Fraunhofer Einrichtung Experimentelles Software Engineering

Integration of System Dynamics Modelling with Descriptive Process Modelling and Goal-Oriented Measurement

Authors:
Dietmar Pfahl
Karl Lebsanft[1]
Ferdinand Vollei[1]

[1] Siemens AG, ZT SE 3, Munich, Germany

A version of this report was accepted for publication in the proceedings of the Software Process Simulation Modeling Workshop (ProSim'98), Silver Falls, Oregon, USA, June 22-24, 1998.

IESE-Report No. 032.98/E
Version 1.0
May 28, 1998

A publication by Fraunhofer IESE

Fraunhofer IESE is an institute of the Fraunhofer Gesellschaft.
The institute transfers innovative software development techniques, methods and tools into industrial practice, assists companies in building software competencies customized to their needs, and helps them to establish a competetive market position.

Fraunhofer IESE is directed by
Prof. Dr. Dieter Rombach
Sauerwiesen 6
D-67661 Kaiserslautern

Abstract

One of the obstacles that seem to impede a more frequent application of the modelling and simulation approach System Dynamics (SD) in the software engineering community is the fact that there is a lack of a well-defined and repeatable procedure for generating or using information that a) stems from real industrial software development practice and b) is suitable for SD modelling. This problem can be resolved, at least partially, by combining SD modelling with already existing and commonly used modelling methods like Process Modelling (PM) and measurement-based Quantitative Modelling (QM). In this paper, an approach is presented that complements SD modelling with software-related PM and QM. The new approach is called IMMoS (Integrated Measurement, Modelling and Simulation). It originates from lessons learnt that have been derived from an industrial SD modelling activity.

Table of Contents

1 Introduction

During the 1950s, the modelling method System Dynamics (SD) was developed at MIT to tackle socio-economic and/or socio-technical problems [For61]. The method is based on the assumption of the ubiquity of feedback processes in human interactions: considered from a high level of abstraction, a socio-economic or socio-technical system can be modelled as a feedback structure, whose complex behaviour is generated by the interaction of many (possibly non-linear) loops over time. Simulation with SD 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. The method has been applied to a wide range of domains, from the management of production-distribution systems to the management of ecosystems. During the last ten years the management of software projects and analysis of software processes has emerged as a new application domain (e.g. [Lin89], [LeL91], [AbM91], [Lin93], [Mad94], [PfK95], [Tve96], [LAS97]. An overview of potential application domains related to software development is presented in [WaP94]. Typical examples of software-related applications are training of project managers, comparison of process alternatives, and optimisation of certain process parameters for the purpose of project planning (including trade-off analysis, project control, risk analysis, re-planning, etc.)

In this paper, a new approach is introduced that combines SD modelling with Process Modelling (PM) and measurement-based Quantitative Modelling (QM) in the application domain of software development. The purpose of the new approach is to generate synergy effects from using already existing modelling methods and, by doing so, to overcome some of the weaknesses of SD model building. The new approach is called IMMoS (Integrated Measurement, Modelling and Simulation).

The structure of the paper is as follows. In the next section, the results of an SD project conducted in the software development department of a large telecommunication company is sketched and the lessons learnt are listed. Some of the lessons learnt serve as a general motivation for defining the IMMoS approach which is outlined in the third section. In Section 4, the relevant products and concepts which are provided or supported by the individual IMMoS components are presented. The relationships between products and concepts of the individual IMMoS components are described in detail in Section 5. Section 6 summarises the main results presented in this paper and draws conclusions for future work.

2 The PSIM project

The motivation for the development of IMMoS originated from experiences with a SD modelling project which was conducted inside a large software development department of the Siemens Private Networks Group (Siemens PN) in the years 1994 and 1995. As a result of this pilot project the SD model PSIM (Project SIMulator) emerged [Pfa94]. The main objective of the PSIM project was to explore the potentials of the SD approach in a real setting, and although the PSIM model was only a prototype which has not been used in productive work on a regular base, it was a very useful experience which generated many new insights, especially about methodological and organisational aspects of SD modelling in software industry.

In the following sub-sections the PSIM project goals of the intended model users, the PSIM model structure, example PSIM applications, and lessons learnt from the PSIM project are summarised. A more detailed description of the PSIM project and potential applications can be found in [PfK95][Pfa95].

2.1 Project goals

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. In the long run, it was hoped to be able to use PSIM as a support tool for continuous process improvement.

Related to project planning and control the following aspects were in the focus of interest:

- Defect generation, detection, and propagation along development phases,
- Trade-off between project duration, product quality, and manpower (effort),
- Effects of unexpected change requests,
- Interdependence between software features,
- Interdependence between concurrent projects.

2.2 PSIM model structure

The first version of the SD model PSIM covered the software development phases high-level design (HLD), low-level design (LLD), implementation (IMP) and developer test (TEST). In a later version, the test phase was divided into two sub-phases, namely component test (CoT) which is conducted by the development team, and system test (SyT) which is conducted by a separate test team. To increase the readability of the overall model, each development phase was implemented as a separate module (or view). Of course, all modules are mutually interrelated, reflecting critical aspects of the interdependency between development phases.

As an important input to the development of the SD model, a wide range of structural and quantitative information about the actual development process was used. The process information gathered from managers and experienced developers formed the basis of the causal structure of the model. Since PSIM is supposed to serve mainly for project planning and control, additional project data must be fed into the model before a simulation can be started. The project performance is then reflected by the simulation output data. A rough idea of the model structure for phase HLD provides Figure 1.

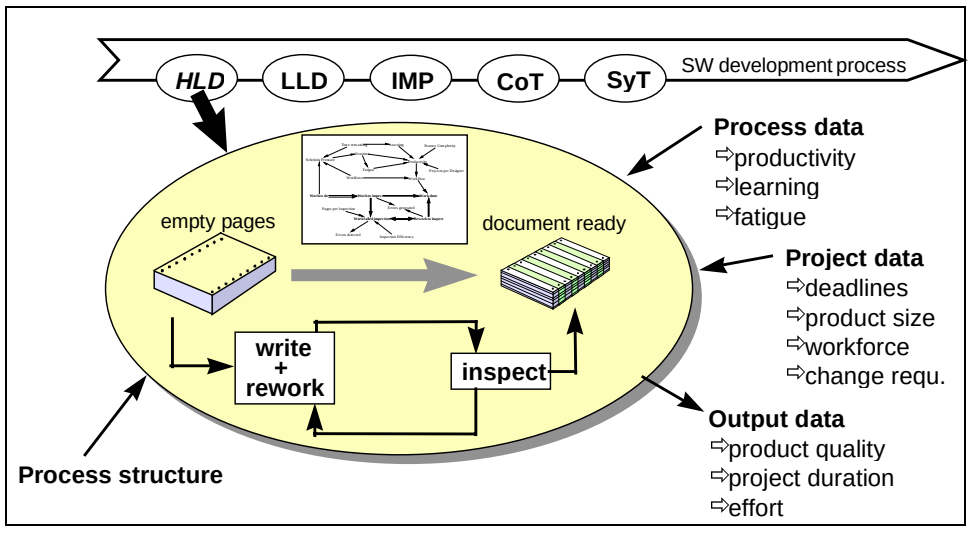

Figure 1: Overall model structure of phase HLD.

In the PSIM model, the generic process behaviour is generated endogenously by the cause-effect structure of the model, i.e. without any additional information which is not already included in the model structure. The project mapping

is done through additional model variables representing the project specific information. These project variables are exogenous, i.e. although they have an impact on the project-specific process dynamics, they are immune against feedback effects. During a simulation, and within certain limits, the project variables can be manipulated by project management to influence the project performance. Since the project management function is not part of the causal structure of the SD model, all project management decisions must as well be interpreted as exogenous influences on the generic process behaviour.

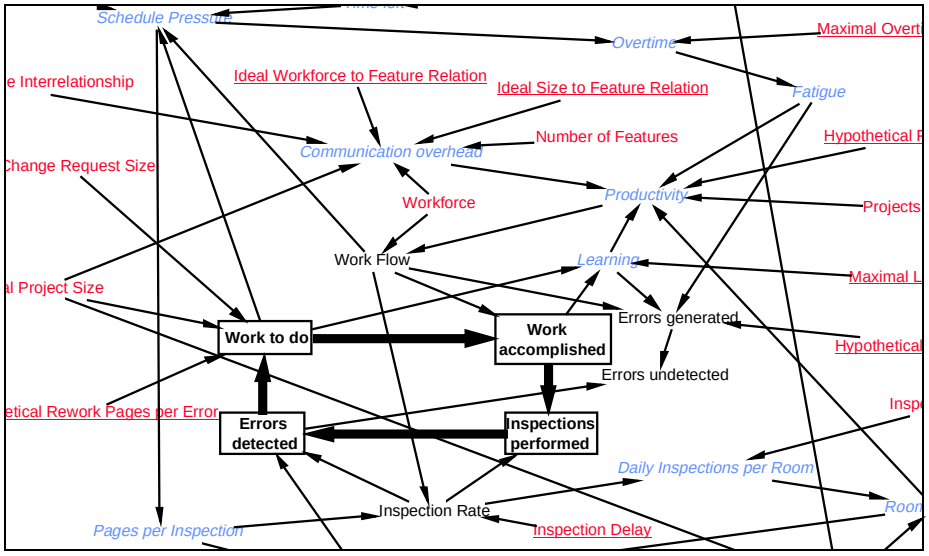

Figure 2: Causal diagram of phase HLD.

For each of the software development phases, a causal diagram containing the qualitative cause-effect relationships, and a more formal flow graph containing the mathematical equations of the simulation model, were set up. Figure 2 sketches the causal diagram of phase HLD, showing a) the core process with the major state variables (surrounded by boxes), and b) the network of influencing factors.

a) Core process: The set of customer requirements defines the amount of work to do (state variable: *Work to do*). The design activity produces a set of design documents (state variable: *Work accomplished*). Before the design documents can be released to the subsequent phase (LLD) they are subject to inspections (state variable: *Inspections performed*). If defects are detected during an inspection (state variable*: Errors detected*) the design

document has to be - at least partially - reworked and goes back to the (virtual) amount of work to do (state variable: *Work to do*).

b) The large set of factors influencing the core process can be subdivided into two groups: exogenous and endogenous factors. Exogenous factors that can be controlled by project management include, among others, *Workforce*, *Projects per Designer*, *Time left*, etc. Endogenous factors are an integral part of the modelled software development system (determined by the number, experience, and skills of the development team members, as well as tools, methods, guidelines, physical equipment, etc.). They cannot directly be manipulated by project management. Examples of endogenous factors include *Learning*, *Time pressure*, *Fatigue*, *Errors generated*, etc.

2.3 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).

2.3.1 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 3). 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 to switch between different phases.

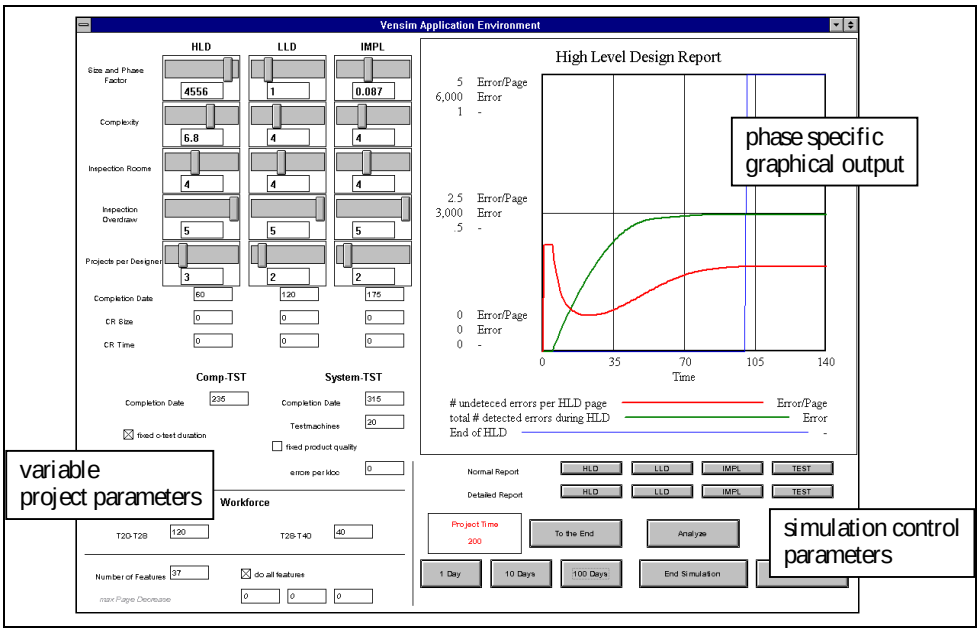

Figure 3: The PSIM simulation cockpit.

2.3.2 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 4 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.

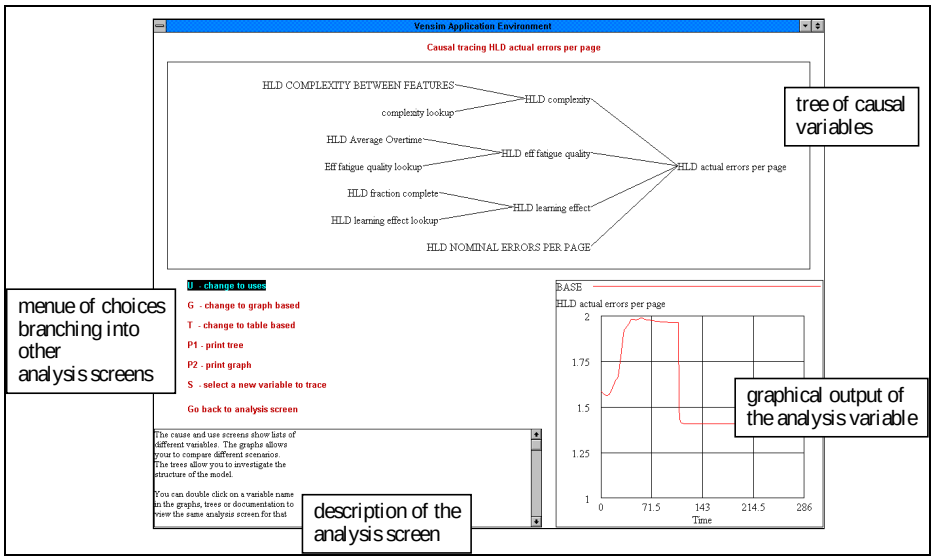

Figure 4: An example PSIM analysis screen.

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).
- Representation 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.

2.4 PSIM applications

The potential applications of PSIM were already mentioned in Sub-Section 2.1, 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.

2.4.1 Project planning

Under the assumption that the SD model 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., manpower, estimated product size, number of features, estimated feature complexity, etc.) she/he predicts the duration of the individual project phases and the overall product quality.

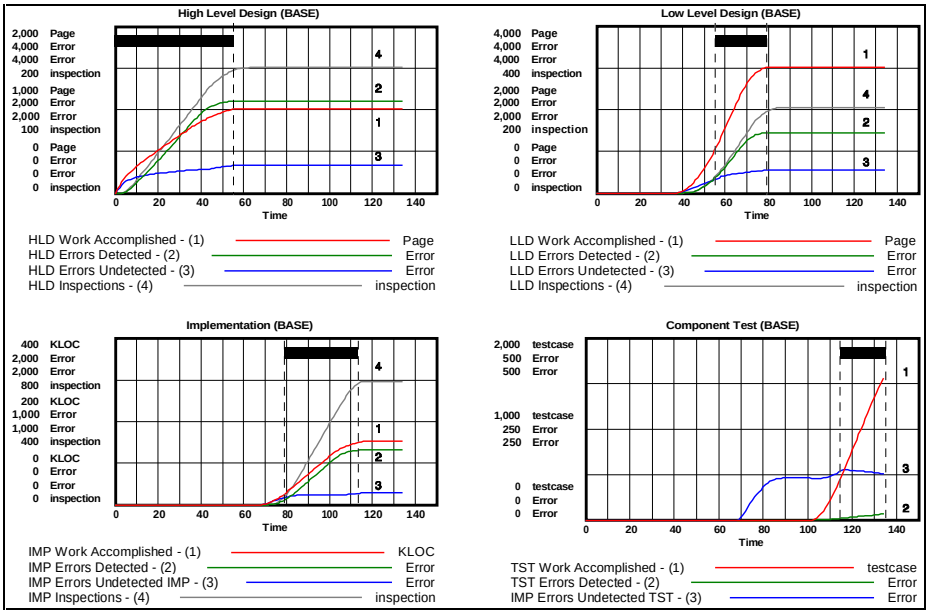

Figure 5: PSIM simulation result for project planning.

Figure 5 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 TEST, work product quality (model variable: "Defects undetected") and other variables are displayed in individual graphs. The phase duration is highlighted with a black bar.

2.4.2 Project control

A possible application of PSIM for project control might be the following: Assume that during the conduct of a project an unexpected change request in-

creases 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? Within a certain limit, she/he can vary two control parameters: workforce and the time planning for intermediate project milestones.

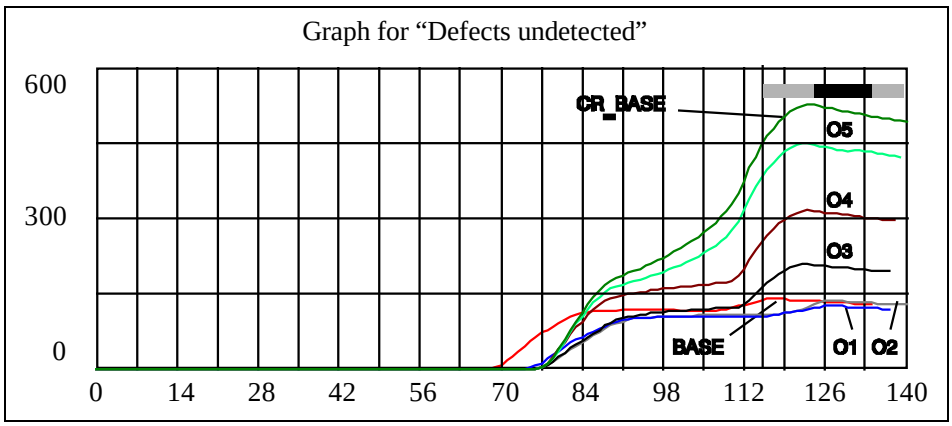

Figure 6: PSIM simulation result for project control.

The curves in Figure 6 show the behaviour of the product quality indicator "Defects undetected" as well as the overall project duration (project stops at the end of test phase). The simulation run BASE indicates the behaviour without occurrence of the change request. The simulation run CR_BASE shows what would happen if the change request occurs and the project manager does not change any of the parameters she/he is controlling (i.e. workforce and milestone planning). The simulation runs O1 to O5 show the project performance if either of the two or both control parameters are altered. The project manager can use these results to find an optimal policy.

2.4.3 Process improvement

PSIM can also be used to evaluate candidate process improvements. Let's assume that an analysis of available data and discussions with project team members have uncovered that formal inspections are done sloppily in the presence of high time pressure, with the consequence of poor product quality. In this situation, project management suggests to implement 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.

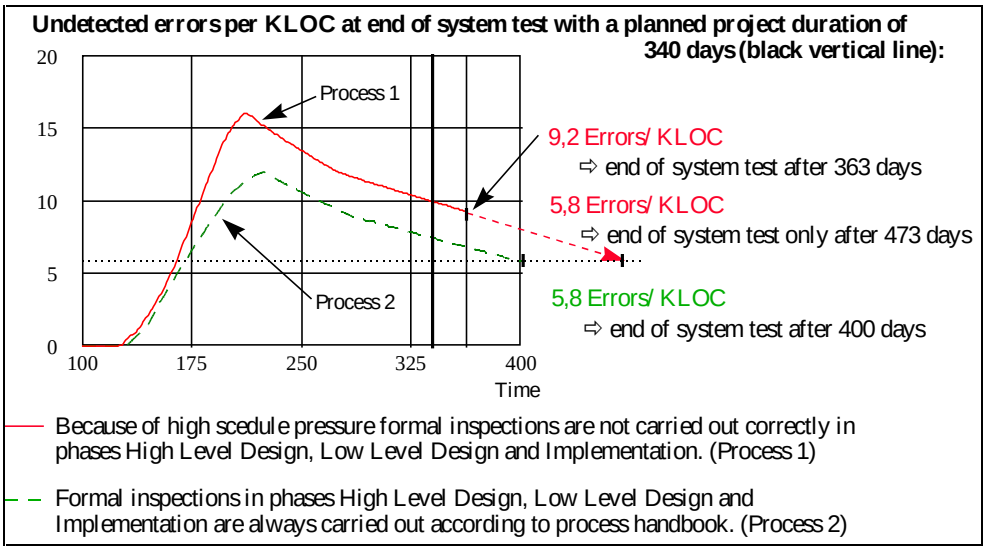

Figure 7: PSIM simulation result for process improvement.

Figure 7 shows simulation results achieved with a differently calibrated model as in the previous sub-sections. It can clearly be observed that the process change leads to improved product quality (process 2). Of course, there is a time delay of about 10% as compared to the old behaviour (process 1). However, the product quality can be increased by more than 40%. If process 1 had to achieve the same product quality as process 2, a time overrun of almost 30% would occur due to extremely extended testing.

2.5 Lessons Learnt

The simulation model PSIM was the first SD application within Siemens PN. Therefore, many valuable lessons could be learnt. Three main clusters of lessons learnt can be distinguished, comprising organisational issues, tool issues, and methodological issues.

A. Organisational issues:

- Before the modelling activity starts, the future model users must be identified. Otherwise, there is a high risk that the model will not be used after completion.
- The purpose for which the SD model will be used by its future users must be stated precisely. Otherwise there is a high risk that the model will be used for the wrong purpose. Also, the model purpose should be

well focused. Otherwise there is a high risk that the model becomes too complex to be useful (i.e. model validity is threatened, and the modelling activity takes too long).

- The future model users should participate in the modelling activity. Otherwise there is a risk that the final model will not meet there needs.
- Process experts and experienced software developers should be interviewed during the modelling activity. Otherwise there is a risk that important aspects of the modelled system are ignored or misunderstood (this threatens model validity).
- The modelling process should proceed iteratively and a first executable model should be at hand soon. Otherwise there is a risk that future model users loose interest.
- The future maintenance of the model should be planned. Otherwise there is a risk that the model will soon become useless for the model users.

B. Tool issues:

- The PSIM model was developed with the help of the SD tool Vensim [Ven95]. The tool was easy to use and no major negative experiences were made. Especially, the availability of a powerful set of analysis tools (different types of graphs, strip graphs, causes trees, tables, etc.) was very much appreciated. One drawback of Vensim worth mentioning is the support for user interface building. It is uncomfortable and should be improved.

C. Methodological issues:

- The SD modelling activity must be goal-driven (i.e. top-down). Otherwise there is a risk to loose focus resulting in a waste of effort and time.
- There is a need for a detailed process description for the planning and execution of the SD modelling activity.
- There is a need for guidelines that help to a) identify existing and b) generate new information which can be used for SD model building.

The IMMoS approach, which will be described in more detail in the remainder of this paper, concentrates mainly on the last of the three issues mentioned in cluster C.

3 Outline of IMMoS

IMMoS is an approach that helps to improve the building of SD models dealing with problems in software development by defining and exploiting links to well-defined and commonly used existing modelling methods which generate static models.

Promising candidates for this kind of complementary modelling activities are a) software Process Modelling (PM) and b) measurement-based quantitative modelling. These methods can be used to provide two types of information:

- Qualitative information, describing the software development process as a whole (i.e. the product flow), as well as the individual products, process steps, resources used, etc.
- Quantitative information, characterising attributes of real-world entities and their relationships.

As a result of a software PM activity [CKO92], qualitative information about software development processes is elicited and represented in a Descriptive Process Model (DPM). A DPM captures a lot of relevant information about what actually happens during software development [BHV97]. Therefore it can help to identify those of the state variables in the SD model that represent physical real world entities in the software organisation.

Valuable quantitative information can be provided by Quantitative Models (QMs) which are derived via statistical analysis from empirical data. According to [BDR96], there are three types of QMs: descriptive QMs (DQMs), predictive QMs (PQMs), and evaluative QMs (EQMs). To make sure that the QM building is based on solid grounds a systematic (i.e. goal-oriented) measurement programme must be applied. Therefore, in the context of IMMoS, it is assumed that measurement is conducted according to the widely used Goal/Question/Metric paradigm [BCR94], following a measurement process as described in [GHW95]. More information on the relationship between QM types and measurement goals is presented in [BDR96]. A recent practical experience of model building based on measurement data can be found in [LSO+98].

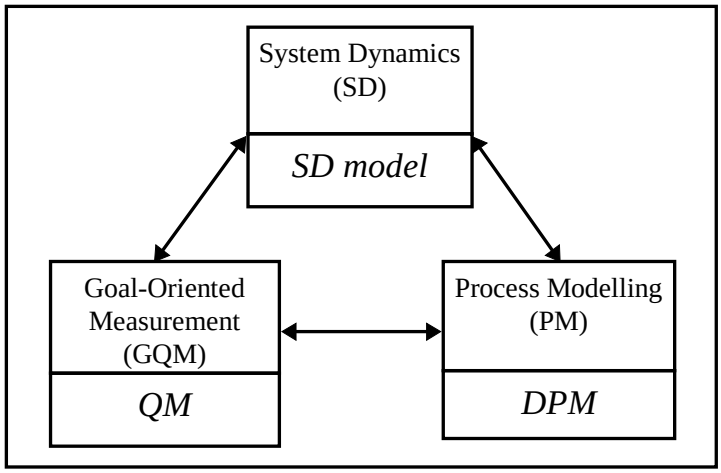

Figure 8: IMMoS overview.

Figure 8 gives an overview of IMMoS, indicating the relationships between the three complementary modelling approaches SD, GQM, and PM, and their main products SD model, QM, and DPM. Each of the three blocks in Figure 9 will be denoted as the *IMMoS components* in the remainder of this paper.

4 Products of IMMoS components

As a preparation for a more detailed description of IMMoS which has to deal with a detailed analysis of the relationships between individual IMMoS components, in this section, the most important products of SD modelling and GQM as well as the main concepts contained in a DPM are briefly introduced.

4.1 SD modelling work products

Relevant work products of SD modelling and simulation are [Ran80][RiP81]:

- Modelling goal: learn, plan, control, improve.
- Reference mode: The reference mode describes the typical or expected behaviour of an entity (state variable) which is in the centre of interest. It is the starting point of any SD modelling activity and serves for defining model boundaries, for generating hypotheses about cause-effect relationships, and for model validation.
- Base mechanisms and causal diagram (joint set of causal relationships).
- Flow graph and associated mathematical model equations (initial executable simulation model).
- Simulation results (e.g. for test purposes).

Relevant end products of SD modelling and simulation are:

- Final executable simulation model (with adequate user interface).
- Analysed and interpreted simulation results that help to solve the problem defined in the modelling goal.

4.2 GQM work products and QM types

Relevant work products of GQM planning and measurement execution are [GHW95]:

- Measurement goal, specifying object, quality focus, purpose, viewpoint, and context of the measurement programme.
- GQM plan (definition of goal-related questions and metrics).
- Measurement plan.
- Measurement data.

End products of a GQM programme are analysed and interpreted QMs, namely [BDR96]:

- DQMs, i.e. models that describe attributes of certain real-world objects quantitatively,
- PQMs, i.e. models that describe the relationships between certain real-world objects and impacting factors quantitatively, or
- EQMs, i.e. models that supports decision making.

4.3 Information contained in a DPM

A DPM captures the current software development practices and organisational issues of a software producing unit. The content of a DPM is mainly based on knowledge elicitation from process experts and software development practitioners, i.e. actual processes and not official processes are represented [BFL+95]. The relevant real-world aspects are represented by entities and relationships between these entities (e.g., input/output relations, sequences of activities, reporting channels between roles, role assignments to activities, etc.). Entities are formalised in an operational way through attributes which characterise the entities. Examples of attributes are size, complexity, status, time, effort, etc.

Different approaches for descriptive process modelling can be applied. Here, no recommendation of a specific method will be given, only a set of the most important types of entities that typically are contained in a DPM are listed, based on the conceptual framework suggested in [ArK94]:

- Artefacts consumed and produced;
- Activities carried out;
- Agents (with roles) involved;
- Tools used;
- Technologies, techniques, and methods used;
- Relationships between activities and artefacts (i.e. flow of artefacts);
- Assignment of roles to activities;
- Usage of tools in activities;
- Application of technologies/techniques/methods in activities;
- Relationships between products (i.e. product hierarchies);
- Relationships between roles (i.e. communication network);
- etc.

A comprehensive formal schema for process models that supports integration with measurement can be found in [BeW97].

Figure 9 presents an overview of the basic components typically contained in a DPM. Using the object-oriented notation of UML (Unified Modelling Language [UML97]) it shows in the upper part classes that can be used to model the core process information (i.e. activities, artefacts, and agents), as well as their major associations. The product flow consists of activities that produce artefacts and

use resources like tools, rooms, physical equipment, artefacts, etc. The lower part shows how agents are linked to activities, as well as classes that can be used to model the organisational structure. Each entity can have a list of characterising and potentially measurable attributes, like status, size, duration, etc.

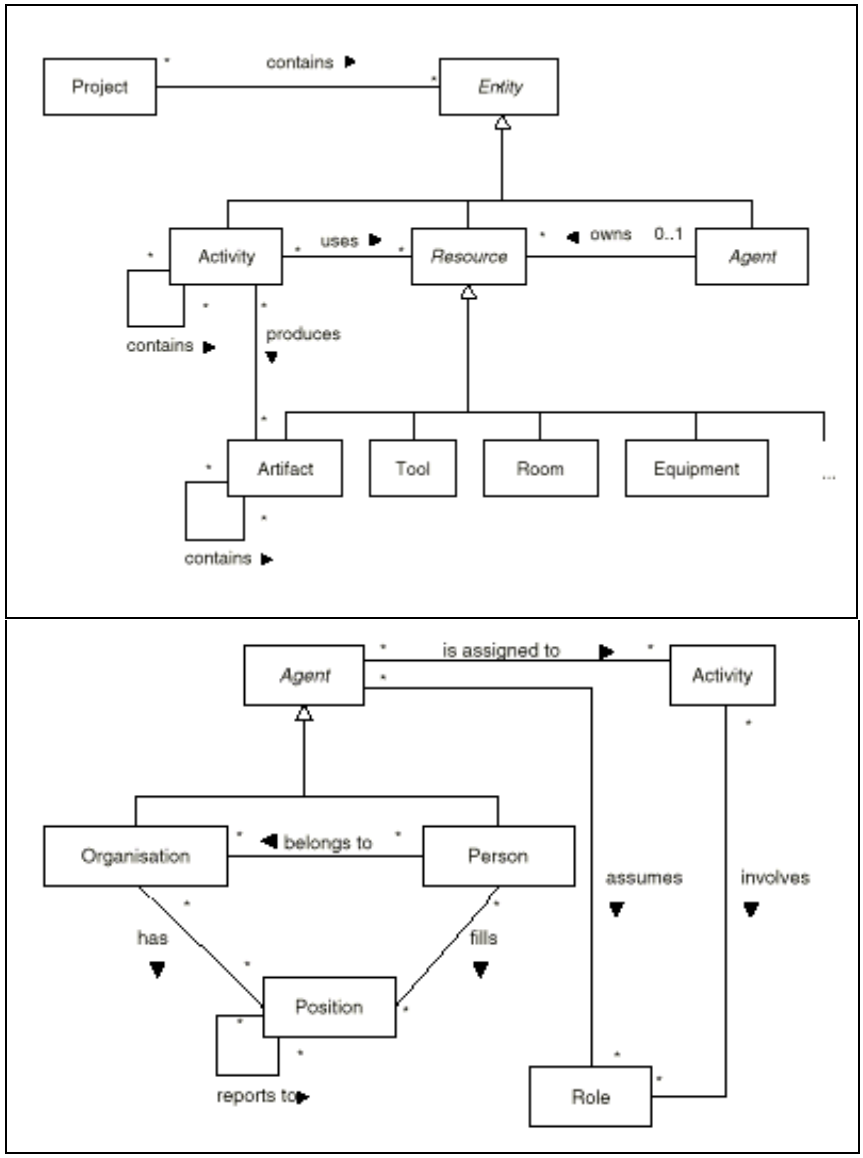

Figure 9: Main DPM components [BeW97].

5 Relationships among IMMoS components

With the information provided in the previous section, a detailed description of IMMoS can be provided in this section. First, a complete overview of the whole set of possible interdependencies between the three IMMoS components is illustrated, then a more detailed insight into the relationships between individual products of the IMMoS components is given. Finally, a short example on the integration of SD model building with information derived from a DPM and an associated PQM is presented to demonstrate the benefits of IMMoS.

5.1 Overview of IMMoS interdependencies

The various relationships between the individual modelling stages as well as the work products and end products of SD, GQM and PM can be identified. A summary is sketched in Figure 10, indicating:

- Model user: role (or agent) and goal(s).
- Modelling stages: descriptive and prescriptive PM, GQM planning and execution, SD modelling and simulation.
- PM, GQM, and SD work products: measurement goals, GQM and measurement plans, SD reference mode, causal diagram, SD model (flow graph with model equations).
- PM, GQM, and SD end products: DPM, Prescriptive Process Model (PPM), measurement data with related QM(s), simulation results.
- Reality: industrial software development.
- Relationships (depicted as arcs) between model user, modelling stages, modelling work/end products, and reality.

When and how the products of the three complementary modelling approaches are combined depends on the goals of the potential model user. There are four different types of goals that can be supported by a modelling activity: to learn, to plan, to control, and to improve. A PPM (grey shaded in Figure 10) is only needed to specify how an intended process change will look like. As soon as the process change is successfully implemented, the PPM automatically becomes the new DPM.

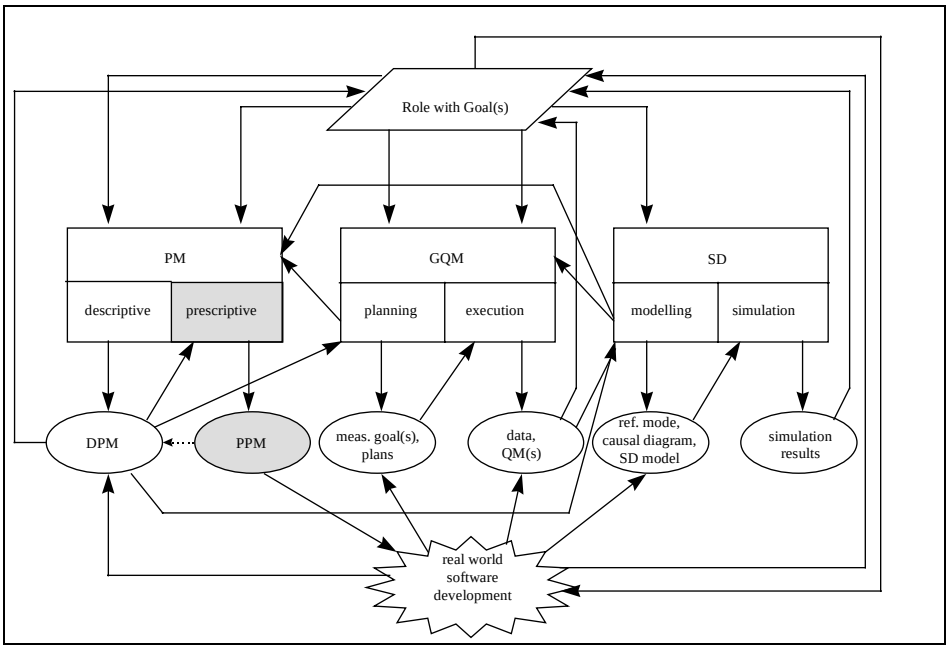

Figure 10: Interaction between IMMoS components.

Three short examples will be given to illustrate the contents of Figure 10.

Example 1 (learn): The role *process engineer* wants to better understand the existing communication channels between different functions in the software organisation. For that purpose she/he sets up a DPM, e.g. using the Strategic Dependency Modelling [Yu94] method which is specifically suited for this purpose. The DPM provides qualitative information only. In a next step, the process engineer would like to know which are the most important communication links. For that purpose she/he plans and executes a little measurement programme which provides him with the relevant quantitative information (i.e, a DQM that counts the occurences and duration of interactions between roles).

Example 2 (improve): The process engineer (same as in example 1) wishes to improve the communication in the organisation. Therefore she/he conducts analyses based on the DPM and DQM. These analyses result in a change of the current process model, and a PPM is defined according to which the future communication channels should be established. The success of this process change can later on be evaluated through a specifically tailored measurement programme. If the evaluation is positive, i.e. the process change turns out to be a real process improvement, the PPM automatically becomes the new DPM.

Example 3 (learn and improve): Now, the process engineer wants to analyse (and subsequently improve) the effectiveness of the software development activities regarding product quality. For his analysis, again, a DPM and several DQMs will be an important input. In analogy to example 2, based on his analyses, the process engineer suggests a process change that is expected to increase the product quality. To reduce the risk of failure, this time, however, the process engineer wants to simulate the effects of the process change before its implementation. This can be done by using an EQM in combination with either a PQM or a SD model. If the simulation shows positive results, the change is implemented and evaluated (cf. example 2).

5.2 Detailed description of interdependencies between IMMoS components

A complete description of the detailed relationships between individual products of IMMoS components, including DPM, is presented in the following.

5.2.1 Relationships between GQM and DPM

The detailed description of the relationships between GQM planning and information contained in a DPM covers the following aspects:

- DPM ↔ Measurement goal: Usually, the object which 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 process modelling activity is initiated.
- DPM → GQM plan: The definition of the measurement programme is done via so-called abstraction sheets (for details cf. [GHW95]). Usually, the set of entities and attributes gathered in abstraction sheets is determined by the information contained in the organisation-specific DPM.
- DPM → Measurement plan: When, how and by whom measures that have been defined in the GQM plan are collected is usually 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 [BDT96].

Of course, the results from the application of a QM can be useful input to process modelling. For example, an EQM can be used to evaluate a suggested process change as specified in a PPM, or a PQM can indicate that either proc-

ess alternative A or process alternative B should be followed, depending on the actual values of certain explanatory variables (cf. example in Figure 11).

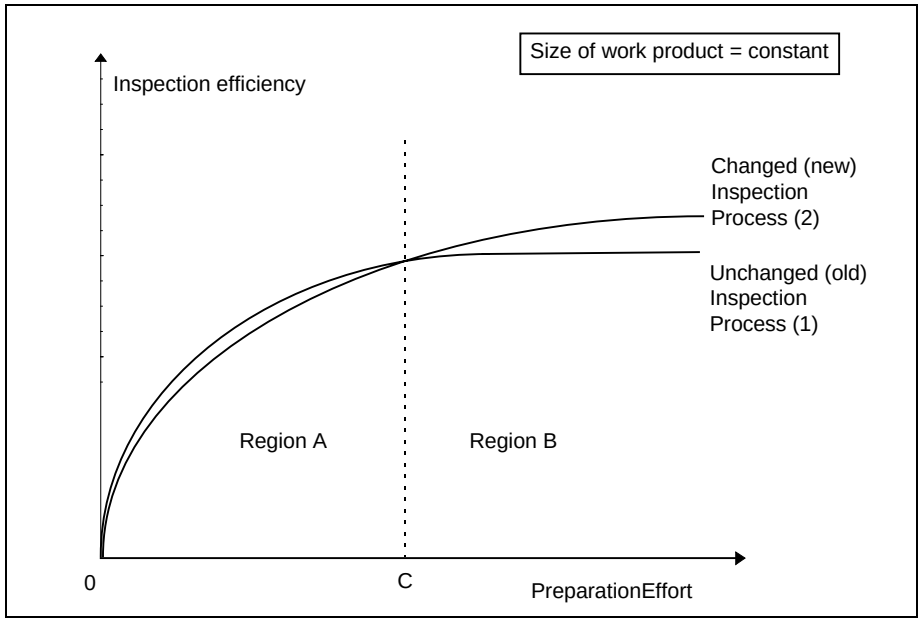

Figure 11: Efficiency characteristic of two different inspection processes.

5.2.2 Relationships between SD and DPM

The detailed description of the relationships between SD modelling and the information contained in a DPM covers the following aspects:

- Indirect relationship via measurement (see above).
- Direct relationship 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 product flow to SD concepts in the flow graph.

Again, the results of simulation runs might be used for (prescriptive) process modelling.

5.2.3 Relationships between SD and GQM

The detailed description of the relationships between SD modelling and GQM covers the following aspects:

- Measurement data → Reference mode: If adequate measurement data is available, it can be used to define the reference mode.
- 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 goal definition. Three of the five aspects are determined by the reference mode. 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 (quantitative characterisation of attribute(s) over time).
- Base mechanisms (causal relationships) → Measurement goal (and GQM plan): Base mechanisms define cause-effect relationships which are supposed to generate the reference mode. 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 (characterisation of attribute relationships) 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.
- GQM plan → Base mechanisms (causal relationships): During the definition of GQM plans (via so-called abstraction sheets [GHW95]) there is a chance that new attributes are identified which are considered as relevant for defining the cause-effect relationship under consideration. These new attributes can be included in the mental model represented by the set of base mechanisms.
- Measurement results → Base mechanisms & model equations: Measurement results that come out of an adequately defined measurement programme can help to identify and define cause-effect relationships qualitatively (base mechanisms) and quantitatively (model equations). For that purpose, statistical modelling and data mining techniques (e.g. Rough-set approach [Ruh96] or Optimised Set Reduction (OSR) [BBH93]) can be applied.

5.3 IMMoS example scenario

In this sub-section, an example scenario of a hypothetical SD modelling activity that is linked to PM and GQM is presented to further illustrate IMMoS.

Adopting the perspective of a SD modelling project, the example scenario is organised according to Figure 12. It begins with the description of the start situation (white area), and then concentrates on the links between SD (dark grey area), GQM (grey area), and PM (light grey area) as follows:

- SD modelling triggers PM and exploits the resulting DPM;
- SD modelling triggers GQM-based measurement and exploits resulting DQMs and PQMs;
- SD simulation results serve for process analysis and subsequent identification of potential improvements, eventually resulting in a PPM for specification of the candidate process improvements;
- Evaluation of the process changes via EQMs and use of the (re-validated) SD model for project planning and control.

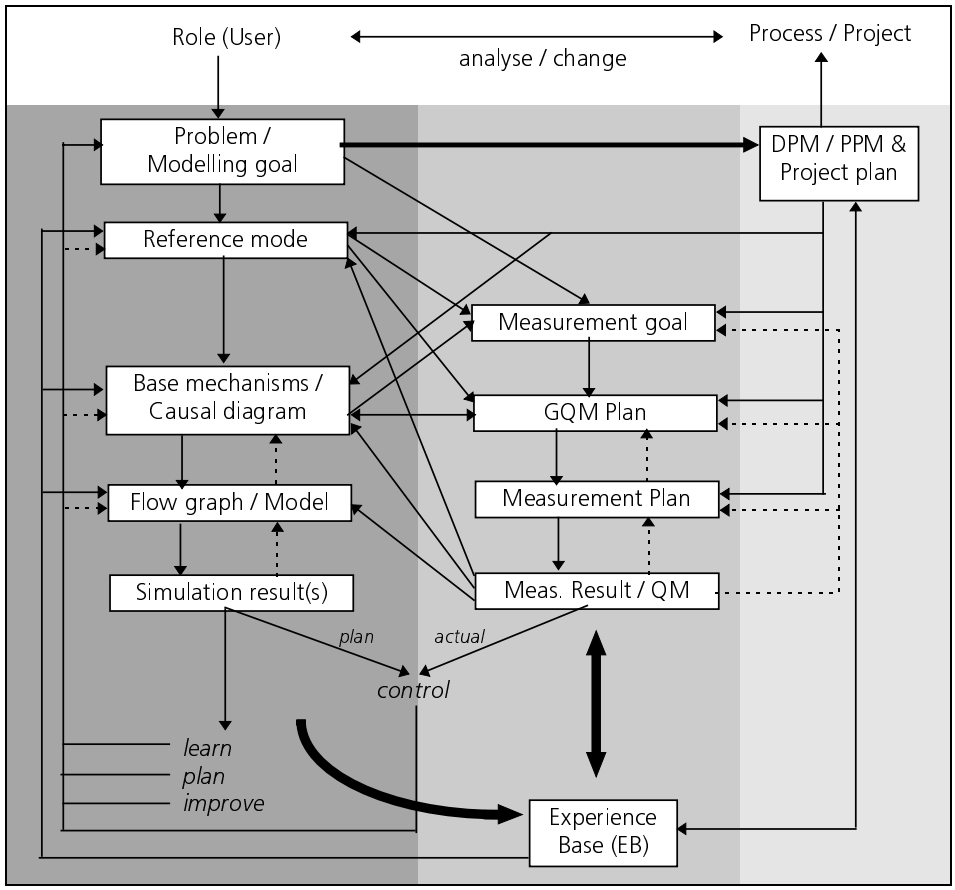

Figure 12: IMMoS example scenario.

The work and end products (white boxes) listed in the respective grey columns coincide with those mentioned in Section 4. The box *Experience Base* indicates that the organisation might want to build up and use a repository of reusable SD, GQM, and PM products.

5.3.1 Start situation and goals

Assume, that a 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.

To sum up, the SEPG has four goals: to learn (understand process/project dynamics), to improve (change process to achieve better project performance), to plan and control (project performance). The team expects, that a SD simulation model can help to support achievement of their goals.

5.3.2 Relationships SD → PM → SD

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

Figure 13 shows an extract from the SD model's flow graph which is developed based on the information in the DPM and additional analyses.

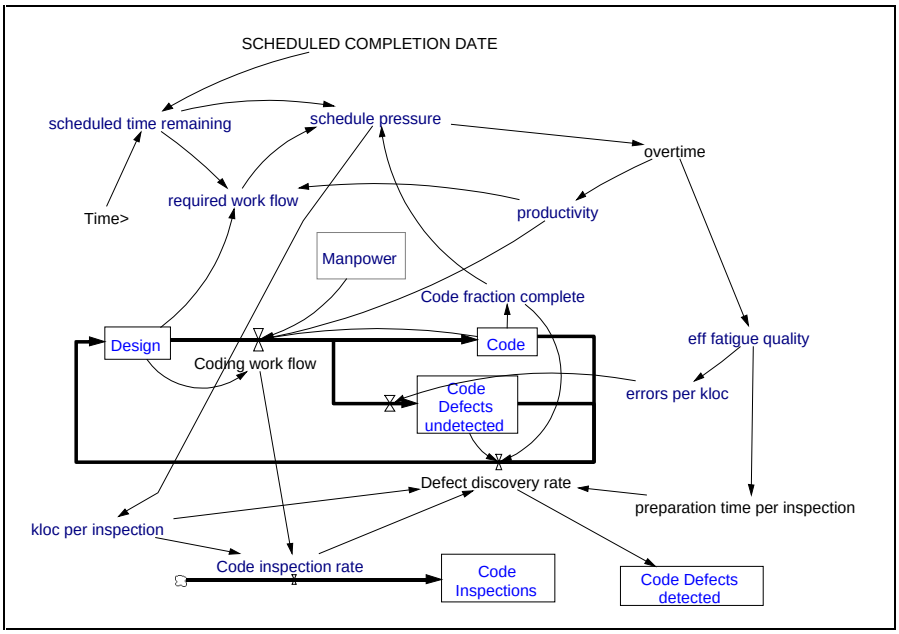

Figure 13: Extract from a view of the SD model's flow graph (implementation phase).

5.3.3 Relationships SD → GQM → PM → GQM → SD

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 parameters 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 SD modelling. The generic specification of a measurement goal for defining a reference mode looks like this:

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

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.

Empirical data can also play a crucial role for the identification of causal relationships. For, example, correlation analysis can be applied to define the SD model variable *Defect discovery rate* in Figure 13. The defect discovery rate defines the number of defects that are detected during a work day. More precisely, it is the sum of the defects found in all the code inspection that were carried out on a work day. The number of inspections is represented by the variable *Code inspection rate*. The (average) number of defects found in a code inspection is a function of the variables *Kloc 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 applied to empirical data.

A fully documented example of such a measurement-based analysis can be found in [BLW97]. The authors describe how a measurement programme was specified, and how the statistical analyses were conducted to derive a PQM on inspection effectiveness (where effectiveness = number of detected defects / size). Of course, similar measurement-based analyses can be conducted for other SD model variables. Sometimes, however, it is very time and effort consuming, or even impossible to carry out an appropriate empirical analysis. In these cases, the only solution is to rely on expert opinion. And often, due to lack of adequate data, empirical analysis cannot result in a precise definition of SD model equations, but still provide enough evidence for an association or correlation between certain model variables. This more qualitative information is still very useful when developing causal diagrams.

5.3.4 Relationships SD → PM / GQM → SD

In the course of building the SD model, the SEPG has already gained much new insight about 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 *to learn*.

Having the SD model at hand, the analysis and interpretation of the simulation outcomes might trigger several process improvement suggestions (goal: *to improve*). To decrease the risk of loosing money with unsuccessful pilot projects,

the improvement suggestions should first be investigated by implementing the intended process changes in the SD model 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.

Of course, before a process change is implemented, it must be specified in a PPM, and after implementation it must be evaluated. This can only be done by using a measurement based EQM that compares the performance of the new process to a baseline. If the improvement is actually confirmed this will also support the validity of the SD model, and the project leaders might use it for project planning, risk management and project control (goals: *to plan* and *to control*).

Process engineers and project leaders might even use the SD model as a tool for process or project benchmarking. This can be done by comparing the process/project behaviour predicted by the SD model with control data derived from an appropriate measurement programme (goal: *to control*). Assuming that the SD model is 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 a decay of the inspection process might result in an reinforcement of the guidelines and prevent from future loss of money (i.e. during maintenance).

6 Conclusion

Motivated by lessons learnt from an SD pilot application in an industrial setting the integration of System Dynamics (SD) with well-known static modelling approaches that are widely used in software engineering was suggested. As promising candidates for complementary modelling methods, Process Modelling (PM) and goal-oriented measurement according to the Goal/Question/Metric (GQM) paradigm were identified. The integrated development and usage of SD models with PM and GQM has been given the acronym IMMoS (Integrated Measurement, Modelling, and Simulation).

In this paper, the mutual interdependencies between the three IMMoS components have been presented and the relevance of IMMoS was illustrated through examples.

The work on IMMoS emerged from an industrial collaboration between Fraunhofer IESE and Siemens Corporate Research. In the course of this collaboration, the following work has been done:

- Description of scenarios for the usage of SD in software industry.
- Clarification of the relationships between SD models and DPMs, as well as SD models and QMs.
- Definition of a detailed SD process model tailored to the needs of software industry.
- Development of supportive guidelines and templates.

The future work will mainly concentrate on:

- Research towards a full integration of the three IMMoS components. This should result in a comprehensive formal and uniform IMMoS process model that combines the SD modelling process with the GQM planning and execution process as defined in [GHW95], and with a recommended (or standard) PM elicitation process.
- Application of IMMoS in industrial environments (pilot projects and case studies).

7 References

[AbM91] Abdel-Hamid TK, Madnick SE: Software Project Dynamics – an Integrated Approach, Prentice-Hall, 1991.

[ArK94] Armitage JW, Kellner MI: "A Conceptual Schema for Process Definitions and Models". *Proc. 3rd Int'l Conference on the Software Process* (ICSP 3), IEEE Computer Soc., pp. 153-165, 1994.

[BFL+95] Bandinelli S, Fuggetta A, Lavazza L, Loi M, Picco GP: "Modeling and Improving an Industrial Software Process". *IEEE Trans. on SE*, Vol. 21, No. 5, pp. 440-453, May 1995.

[BCR94] Basili VR, Caldiera G, Rombach HD: "Goal Question Metric Paradigm". In Marciniak JJ (ed.), Encyclopedia of Software Engineering, Vol. 1, pp. 528-532, John Wiley & Sons, 1994.

[BHV97] Becker U, Hamann D, Verlage M: "Descriptive Modeling of Software Processes". *Proceedings of the Third Conference on Software Process Improvement (SPI '97)*, Barcelona, Spain, December 1997.

[BeW97] Becker U, Webby R: "Towards a Comprehensive Schema Integrating Software Process Modeling and Software Measurement". Technical Report, No. 021.97/E, Fraunhofer IESE, Kaiserslautern, July 1997.

[BBH93] Briand LC, Basili VR, Hetmanski C: "Developing Interpretable Models with Optimized Set Reduction for Identifying High Risk Software Components", IEEE Trans. Software Engineering, Vol. SE-19 (11), pp. 1028-1044, 1993.

[BDR96] Briand LC, Differding CM, Rombach HD: "Practical Guidelines for Measurement-Based Process Improvement". *Software Process Improvement and Practice*, Vol. 2, No. 4, pp. 253-280, 1996.

[BLW97] Briand LC, Laitenberger O, Wieczorek I: "Building Resource and Management Models for Software Inspections". *Technical Report*, No. 041.97/E, Fraunhofer IESE, Kaiserslautern, December 1997.

[BDT96] Bröckers A, Differding C, Threin G: "The role of software process modeling in planning industrial measurment programs". *Proceedings of the Third International Software Metrics Symposium*, Berlin, IEEE Computer Society Press, March 1996.

[CKO92] Curtis B, Kellner M, Over J: "Process Modeling", *Communications of the ACM*, Vol. 35, No. 9, Sept. 1992.

[For61] Forrester JW: *Industrial Dynamics*. Productivity Press, 1961.

[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 Kaiserslautern, October 1995.

[LSO+98] van Latum F, van Solingen R, Oivo M, Hoisl B, Rombach HD, Ruhe G: "Shifting to Goal-Oriented Measurement in Industrial Environments - Experiences of Schlumberger", *IEEE Software*, pp. 78-86, January-February 1998.

[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.

[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 T, Sherif JS: "Software-Engineering Process Simulation Model (SEPS)". Journal of Systems and Software, Vol. 38, pp. 263-277, 1997.

[Mad94] Madachy RJ: *A Software Project Dynamics Model for Process Cost, Schedule and Risk Assessment*. PhD Dissertation, University of Southern California, December 1994.

[Pfa94] Pfahl D: *PSIM Bedienungsanleitung* (in German), Siemens AG, 1994 (unpublished).

[Pfa95] Pfahl D: "Software Quality Measurement Based on a Quantitative Project Simulation Model", *Proc. of the European Software Cost Modelling Conference* (ESCOM '95). Rolduc, The Netherlands, 15-17 May 1995.

[PfK95] Pfahl D, Kammerer R: "Optimierung von Projekten und Prozessen in der Software-Entwicklung mit PROSYD". *ZFE Technical Report* (in German), No. Z0950040, Siemens AG, March 1995.

[Ran80] Randers J (Ed.): Elements of the System Dynamics Method. Productivity Press, Cambridge, 1980.

[RiP81] Richardson GP, Pugh AL: *Introduction to System Dynamics Modeling with DYNAMO*. Productivity Press, Cambridge, 1981.

[Ruh96] Ruhe G: "Qualitative Analysis of Software Engineering Data Using Rough Sets", *Proc. of the Fourth International Workshop on Rough Sets, Fuzzy Sets, and Machine Discovery*, Tokyo, pp. 292-299, Nov. 1996.

[Tve96] Tvedt JD: *An Extensible Model for Evaluating the Impact of Process Improvements on Software Development Cycle Time.* PhD Dissertation, Arizona State University, May 1996.

[UML97] *Unified Modeling Language*. Version 1.0. Rational Software Company, 1997.

[Ven95] *Ventana Simulation Environment (Vensim) - Reference Manual, Version 1.62*, Ventana Systems, Inc., 1995.

[WaP94] Waeselynck H, Pfahl D: "System Dynamics Applied to the Modelling of Software Projects". *Software Concepts and Tools*, Vol. 15, pp. 162-176, 1994.

[Yu94] Yu ES: "Modelling Strategic Relationships for Process Reengineering", *Technical Report*, DKBS-TR-94-6, Department of Computer Science, University of Toronto, December 1994.

Acknowledgements

The authors would like to thank Siemens AG for supporting much of the work presented in this report. Special thanks go to Reinhard Kammerer for initiating the PSIM project, as well as to Jörg Lottermoser for helpful discussions during the project. The review comments on earlier versions of this report provided by Andreas Birk, Khaled El Emam, and Günther Ruhe contained many valuable improvement suggestions which were highly appreciated.

Document Information

Title:	Integration of System Dynamics Modelling with Descriptive Process Modelling and Goal-Oriented Measurement
Date:	May 28, 1998
Report:	IESE-032.98/E
Status:	Final
Distribution:	Public