Incr B duration = INTEG( B active , 0)
Incr B status = INTEG( - switch status B , 1)
Incr C duration = INTEG( C active , 0)
Incr C status = INTEG( - switch status C , 1)
SW development Cost = INTEG( cost accumulation rate , 0)
SW product = INTEG( sw development - replace requ , 0)
SW replace requ = INTEG( replace requ , 0)
SW requirements = INTEG( receive requ - sw development , 0)
Systems Engineering = INTEG( new requ - receive requ + replace requ ,
Requ start )
```

Rates

```
A active = Incr A status
B active = IF THEN ELSE ( Incr A status > 0, 0, Incr B status )
C active = IF THEN ELSE ( Incr B status > 0, 0, Incr C status )
cost accumulation rate = effort accumulation rate * cost per person week
effort accumulation rate = Manpower SW * project active
mp adjustment = IF THEN ELSE ( project active > 0, INTEGER ( manpower
difference / manpower adjustment time) , - Manpower SW )
new requ A = IF THEN ELSE ( A active > 0, INTEGER ( new requirements
fraction * Requ start / ( 1 + Incr A duration ) ^ 2) , 0)
new requ B = IF THEN ELSE ( B active > 0, INTEGER ( new requirements
fraction * Requ start / ( 1 + Incr B duration ) ^ 2) , 0) + switch status
A * Requ start * new requirements fraction * B multiplier
new requ C = IF THEN ELSE ( C active > 0, INTEGER ( new requirements
fraction * Requ start / ( 1 + Incr C duration ) ^ 2) , 0) + switch status
B * Requ start * new requirements fraction * C multiplier
receive requ = Systems Engineering * project active
replace requ = IF THEN ELSE ( SW product > 0 :AND: SW requirements > sw
development , INTEGER ( RANDOM UNIFORM ( 0, Requ start * weekly replace
factor , 0) ) , 0) * project active
sw development = IF THEN ELSE ( SW requirements > development rate ,
development rate, IF THEN ELSE ( SW requirements > 0, SW requirements ,
0) ) * project active
switch status A = IF THEN ELSE ( Incr A status > 0, stop flag A , 0)
```

```
switch status B = IF THEN ELSE ( Incr B status > 0, stop flag B , 0)
switch status C = IF THEN ELSE ( Incr C status > 0, stop flag C , 0)
```

Auxiliary Variables

```
actual all SW requirements = all SW requirements-SW replace requ
actual total muster duration = A active * Muster A duration + B active *
Muster B duration + C active * Muster C duration
Customer new requirements = Customer new requirements B + Customer new
requirements A + Customer new requirements C
development rate = INTEGER ( process nominal dev rate per person * Man-
power SW / EXP ( weekly replace factor ) )
estimated needed dev rate per week = A active * MAX ( Requ start + Cus-
tomer new requirements A , Requ start * ( 1 + new requirements fraction )
) * 3 / target time + B active * MAX ( Customer new requirements B , Requ
start * ( 1 + B multiplier ) * new requirements fraction ) * 3 / target
time + C active * MAX ( Customer new requirements C , Requ start * ( 1 +
C multiplier ) * new requirements fraction ) * 3 / target time + Requ
start * weekly replace factor / 2
filter costs = effort provided for systems engineering * cost per person
week
initial manpower = INTEGER ( manpower lookup ( Requ start ) )
manpower adjustment time = IF THEN ELSE ( ABS ( manpower difference ) >=
2, ABS ( manpower difference ) / 2, 1)
manpower difference = INTEGER ( target manpower - Manpower SW )
manpower lookup = ( [(0,0)
(10000,40)],(0,1),(200,2),(900,4),(1400,5),(3000,7), (7000,9),
(10000,10) )
new requ = new requ B + new requ A + new requ C
process nominal dev rate per person = A active * process nominal dev rate
A + B active * process nominal dev rate B + C active * process nominal
dev rate C
project active = IF THEN ELSE ( ( Muster A status > 0) :OR: ( Muster B
status > 0) :OR: ( Muster C status > 0) , 1, 0)
stop flag A = IF THEN ELSE ( Muster A duration > 0 :AND: SW requirements
< 1 :AND: Muster B duration < 1, 1, 0)
stop flag B = IF THEN ELSE ( Muster B duration > 0 :AND: SW requirements
< 1 :AND: Muster C duration < 1, 1, 0)
stop flag C = IF THEN ELSE ( Muster C duration > 0 :AND: SW requirements
< 1, 1, )
target manpower = MIN ( 3 * initial manpower , Manpower SW * time pres-
sure )
time pressure = IF THEN ELSE ( development rate > 0, estimated needed dev
rate per week / development rate , 1)
total cost = SW development Cost + filter costs
weekly replace factor = MIN ( 0.05, 1 / ( 2 + effort provided for systems
engineering) ^ 1.7)
```

Constants

```
B multiplier = 1.8
C multiplier = 0.5
cost per person week = 2000
new requirements fraction = 0.15
process nominal dev rate A = 11
process nominal dev rate B = 4
process nominal dev rate C = 2.5
Requ start = 1000
target time = 100
```

Appendix C: SD Model GENSIM

Quantitative Relationships between Key Variables (Examples)

Figure 58 shows the effect of *Unbalanced Average Manpower* (expressed as the ratio P-AMP / N-AMP) and *Manpower skill* on productivity (output per person-day). *Unbalanced Average Manpower* is expressed as the ratio of *Planned Average Manpower* (P-AMP) and *Nominal Average Manpower* (N-AMP), i.e. P-AMP/N-AMP (cf. Figure 39 in Section 12.2.2). The effect of Unbalanced Average Manpower is always negative, while the effect of *Manpower Skill* is positive when *Manpower Skill* is greater than "1".

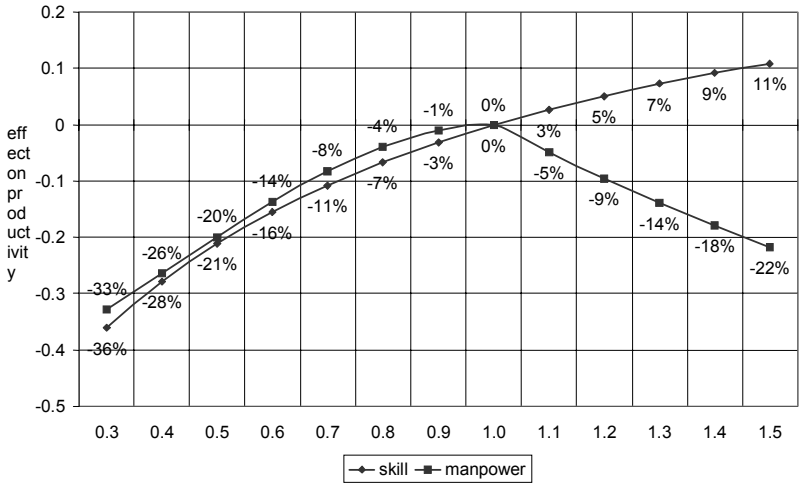


Figure 58: Impact of skill and unbalanced manpower on productivity

Figure 59 shows the effect of *Unbalanced Average Manpower* (expressed as the ratio P-AMP / N-AMP) and *Manpower skill* on defect injection (defects per size unit). The effect of Unbalanced Average Manpower is always positive, i.e. the qualitive decreases due to higher defect injection, while the effect of *Manpower Skill* is negative when *Manpower Skill* is greater than "1".

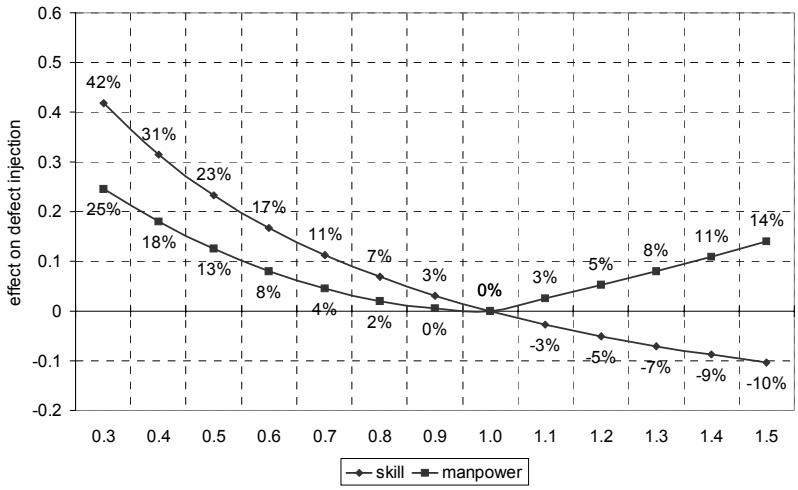


Figure 59: Impact of skill and unbalanced manpower on defect injection

Graphical User Interface (GUI)

Figure 60 shows the set of I/O windows provided by the GENSIM graphical user interface (GUI). There are three sets of windows available that can be offered to the simulation model user whenever appropriate. Each set is dedicated to a special purpose, namely input of data, output of simulation results, and analysis of simulation results.

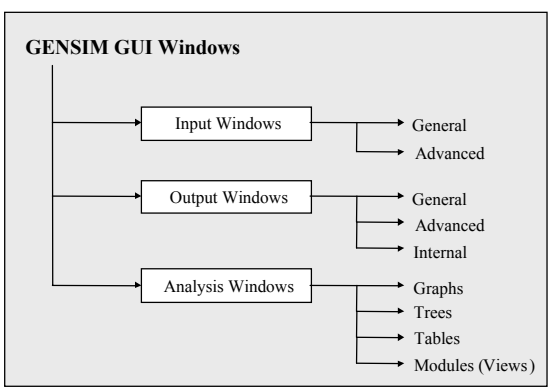


Figure 60: GENSIM GUI windows

Input Windows

Two different input windows are provided. The first one ("General") facilitates the input of mandatory project characterisation parameters and optional project management and QA technology-related parameters as listed in Table 38 (cf. Section 12.2.1). Figure 61 shows a screenshot of the "General" input window. In order to put in data either a slide bar or the data box on the right hand side of a slide bar can be used. If the box on the left hand side of a slide bar is ticked, then the pre-defined default values are used for the simulation.

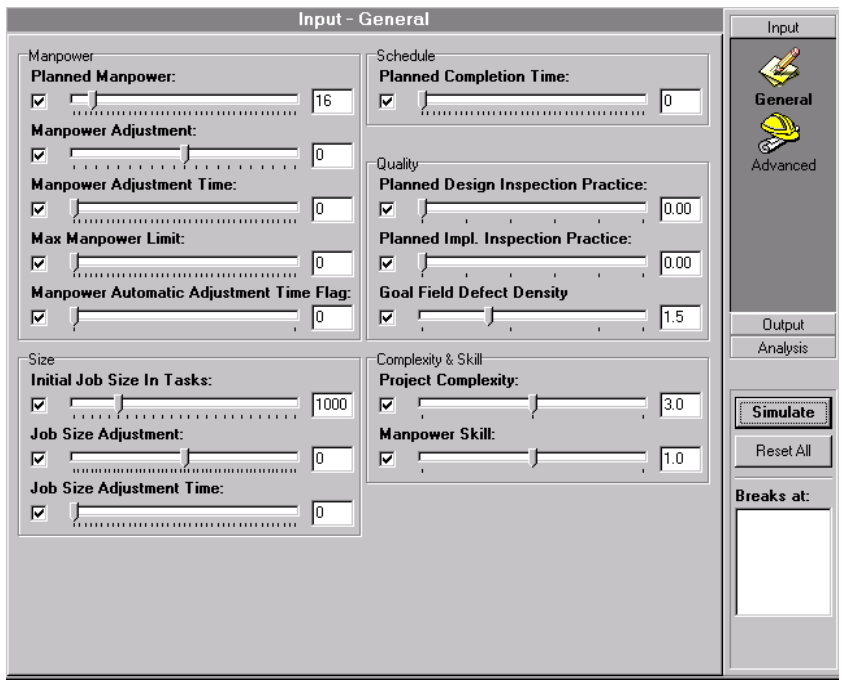


Figure 61: GENSIM input window "General"

The second input window ("Advanced") allows for adjustment of the software process parameters that determine the behaviour of the development processes instantiated during project performance. Typical process parameters include phase-specific nominal defect injection rate per code size unit, nominal inspection effort used per document size unit, nominal test effectiveness, and phase-specific nominal rework effort needed per defect. These process-related model parameters allow for calibration of the model to environment specific empirical data in order to make the simulations more realistic to a particular group of trainees that use the simulation model. Therefore, the "Advanced" input window mainly serves as a service interface for tailoring the simulation model to a specific training scenario and context. As a consequence, the "Advanced" input window should be offered to non-

expert trainees only in order to present and discuss the calibration of the process parameters and their influence on project dynamics.

Output Windows

Three different output windows are provided. The first one ("General") presents the most important simulation results to the model user. Figure 62 shows a screenshot of the "General" output window. In the upper part of the window, on the left hand side, results summarising the overall project performance are provided, namely actual product size at project end, average manpower allocated, field defect density of the tested code, actual project duration ("Time") and total effort consumed. The three columns on the right hand side of the upper part of the window provide additional phase-specific data. The lower part of the window visualises the project behaviour by plotting the dynamic change of three key variables: number of requirements (functions) to be designed, number of tasks to be implemented, and number of tasks to be tested. The values of the current simulation run are compared to the simulation results with default settings. For example, in Figure 62 the simulation results when applying design and code inspections (run name: "current") can be compared to the default results (run name: "default"). The behavioural pattern of the current simulation shows interesting differences to the default setting. In order to achieve the same field defect density with the same average manpower allocation, the overall project duration and effort consumption is smaller in the current run, even though the duration and effort consumption for the design and coding phases are larger due to additional QA activities (inspections) and resulting rework.

The second output window ("Advanced") provides even more phase-specific simulation results, including detailed information about defect injection, defect detection, and effort consumption for QA activities and rework.

The third output window ("Internal") provides information about technical model variables such as COCOMO estimates and calculated average productivity per effort unit and per time unit.

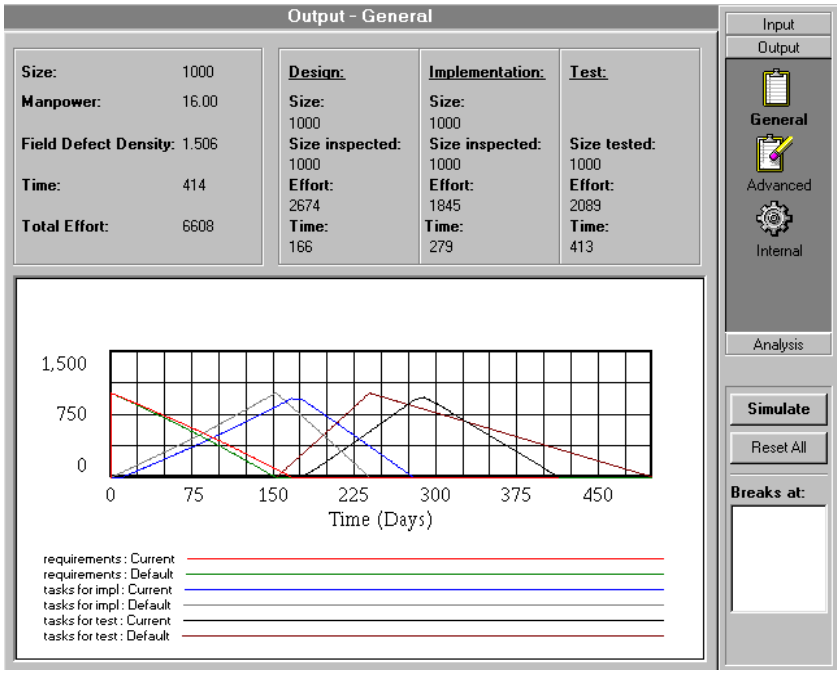


Figure 62: GENSIM output window "General"

Analysis Windows

The purpose of the analysis windows is to provide to the model user a set of tools that help interpret simulation results by connecting them with structural information about the simulation model. The tools offered through the analysis windows of GENSIM are a subset of the standard analysis tools provided by the SD tool Vensim. Four different analysis windows are provided. The analysis window "Graphs" offers possibilities to visualise the behaviour of any model variable individually, and together with the behaviour of its causal variables (strip graph functionality). The analysis window "Trees" provides means to create tree-type graphical representations showing either the causes or the uses of any selected model variable (cause tree or use tree). For example, Figure 63 shows the causal variables of the model variable "field defect density". The analysis window "Tables" offers a tabular presentation of the behaviour of any model variable. The analysis window "Models" provides means to represent the flow graphs of the model views (cf. Section 12.2.2).

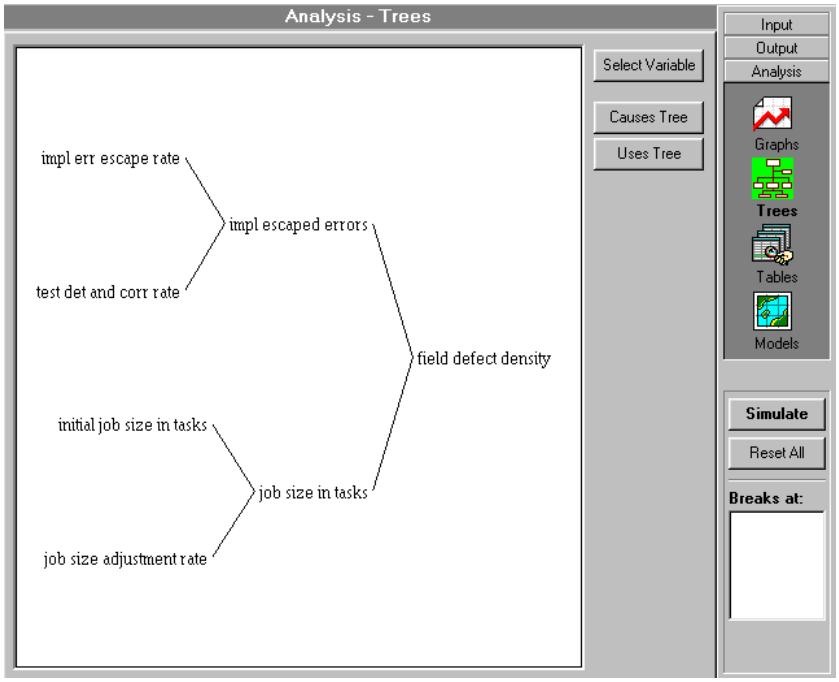


Figure 63: GENSIM analysis window "Trees"

Appendix D: Questionnaires used for Validating GENSIM

Influencing Factors - Background Characteristics / Before Pre-Test (5 min)

Name: _____ Subject ID:
<do not fill in this field>

Gender (optional): male: ___ female: ___

Age (optional): ___ years

DF 0.1: University Education

010 Major Subject (Hauptfach): _____

011 Minor Subject (Nebenfach): _____
(if more than one, please mention all)

012 Number of terms (Fachsemester) completed: _____

013 Is this your first course of studies (Erststudium): yes: ___ no: ___

014 Is this your second course of studies (Zweitstudium): yes: ___ no: ___

DF 0.2: Practical Software Engineering Experience

020 Have you ever written software? yes: ___ no: ___

021 Have you participated in the "SE Praktikum"? yes: ___ no: ___

022 Have you developed software in a large team (>4 persons) with distributed roles?
yes: ___ no: ___

023 Have you developed software in a project with long duration (>6 months)?
yes: ___ no: ___

024 Have you developed software in an industrial environment, i.e. in a project that developed software for a company product? yes: ___ no:___

DF 0.3: Software Project Management Literature <put exactly ONE answer per question>

031 How many books about software project management have you read?
___ 0
___ 1-2
___ 3-5
___ more than 5

032 What does the acronym COCOMO stand for (in software engineering)?
___ Constructive Cost Model

___ Communication Cost Model
___ Co-operative Concept Modelling
___ Constructive Collaboration Model
___ I don't know

033 What does Brook's Law state?

___ Adding more people to a late project always makes it more costly, but it does not always cause it to be completed later.
___ Hardly any projects succeed in less than ¾ of the calculated cost-optimum schedule, regardless of the number of people allocated.
___ Adding manpower to a late software project makes it later.
___ Adding manpower to a software project can cut down project duration by more than 50%.
___ I don't know

DF 0.4: Learning Style <more than one answer is possible>

041. What is your preferred learning style?

___ reading of text books (with exercises)
___ classroom lectures (with exercises)
___ group work (interaction with peers and teacher / including exercises)
___ web-based training modules (with computer interaction / including examples and exercises)

Pre-Test

Name: _____ Subject ID:

<do not fill in this field>

Questions on "Interest in software project management" (3 min)

Below you will find a number of opposing adjectives on both sides of each line. You can react to the statements by checking the appropriate point on the line, as in the next example

	1	2	3	4	5	
agree	X	O	O	O	O	disagree

when your opinion is that you fully agree.

Key:
1: fully agree
2: agree
3: undecided
4: disagree
5: fully disagree

071 I consider it very important for computer science students to know as much as possible about software project management

	1	2	3	4	5	
agree	O	O	O	O	O	disagree

072 I would like to get more information on software project management in my software engineering lectures at the university

	1	2	3	4	5	
agree	O	O	O	O	O	disagree

073 I would like to participate in a seminar (either at university or in the form of a training course) about software project management

	1	2	3	4	5	
agree	O	O	O	O	O	disagree

074 I consider it very important for software engineers to know as much as possible about software project management

	1	2	3	4	5	
agree	O	O	O	O	O	disagree

075 I would like to learn more about software project management (e.g. planning, control, improvement, human factors)

	1	2	3	4	5	
agree	O	O	O	O	O	disagree

Questions on "Knowledge about typical (empir.) patterns observed in SW projects" (5 min)

<For each question tick exactly one answer. / If in doubt, choose the answer that you think is most appropriate.>

081 For a typical software project, finding and fixing a software problem (defect) after delivery is about

___ 3 times
___ 5 times
___ 10 times
___ 100 times

more expensive than finding and fixing it during the requirements and early design phases

082 By adding manpower, the nominal schedule of a typical software development project can be compressed up to

___ 10 %(e.g. reduction from nominal 100 days to 90 days)
___ 25 %(e.g. reduction from nominal 100 days to 75 days)
___ 40 %(e.g. reduction from nominal 100 days to 60 days)
___ 60 %(e.g. reduction from nominal 100 days to 40 days)

but no more. (Note: the nominal schedule is the cost minimum schedule when using the standard process of the organisation.)

083 Software development cost is primarily a function of

___ product size
___ tool usage
___ product quality
___ workforce allocation

084 When comparing software development projects, variation between

___ programming language
___ tool support
___ programming style
___ people skills

accounts for the biggest difference in software productivity.

085 In typical software development projects, on average, software inspections detect about

___ 25 %
___ 40 %
___ 60 %
___ 90 %

of all defects contained in inspected documents (design or code).

Questions on "Knowledge about simple SW project dynamics" (10 min)

091 You have to estimate schedule and effort for a project of size 60,000 SLOC (source lines of code). Assume that you don't have any additional information about project specifics so that you can take the standard (or nominal) project performance as a baseline. Which cost-optimal schedule is most probable for the phases Design (high-level and detailed) – Coding (incl. unit test) – Test ?

___ 10 months
___ 14 months
___ 18 months
___ 22 months

092 For the same project as in 091, how much total effort is needed for phases Design – Coding – Test?

___ 150 person-months
___ 200 person-months
___ 250 person-months
___ 300 person-months

093 Using your estimates from 091 and 092, what is the average staff size?

___ persons per month

094 Assume that you can increase the average staff level of 093 without limitation, how much can you possibly shorten the schedule of your project (without changing product quality, development process, and skill level of developers)?

___ not at all
___ up to 25%
___ up to 50%
___ more than 50%

095 What would be your answer in 094 if – instead of increasing average staff level – you would only assign the most capable programmers to your project? The schedule of the project would be shortened:

___ not at all
___ up to 10%
___ up to 30%
___ more than 30%

096 In 095, how much would the effort be reduced?

___ not at all
___ up to10%
___ up to 30%
___ more than 30%

097 Assume that you are responsible for a 100,000 SLOC software project and you have estimated a cost-optimal schedule of about 20 calendar months, given that you can allocate only the best people. The overall budget amounts to a total of 10,000,000 DM. When you show these numbers to your customer he is concerned about the project duration and offers you additional 4,000,000 DM if you can finish the project within one year (12 calendar months). Should you accept the deal? (Note: if

you can't hold the agreed deadline your customer will be really upset and you are at risk that you won't get follow-up projects):

___ yes
___ no [please give a justification if you ticked "no":]

_____

_____

_____

Questions on "Knowledge about difficult project management issues" (12 min)

101 Assume you are responsible for a software project of size 60,000 SLOC. Assume that you don't have any additional information about project specifics so that you can take the standard (or nominal) project performance as a baseline. The standard process implies that 50% of the design and code documents are inspected. Using the standard process, Y person-months is the cost-optimal effort consumption for conducting the phases Design (high-level and detailed) – Coding (incl. unit test) – Test. Due to new customer requirements the reliability level of the software has to be "very high" (instead of "nominal"). Without changing the standard process, which development phases will be intensified most (by adding effort and extending the schedule) in order to achieve the increased reliability level?

___ design
___ implementation
___ test
___ all phases are intensified equally

102 Assume that in 101 it was not allowed to exceed the nominal effort consumption Y. What is the maximum reliability level that can be achieved if the number of design and code inspections is increased up to 100%?

___ "very high" reliability
___ "high" reliability
___ not significantly more than "nominal" reliability

103 Assume you are responsible for a software project of size 60,000 SLOC (as in 101). Assume that you don't have any additional information about project specifics so that you can take the standard (or nominal) project performance as a baseline. Using the standard process, X calendar months is the cost-optimal schedule for the project, consuming a total of Y person-months of effort. Assume the following situation: The project has already started, you have concluded the design phase(s) according to plan, and you are about to start with the implementation. In this situation, you receive additional requirements from the customer, which increase the overall product size to a total of 75,000 SLOC (= 60,000 SLOC + 15,000 SLOC new code). Assuming that you can add unlimited manpower to the project (without delay), can you keep the schedule of your project within the nominal schedule X+10%, if all other process parameters (e.g. QA activities, average manpower capability) are kept unchanged?

___ yes (go to 104a and skip 104b)
___ no(skip 104a and go directly to 104b)

104a If you answered "yes" in 103, how much total effort do you need?

___ Y + 15% additional effort
___ Y + 25% additional effort
___ Y + 40% additional effort
___ Y + 60% additional effort

104b If you answered "no" in 104, please explain why you think that you cannot keep the project deadline within X+10%:

_____

_____

_____

105 What would be your answer in 103, if you had to consider the following constraints:

a) You can add additional manpower only with a delay of 1.5 month (30 work days)
b) The overall effort consumption must not be larger than Y+40%?

___ yes
___ no

106 Before the start of a typical software project, for a fixed set of customer requirements, a cost-minimalestimate of the project duration (in calendar time) and the average needed staff size (in number of persons) has been made. Assuming that the project manager wants to reduce project duration by changing the staff size (ignoring any potential financial constraints), which of the patterns presented below describes the typical effect of staff size variation on project duration most appropriate?

Figure 1: ___ Figure 2: ___ Figure 3: ___ Figure 4: ___

Figure 1

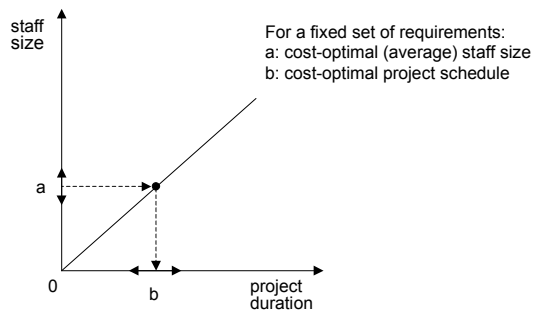


Figure 2

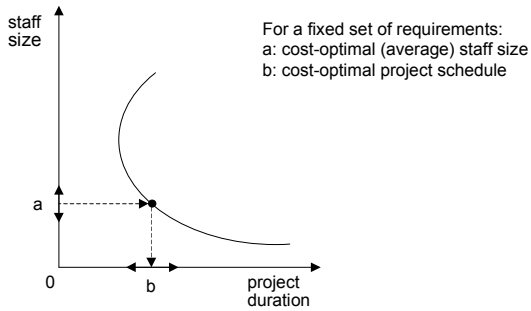


Figure 3

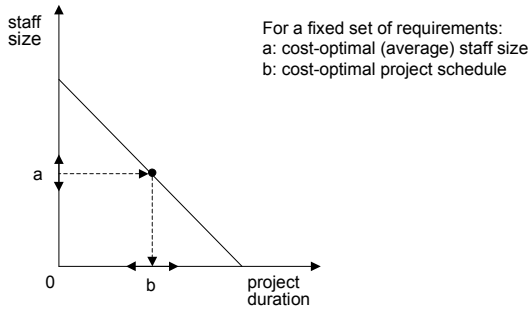
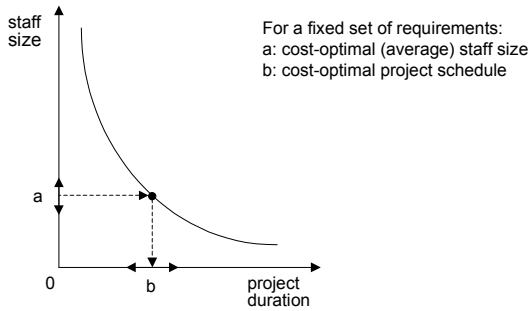


Figure 4



Post-Test / Group A

Name: _____ Subject ID:

<do not fill in this field>

Questions on "Interest in software project management" (3 min)

Below you will find a number of opposing adjectives on both sides of each line. You can react to the statements by checking the appropriate point on the line, as in the next example

	1	2	3	4	5	
agree	X	O	O	O	O	disagree

when your opinion is that you fully agree.

Key:
1: fully agree
2: agree
3: undecided
4: disagree
5: fully disagree

071 I consider it very important for computer science students to know as much as possible about software project management

	1	2	3	4	5	
agree	O	O	O	O	O	disagree

072 I would like to get more information on software project management in my software engineering lectures at the university

	1	2	3	4	5	
agree	O	O	O	O	O	disagree

073 I would like to participate in a seminar (either at university or in the form of a training course) about software project management

	1	2	3	4	5	
agree	O	O	O	O	O	disagree

074 I consider it very important for software engineers to know as much as possible about software project management

	1	2	3	4	5	
agree	O	O	O	O	O	disagree

075 I would like to learn more about software project management (e.g. planning, control, improvement, human factors)

	1	2	3	4	5	
agree	O	O	O	O	O	disagree

Questions on "Knowledge about typical (empir.) patterns observed in SW projects" (5 min)

<For each question tick exactly one answer. / If in doubt, choose the answer that you think is most appropriate.>

081 For a typical software project, finding and fixing a software problem (defect) after delivery is about

___ 3 times
___ 5 times
___ 10 times
___ 100 times

more expensive than finding and fixing it during the requirements and early design phases

082 By adding manpower, the nominal schedule of a typical software development project can be compressed up to

___ 10 %(e.g. reduction from nominal 100 days to 90 days)
___ 25 %(e.g. reduction from nominal 100 days to 75 days)
___ 40 %(e.g. reduction from nominal 100 days to 60 days)
___ 60 %(e.g. reduction from nominal 100 days to 40 days)

but no more. (Note: the nominal schedule is the cost minimum schedule when using the standard process of the organisation.)

083 Software development cost is primarily a function of

___ product size
___ tool usage
___ product quality
___ workforce allocation

084 When comparing software development projects, variation between

___ programming language
___ tool support
___ programming style
___ people skills

accounts for the biggest difference in software productivity.

085 In typical software development projects, on average, software inspections detect about

___ 25 %
___ 40 %
___ 60 %
___ 90 %

of all defects contained in inspected documents (design or code).

Questions on "Knowledge about simple SW project dynamics" (10 min)

091 You have to estimate schedule and effort for a project of size 1,000 tasks (functions) with a goal defect density of 1.7 defects/function. Assume that you don't have any additional information about project specifics so that you can take the current standard project performance as a baseline. Which cost-optimal schedule is most probable for the phases Design – Implementation – Test?

___ 10 months
___ 14 months
___ 18 months
___ 22 months

0101 For the same project as in 091, how much total effort is needed for phases Design – Implementation – Test?

___ 150 person-months
___ 200 person-months
___ 250 person-months
___ 300 person-months

0102 Using your estimates from 091 and 092, what is the average staff size?

___ persons per month

0103 Assume that you can increase the average staff level of 093 without limitation, how much can you possibly shorten the schedule of your project (without changing product quality, development process, and skill level of developers)?

___ not at all
___ up to 25%
___ up to 50%
___ more than 50%

0104 What would be your answer in 094 if – instead of increasing average staff level – you would only assign the most capable programmers to your project? The schedule of the project would be shortened:

___ not at all
___ up to 10%
___ up to 30%
___ more than 30%

0105 In 095, how much would the effort be reduced?

___ not at all
___ up to10%
___ up to 30%
___ more than 30%

0106 Assume that you are responsible for a software project of size 1,700 tasks with a goal defect density of 1.2 defects/task, and you have estimated a cost-optimal schedule of about 20 calendar months, given that you can allocate only the best people. The overall budget amounts to a total of 10,000,000 DM. When you show these numbers to your customer he is concerned about the project duration and

offers you additional 4,000,000 DM if you can finish the project within one year (12 calendar months). Should you accept the deal? (Note: if you can't hold the agreed deadline your customer will be really upset and you are at risk that you won't get follow-up projects):

___ yes
___ no [please give a justification if you ticked "no":]

_____

_____

_____

Questions on "Knowledge about difficult project management issues" (12 min)

101 Assume you are responsible for a software project of size 1,000 tasks. Assume that you don't have any additional information about project specifics so that you can take the standard (or nominal) project performance as a baseline. The standard process implies that 50% of the design and code documents are inspected. Using the standard process, Y person-months is the cost-optimal effort consumption for conducting the phases Design – Coding (incl. unit test) – Test. Due to new customer requirements the quality level of the software has to be "very high", i.e. 0.2 defects per task (instead of "nominal", i.e. 1.7 defects per task). Without changing the standard process, which development phase(s) will be intensified most (by adding effort and extending the schedule) in order to achieve the increased reliability level?

___ design
___ implementation (coding)
___ test
___ all phases are intensified equally

102 Assume that in 101 it was not allowed to exceed the nominal effort consumption Y. What is the maximum quality level that can be achieved if the number of design and code inspections is increased up to 100%?

___ "very high" quality (better than 0.2 defects per task)
___ "high" quality (better than 0.8 defects per task)
___ not significantly more than "nominal" quality (1.7 defects per task)

103 Assume you are responsible for a software project of size 1,000 tasks (as in 101). Assume that you don't have any additional information about project specifics so that you can take the standard (or nominal) project performance as a baseline. Using the standard process, X calendar months is the cost-optimal schedule for the project, consuming a total of Y person-months of effort. Assume the following situation: The project has already started, you have concluded the design phase(s) according to plan, and you are about to start with the implementation. In this situation, you receive additional requirements from the customer, which increase the overall product size to a total of 1,250 tasks (= 1,000 tasks + 250 new tasks). Assuming that you can add unlimited manpower to the project (without delay), can you keep the schedule of your project within the nominal schedule X+10%, if all other process parameters (e.g. QA activities, average manpower skill) are kept unchanged?

___ yes(go to 104a and skip 104b)
___ no(skip 104a and go directly to 104b)

104a If you answered "yes" in 103, how much total effort do you need?

___ Y + 15% additional effort
___ Y + 25% additional effort
___ Y + 40% additional effort
___ Y + 60% additional effort

104b If you answered "no" in 104, please explain why you think that you cannot keep the project deadline within X+10%:

_____

_____

_____

105 What would be your answer in 103, if you had to consider the following constraints:

a) You can add additional manpower only with a delay of 1.5 month (30 work days)
b) The overall effort consumption must not be larger than Y+40%?

___ yes
___ no

106 Before the start of a typical software project, for a fixed set of customer requirements, a cost-minimal estimate of the project duration (in calendar time) and the average needed staff size (in number of persons) has been made. Assuming that the project manager wants to reduce project duration by changing the staff size (ignoring any potential financial constraints), which of the patterns presented below describes the typical effect of staff size variation on project duration most appropriate?

Figure 1: ___ Figure 2: ___ Figure 3: ___ Figure 4: ___

Figure 1

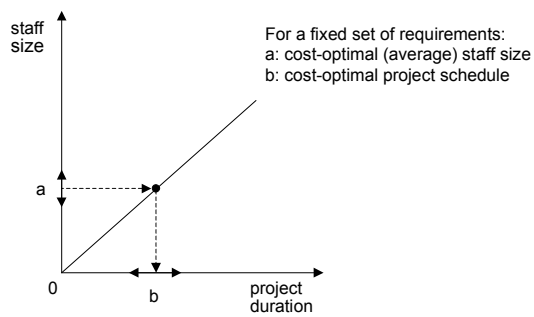


Figure 2

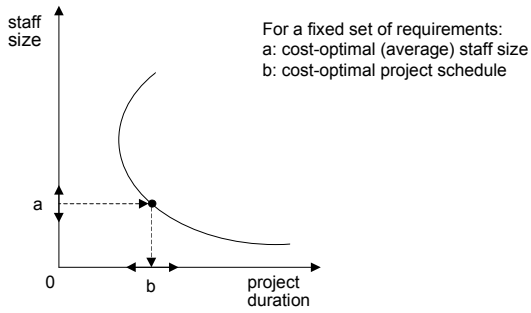


Figure 3

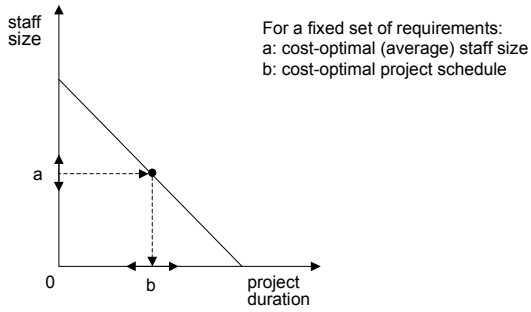
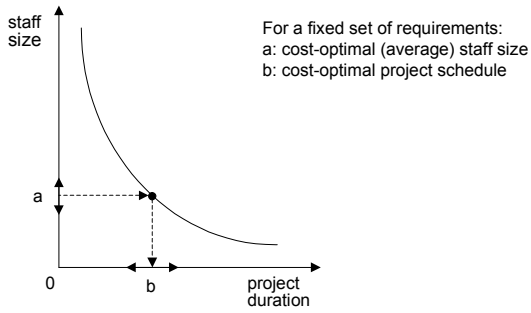


Figure 4



Post-Test / Group B

Name: _____ Subject ID:

<do not fill in this field>

Questions on "Interest in software project management" (3 min)

Below you will find a number of opposing adjectives on both sides of each line. You can react to the statements by checking the appropriate point on the line, as in the next example

	1	2	3	4	5	
agree	X	O	O	O	O	disagree

when your opinion is that you fully agree.

Key:
1: fully agree
2: agree
3: undecided
4: disagree
5: fully disagree

071 I consider it very important for computer science students to know as much as possible about software project management

	1	2	3	4	5	
agree	O	O	O	O	O	disagree

072 I would like to get more information on software project management in my software engineering lectures at the university

	1	2	3	4	5	
agree	O	O	O	O	O	disagree

073 I would like to participate in a seminar (either at university or in the form of a training course) about software project management

	1	2	3	4	5	
agree	O	O	O	O	O	disagree

074 I consider it very important for software engineers to know as much as possible about software project management

	1	2	3	4	5	
agree	O	O	O	O	O	disagree

075 I would like to learn more about software project management (e.g. planning, control, improvement, human factors)

	1	2	3	4	5	
agree	O	O	O	O	O	disagree

Questions on "Knowledge about typical (empir.) patterns observed in SW projects" (5 min)

<For each question tick exactly one answer. / If in doubt, choose the answer that you think is most appropriate.>

081 For a typical software project, finding and fixing a software problem (defect) after delivery is about

___ 3 times
___ 5 times
___ 10 times
___ 100 times

more expensive than finding and fixing it during the requirements and early design phases

082 By adding manpower, the nominal schedule of a typical software development project can be compressed up to

___ 10 %(e.g. reduction from nominal 100 days to 90 days)
___ 25 %(e.g. reduction from nominal 100 days to 75 days)
___ 40 %(e.g. reduction from nominal 100 days to 60 days)
___ 60 %(e.g. reduction from nominal 100 days to 40 days)

but no more. (Note: the nominal schedule is the cost minimum schedule when using the standard process of the organisation.)

083 Software development cost is primarily a function of

___ product size
___ tool usage
___ product quality
___ workforce allocation

084 When comparing software development projects, variation between

___ programming language
___ tool support
___ programming style
___ people skills

accounts for the biggest difference in software productivity.

085 In typical software development projects, on average, software inspections detect about

___ 25 %
___ 40 %
___ 60 %
___ 90 %

of all defects contained in inspected documents (design or code).

Questions on "Knowledge about simple SW project dynamics" (10 min)

091 You have to estimate schedule and effort for a semi-detached project of size 60,000 DSI (delivered source instructions). Assume that you don't have any additional information about project specifics so that you can take the standard (or nominal) project performance as a baseline. Which cost-optimal schedule is most probable for the phases PD – DD – CT – IT?

___ 10 months
___ 14 months
___ 18 months
___ 22 months

092 For the same project as in 091, how much total effort is needed for phases PD – DD – CT – IT?

___ 150 person-months
___ 200 person-months
___ 250 person-months
___ 300 person-months

093 Using your estimates from 091 and 092, what is the average staff size?

___ persons per month

094 Assume that you can increase the average staff level of 093 without limitation, how much can you possibly shorten the schedule of your project (without changing product quality, development process, and skill level of developers)?

___ not at all
___ up to 25%
___ up to 50%
___ more than 50%

095 What would be your answer in 094 if – instead of increasing average staff level – you would only assign the most capable programmers to your project? The schedule of the project would be shortened:

___ not at all
___ up to 10%
___ up to 30%
___ more than 30%

096 In 095, how much would the effort be reduced?

___ not at all
___ up to10%
___ up to 30%
___ more than 30%

097 Assume that you are responsible for a 100000 DSI semi-detached software project and you have estimated a cost-optimal schedule of about 20 calendar months, given that you can allocate only the most capable programmers. The overall budget amounts to a total of 10,000,000 DM. When you show these numbers to your customer he is concerned about the project duration and offers you additional

4,000,000 DM if you can finish the project within one year (12 calendar months). Should you accept the deal? (Note: if you can't hold the agreed deadline your customer will be really upset and you are at risk that you won't get follow-up projects):

___ yes
___ no [please give a justification if you ticked "no":]

_____

_____

_____

Questions on "Knowledge about difficult project management issues" (12 min)

101 Assume you are responsible for a software project of size 60,000 DSI. Assume that you don't have any additional information about project specifics so that you can take the standard (or nominal) project performance as a baseline. The standard process implies that 50% of the design and code documents are inspected. Using the standard process, Y person-months is the cost-optimal effort consumption for conducting the phases PD – DD – CT – IT. Due to new customer requirements the reliability level [cost driver: RELY] of the software has to be "very high" (instead of "nominal"). Without changing the standard process, which development phase(s) will be intensified most (by adding effort and extending the schedule) in order to achieve the increased reliability level?

___ design (PD and DD)
___ implementation (CT)
___ test (IT)
___ all phases are intensified equally

102 Assume that in 101 it was not allowed to exceed the nominal effort consumption Y. What is the maximum reliability level that can be achieved if the number of design and code inspections is increased up to 100% [cost driver MODP is set to "very high"]? ___ "very high" reliability

___ "high" reliability
___ not significantly more than "nominal" reliability

103 Assume you are responsible for a software project of size 60,000 DSI (as in 101). Assume that you don't have any additional information about project specifics so that you can take the standard (or nominal) project performance as a baseline. Using the standard process, X calendar months is the cost-optimal schedule for the project, consuming a total of Y person-months of effort. Assume the following situation: The project has already started, you have concluded the design phase(s) according to plan, and you are about to start with the implementation. In this situation, you receive additional requirements from the customer, which increase the overall product size to a total of 75,000 DSI (= 60,000 DSI + 15,000 DSI new code). Assuming that you can add unlimited manpower to the project (without delay), can you keep the schedule of your project within the nominal schedule X+10%, if all other process parameters (e.g. QA activities, average manpower capability) are kept unchanged?

___ yes(go to 104a and skip 104b)
___ no(skip 104a and go directly to 104b)

104a If you answered "yes" in 103, how much total effort do you need?

___ Y + 15% additional effort
___ Y + 25% additional effort
___ Y + 40% additional effort
___ Y + 60% additional effort

104b If you answered "no" in 104, please explain why you think that you cannot keep the project deadline within X+10%:

_____

_____

_____

105 What would be your answer in 103, if you had to consider the following constraints:

a) You can add additional manpower only with a delay of 1.5 month (30 work days)
b) The overall effort consumption must not be larger than Y+40%?

___ yes
___ no

0106 Before the start of a typical software project, for a fixed set of customer requirements, a cost-minimal estimate of the project duration (in calendar time) and the average needed staff size (in number of persons) has been made. Assuming that the project manager wants to reduce project duration by changing the staff size (ignoring any potential financial constraints), which of the patterns presented below describes the typical effect of staff size variation on project duration most appropriate?

Figure 1: ___ Figure 2: ___ Figure 3: ___ Figure 4: ___

Figure 1

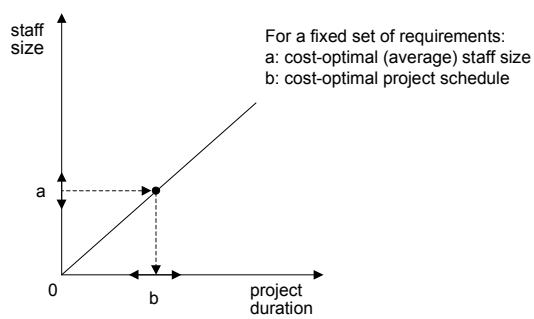


Figure 2

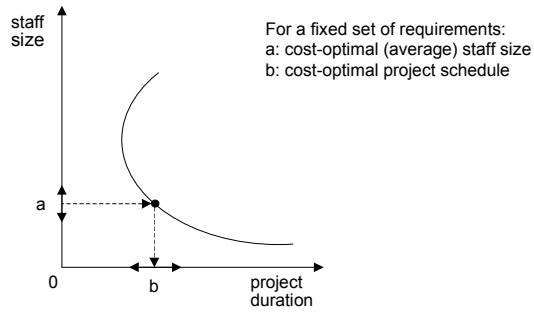


Figure 3

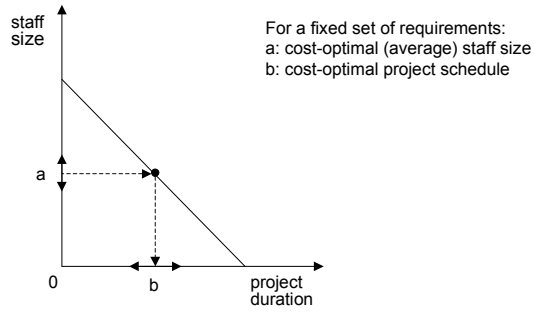
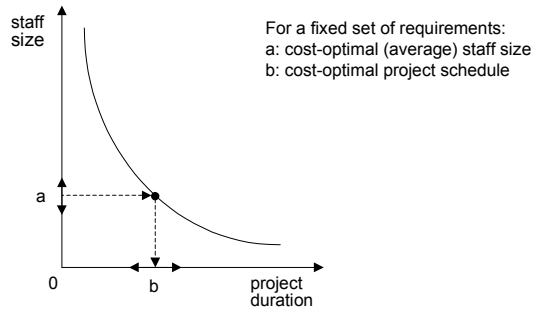


Figure 4



Influencing Factors – After Post-Test / Group A

Name: _____ Subject ID:
<do not fill in this field>

DF 0.5: Time Need <more than one answer is possible> (1 min)

051 I did not have enough time to

___ read the materials on project management (during training session), particularly:
- ___ Block 1
- ___ Block 2
- ___ Block 3
- ___ Block 4

___ familiarize with the tool(s) (during training session)

___ complete the post-test

052 I spent ____ minutes on Block 2 (Role Play)

053 I completed at least one full scenario of the Role Play in Block2

___ yes

___ no

054 I would have liked to spend more time for Block3 and Block 4 instead of conducting Block2 (Role Play)

___ yes

___ no

DF 0.6: Session Evaluation (4 min)

Below you will find a number of opposing adjectives on both sides of each line. You can react to the statements by checking the appropriate point on the line, as in the next example

	1	2	3	4	5	
useful	X	O	O	O	O	useless

when your opinion is that it was very useful.

Key:
1: very useful
2: useful
3: undecided
4: useless
5: very useless

061a I consider the explanations / information provided in Block2 (Role Play) in general

	1	2	3	4	5	
useful	O	O	O	O	O	useless
	1	2	3	4	5	
boring	O	O	O	O	O	absorbing
	1	2	3	4	5	
difficult	O	O	O	O	O	easy
	1	2	3	4	5	
clear	O	O	O	O	O	confusing

061b I consider the explanations / information provided in Block1, Block3, and Block4 in general

	1	2	3	4	5	
useful	O	O	O	O	O	useless
	1	2	3	4	5	
boring	O	O	O	O	O	absorbing
	1	2	3	4	5	
difficult	O	O	O	O	O	easy
	1	2	3	4	5	
clear	O	O	O	O	O	confusing

062 II would like to make the following comment(s) / improvement suggestion(s) (can be in German):

_____

_____

_____

063 I had a problem with … <please explain (can be in German)>:

_____

_____

_____

Influencing Factors – After Post-Test / Group B

Name: _____ Subject ID:
<do not fill in this field>

DF 0.5: Time Need <more than one answer is possible> (1 min)

051 I did not have enough time to

___ read the materials on project management (during training session), particularly:
- ___ Block 1
- n/a Block 2 (not available for Group B)
- ___ Block 3
- ___ Block 4

___ familiarize with the tool(s) (during training session)

___ complete the post-test

DF 0.6: Session Evaluation (4 min)

Below you will find a number of opposing adjectives on both sides of each line. You can react to the statements by checking the appropriate point on the line, as in the next example

	1	2	3	4	5	
useful	X	O	O	O	O	useless

when your opinion is that it was very useful.

Key:
1: very useful
2: useful
3: undecided
4: useless
5: very useless

061 I consider the explanations / information provided by the training materials in general

	1	2	3	4	5	
useful	O	O	O	O	O	useless
	1	2	3	4	5	
boring	O	O	O	O	O	absorbing
	1	2	3	4	5	
difficult	O	O	O	O	O	easy
	1	2	3	4	5	
clear	O	O	O	O	O	confusing

062 II would like to make the following comment(s) / improvement suggestion(s) (can be in German):

_____

_____

_____

063 I had a problem with … <please explain (can be in German)>:

_____

_____

_____

Appendix E: Product-Flow Representation of IMMoS Process Model

Figure 64 shows a product-flow representation of the IMMoS Process Model. This representation was created with the SPEARMINT tool. Based on this representation, a web-based Electronic Process Guide of the IMMoS Process Model can be generated automatically.

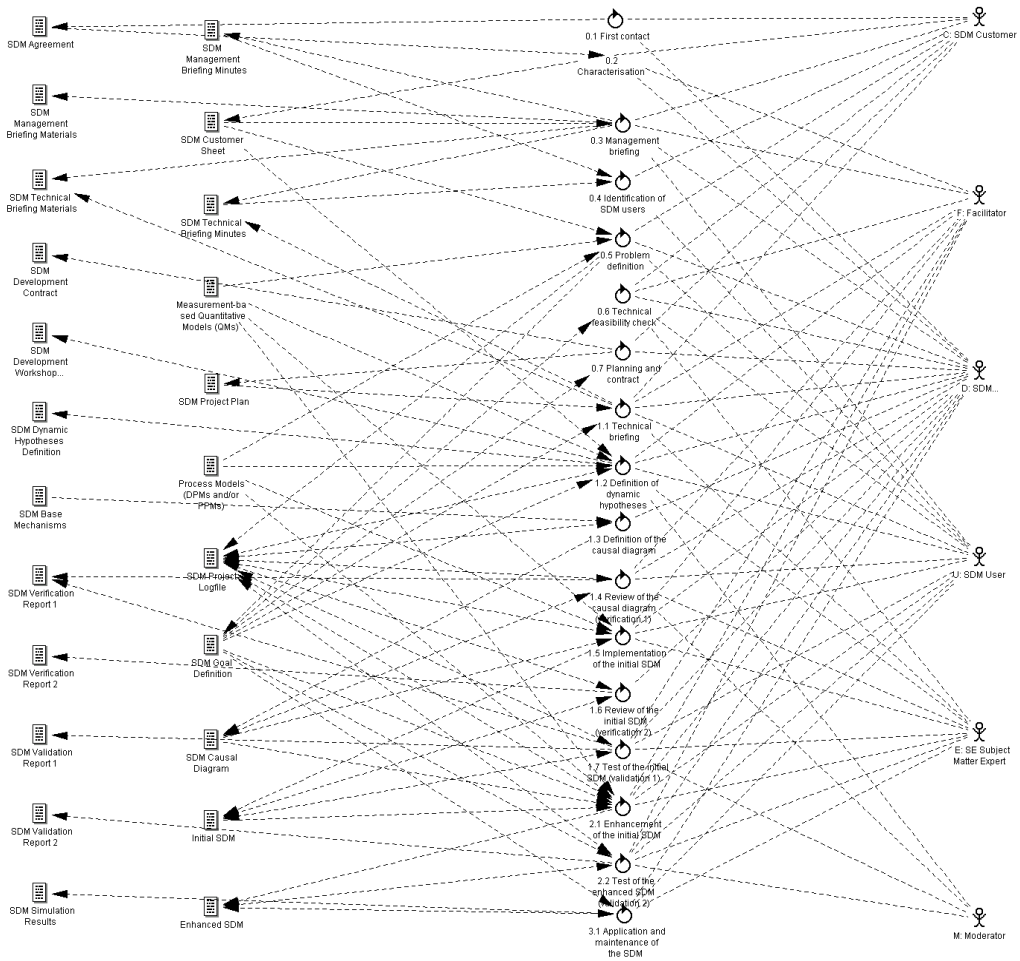


Figure 64: Product-flow representation of the IMMoS Process Model

Lebenslauf

Name	Dietmar Pfahl
geboren	am 9. Januar 1963 in München
Familienstand	verheiratet
Anschrift	Rubensstraße 23 D-60596 Frankfurt/Main
1969-1972	Grundschule
1972-1981	Max-Born-Gymnasium Germering
1981-1986	Universität Ulm Studium der Wirtschaftsmathematik
1984	Auslandssemester an der University of Southern California, Los Angeles Teaching Assistant
1986	Studienabschluß zum Diplom-Wirtschaftsmathematiker
1987-1990	Siemens AG, München Mitarbeiter im Zentralbereich Produktion und Logistik Abteilung "Qualitätsaufgaben Software"
1990-1991	Universität des Baskenlandes in Donostia (San Sebastian) Fachbereich: Grundlagen der Informatik und Kognitionswissenschaften Studienfächer: Logik, Kognitive Psychologie, Sprachverarbeitung (Auslandsstipendium des Freistaates Bayern)
1992	Deutschen Forschungsanstalt für Luft- und Raumfahrt (DLR), Oberpfaffenhofen Wiss. Mitarbeiter im Deutschen Fernerkundungsdatenzentrum (DFD), Arbeitsgruppe "Software Engineering"
1993-1996	Siemens AG, München Mitarbeiter im Zentralbereich Forschung und Entwicklung Abteilung "Software und Engineering"
Seit 1996	Fraunhofer-Institut für Experimentelles Software-Engineering (IESE), Kaiserslautern Projektleiter, Abteilungsleiter (seit 7/2001)

PhD Theses in Experimental Software Engineering

Volume 1 **Oliver Laitenberger** (2000), *Cost-Effective Detection of Software Defects Through Perspective-based Inspections*

Volume 2 **Christian Bunse** (2000), *Pattern-Based Refinement and Translation of Object-Oriented Models to Code*

Volume 3 **Andreas Birk** (2000), *A Knowledge Management Infrastructure for Systematic Improvement in Software Engineering*

Volume 4 **Carsten Tautz** (2000), *Customizing Software Engineering Experience Management Systems to Organizational Needs*

Volume 5 **Erik Kamsties** (2001), *Surfacing Ambiguity in Natural Language Requirements*

Volume 6 **Christiane Differding** (2001), *Adaptive Measurement Plans for Software Development*

Volume 7 **Isabella Wieczorek** (2001), *Improved Software Cost Estimation A Robust and Interpretable Modeling Method and a Comprehensive Empirical Investigation*

Volume 8 **Dietmar Pfahl** (2001), *An Integrated Approach to Simulation-Based Learning in Support of Strategic and Project Management in Software Organisations*

Software Engineering has become one of the major foci of Computer Science research in Kaiserslautern, Germany. Both the University of Kaiserslautern's Computer Science Department and the Fraunhofer Institute for Experimental Software Engineering (IESE) conduct research that subscribes to the development of complex software applications based on engineering principles. This requires system and process models for managing complexity, methods and techniques for ensuring product and process quality, and scalable formal methods for modeling and simulating system behavior. To understand the potential and limitations of these technologies, experiments need to be conducted for quantitative and qualitative evaluation and improvement. This line of software engineering research, which is based on the experimental scientific paradigm, is referred to as 'Experimental Software Engineering'.
In this series, we publish all PhD theses from the Fraunhofer Institute for Experimental Software Engineering (IESE) and from the Software Engineering Research Groups (e.g., AGSE and AGCE) of the Computer Science Department at the University of Kaiserslautern. PhD theses that originate elsewhere can be included, if accepted by the Editorial Board.

Prof. Dr. Dieter Rombach
Director of Fraunhofer IESE and Head of the AGSE Group of the Computer Science Department, University of Kaiserslautern

IESE
Fraunhofer Institut Experimentelles Software Engineering

UNIVERSITÄT KAISERSLAUTERN
AG Software Engineering
AG Component Engineering

PhD Theses in Experimental Software Engineering