Virtual Software Engineering Laboratories in Support of Trade-off Analyses

Jürgen Münch, Dietmar Pfahl
Fraunhofer Institute for Experimental Software Engineering (IESE)
Sauerwiesen 6, 67661 Kaiserslautern, Germany
{muench, pfahl}@iese.fraunhofer.de

Ioana Rus
Fraunhofer Center Maryland,
4321 Hartwick Road, #500, College Park, MD, USA
irus@fc-md.umd.edu

Abstract

Due to demanding customer needs and evolving technology, software organizations are forced to trade individual functional and non-functional product quality profiles against other factors such as cost, time, or productivity. The ability to influence or even control these factors requires a deep understanding of the complex relations between process and product attributes in relevant contexts. Based on such understanding, decision support is needed to adjust processes so that they match the product quality goals without violating given project constraints. We propose to use a Virtual Software Engineering Laboratory (VSEL) to establish such decision support cost-effectively. VSELs can be considered as being complementary to existing (empirical) Software Engineering Laboratories. This paper gives an introduction into the cornerstones of VSELs, discusses how they complement traditional empirically based Software Engineering Laboratories (SELs), and illustrates with the help of case examples from industrial and research environments, how to use them in support of product-focused trade-off analyses.

Keywords: Decision support, software engineering laboratories, software process modeling, software process simulation, trade-off-analyses.

1. Introduction

Trade-offs are common phenomena in software development. For example, increase or change in functionality during development may compromise achievement of quality goals, or violate effort and time constraints (or all together). In order to understand trade-offs between concurrent product goals, their relation with overall project goals (and constraints), and the underlying processes needs to be understood.

One way to systematically explore and analyze the complex mutual interrelations of project, process, people, and product in software development is to conduct empirical studies in laboratory-like settings.

The idea of Software Engineering Laboratories (SELs) has been around since the 1970s (Basili et al., 2002). SELs are understood as environments for performing software engineering related experiments in a specific domain and for a specific purpose (Rombach et al., 1993, Rombach, 1999). One of the main reasons why SELs are not too widely spread is the cost and time needed for running experiments in software development, where most projects are struggling to stay within budget and delivery time. Similar to experiments in social sciences, experiments in software engineering always require the inclusion of human subjects and are therefore very expensive. In addition, real-world experiments such as pilot projects or case studies bear the risk of failure.

We propose to complement the concept of a SEL by the concept of a Virtual Software Engineering Laboratory (VSEL). The core element of a VSEL is an extensible set of sufficiently valid Software Process Simulation (SPS) models. We talk of a comprehensive VSEL when the VSEL consists of a systematically developed set of reusable SPS modules which can be composed and tailored in order to support problem solving, decision support or analysis tasks.

One major motivation for creating a VSEL is reduction of experimentation cost and risk. The available SPS models contained in a VSEL help to explore and demonstrate the effects of alternative process adjustments corresponding to different trade-off decisions in a reproducible and explanatory way (Rus et al., 2002). In particular, the multitude of variations of an experiment, which is often necessary to cover different impact factors when exploring alternative decisions, can easily be performed in a controlled manner. Consequently, learning cycles can be shortened, because different starting conditions, external influences, or process characteristics can easily be generated without much effort or time consumption. A VSEL would also significantly support the

automation of data collection for experiments, which in a real world setting is a big overload.

VSELs can provide support to a variety of managerial tasks in software development, i.e., project planning and controlling, decision-making in the context of technology evaluation and process improvement, and demonstration and training (Kellner et al., 1999).

This paper is structured as follows. In Section 2, we briefly characterize VSELs by giving more information about the potentiality of process simulation in the context of software development. In Section 3, we describe the complementarity of VSELs and traditional SELs, and discuss the potential of VSELs to create synergies with SELs by properly combining SPS modeling and application with empirical work. In Section 4, we show with the help of case examples how virtual laboratories can be used to conduct trade-off analyses in the software process. In Section 5, we discuss the conditions under which VSELs can become a feasible option for software development organizations, in particular from the perspectives of efficiency and effectiveness. In Section 6, we sketch future perspectives of evolving the presented concepts by seamlessly integrating VSELs with SELs. Finally, in Section 7, we summarize the conclusions.

2. Virtual Software Engineering Laboratories

Process simulation is a standard technology in many engineering disciplines. In the software process literature, according to our understanding, there is a general agreement that people who understand the static process (i.e., process activities, artifacts, resources, roles, and their relationships), and who have all data needed to reason about it at hand, still have difficulties to anticipate the actual process behavior. This is due to the inherent (dynamic) complexity of software development processes. Building and executing a computer model that reflects all important process interactions, dynamics, information feedback loops, delays, etc., helps to make process experts' (implicit) mental models explicit and precise, and offers the possibility to simulate the process behavior in support of understanding, analysis, prediction, and decision-support.

In software engineering, during the last 15 years, a variety of SPS modeling tools and techniques have been adopted. Discrete-event simulation and continuous simulation approaches such as System Dynamics (Forrester, 1961) are the most prominent ones. It is beyond the scope of this paper to describe and discuss the various SPS modeling paradigms in-depth or to give advice on how to develop SPS

models. The interested reader can refer to (Kellner et al., 1999) for a general introduction into SPS modeling paradigms. Detailed descriptions of SPS modeling methods specialized to instances of the discrete-event and continuous simulation modeling paradigms can be found in (Rus et al., 2003) and (Pfahl and Ruhe, 2002), respectively.

SPS models constitute the core elements of a VSEL. Several authors have pointed out the potential of SPS as an analysis and decision-support tool for software managers (e.g., (Christie, 1999a, Kellner et al., 1999, Waeselynck and Pfahl, 1994)).

Abdel-Hamid was the first to apply simulation modeling in software project management (Abdel-Hamid and Madnick, 1991, Lin et al., 1997). Typical applications of Abdel-Hamid's models focus on project estimation and the effects of project planning on product quality and project performance.

During the last decade many new SPS applications in software engineering have been published, focusing on more specific topics within software project and process management, such as multi-project management (Lee and Miller, 2004), concurrent engineering (Powell et al., 1999), requirements engineering (Höst et al., 2001, Pfahl and Lebsanft, 2000a, Stallinger and Grünbacher, 2001), software process improvement (Christie, 1999b, Pfahl and Birk, 2000, Raffo et al., 1999), strategic planning (Williford and Chang, 1999), software verification and validation (Madachy, 1996, Raffo et al., 2004), software reliability management (Rus et al., 1999), software risk management (Houston et al., 2001), software maintenance (Cartwright and Shepperd, 1999), software evolution (Wernick and Hall, 2004), COTS software development (Ruiz et al., 2004), global software development (Raffo et al., 2003), software outsourcing (Roehling et al., 2000), open source development (Jensen and Scacchi, 2003), agile software development (Mišic et al., 2004), and software engineering training (Drappa and Ludewig, 1999, Madachy and Tarbet, 2000, Pfahl et al., 2001).

Due to the fact that SPS models capture cause-effect relations between project, process, people and product aspects, VSELs can also play an important role in analyzing trade-offs among product qualities, and between elements of product quality profiles and project constraints such as budget, personnel experience and skills, and delivery date.

3. Complementarity of VSELs and SELs

In the introduction we mentioned that VSELs and traditional SELs are complementary. In the following sub-sections, we argue that the combination of VSELs with SELs offers several opportunities for synergy. More concretely, we discuss the conditions under which SPS models can become a powerful tool for decision support and trade-off analyses. When firmly supported by real-life studies and empirical data, SPS models can not only enhance our understanding of observed phenomena in software development projects, but also provide valuable guidance for deciding where to perform further empirical research in order to understand the complex cause-effect relations between project, process, people, and product aspects in software development. Empirical studies and SPS can be combined in such a way that, firstly, empirical knowledge is used for the development and calibration of SPS models, secondly, results from process simulation are used for supporting real-life experiments, and thirdly, real-life experiments and process simulation are performed in parallel (i.e., online simulation).

3.1. Using Empirical Knowledge for SPS

Using a simulation model for prediction and understanding requires that the model reflects real world behavior for the intended contexts as accurately as possible. Simulation models are abstractions of the real world and, therefore, represent only selected aspects and vary in level of detail. Several ways exist for constructing simulation models that represent real world behavior as accurately as possible.

Descriptive software process modeling, for instance, can be used to elicit static process knowledge (such as process and product structure, product flow, or resources). Surveys can be used for identifying state-of-the-practice procedures. Literature studies and expert interviews can be used for identifying influences between input and output parameters.

An important source for the creation of accurate SPS models is empirical knowledge from real software engineering empirical studies (Raffo et al., 1999a). This knowledge can originate from previously performed experimental studies (if the context is comparable) or from experiments that are especially designed to support the development of a simulation model. Experimental software engineering studies can be classified along two dimensions (Basili et al., 1986): “teams” (i.e., groups of engineers, including the extreme case of individuals working independently), and “projects” (i.e., separate problems on which teams work). A

team, for example, is a group of inspectors, and a project can be the inspection of a code document using a specific inspection technique. The number of teams and projects can then be used to classify real-life experiments (see Table 1).

Table 1. Classification of experiments (Basili et al., 1986).

#Teams per project	#Projects	
	one	more than one
one	single project	multi-project variation
more than one	replicated project	blocked subject-project

It should be mentioned that the degree of control varies: On the one hand, single project studies (such as case studies) usually provide only little control and, therefore, do not deliver statistically significant results. The advantage of single project studies is that typically they can be conducted in realistic contexts. On the other hand, blocked subject-project studies allow for a high degree of control and for statistically significant results. The disadvantage of blocked subject-project studies is that they typically have a very narrow focus and it is difficult to conduct those studies in a realistic (i.e., industrial) setting. In the following, we describe how results from these study types can be used for simulation modeling:

- Empirical data from blocked subject-project studies (trends and behavior of software development parameters) can be used for developing simulation models. This data can help to identify patterns and relations between parameters. Furthermore, blocked subject-project studies can be especially designed to determine the effects of changes in defined contexts, i.e., detailed influence relations can be quantified.
- Empirical data from single project studies can be used to initialize the input parameters and to calibrate a simulation model to a specific context (e.g., project, environment).
- Replicated project studies can be used to increase the significance of empirical results. Consequently, using data from replicated project studies in simulation models improves the empirical basis of the model and can lead to better calibrations.
- Multi-project variation studies involve a single team in a set of projects. Multi-project variation data can also be used for calibrating the simulation

model to a specific context. Using empirical data from multi-project variation studies requires careful analysis of learning effects.

The use of data from experimental studies can involve the application of further data analysis steps. Quantifying relationships can be done, for instance, by using stochastic analysis or data mining techniques. Data mining techniques can be used to generate new rules to describe relations that were not considered before or were not elicited by expert interviews. The available empirical data determines the data mining technique. An example for applying data mining techniques to simulation model development can be found in (Neu et al., 2002).

Furthermore, data from experimental studies can be used to validate a model against real world behavior, and to further tune the model. Deviations between model behavior and real world behavior can exist and can have several causes. Examples for such causes are a mismatch between the real context and the scope of validity of the model, incorrect assumptions in the model, or the non-consideration of an influence factor in the model.

3.2. Using SPS for Real Experiments

Simulation can be seen as a means to improve the planning, design, and analysis of real experiments. Additionally, it can be applied to generalize the results of real experiments (within limits). In detail, process simulation can be used for supporting real experiments in the following ways:

- New hypotheses can be investigated by simulation, before performing expensive real experiments. As a consequence, uncontrolled and controlled parameters of real experiments can be determined more precisely. Sensitivity analyses, for instance, allow for investigations with respect to parameter combinations that have significant impact on the variables of interest. Investigating a hypothesis with a simulation model might also lead to observations that are questionable. Both situations can motivate real experiments. Using simulation helps to determine which real experiments are needed most, and to prioritize real experiments.
- Simulations of empirical studies and their variations can be executed in different contexts (by changing simulation parameters' values) with negligible cost and time. Independent variables in real experiments are typically fixed or varying in an uncontrolled manner. During the design of a real experiment,

it should be taken into consideration that the varying uncontrolled variables do not have an impact on the dependent variables (i.e., the parameters of interest). The independent variables that are fixed in a real experiment characterize the context of the experiment. In order to apply the model to different (but similar) contexts, a variation of these variables is necessary. This can be done cost-effectively with simulation. It should be mentioned that the variation of the context affects the external validity of the results. A careful analysis of threats to validity should be performed (see (Judd et al., 1991) for details).

- The scale of empirical studies can be enlarged. The variability of controlled parameters in real experiments is typically small. The variation range of a controlled variable can be extended within a simulation model. Furthermore, simulation models can be developed to cover the scope of several real experiments. In both cases, a careful validity analysis is necessary before using the simulation results.
- Global effects can be studied by modeling the bigger process that includes the empirically studied sub-process. Due to practical constraints, real experiments very often cover only small parts of a software process (e.g., an inspection process). Practical questions often require a broader process view. An example question is, how the inspection effectiveness impacts the overall cost of a project. Modeling the global process in a simulation model can help to determine the global effects of a local real experiment.
- Simulation can be used to determine data needs for real experiments. The development of a simulation model before a real experiment enforces the determination of the dependent and independent variables. This alleviates the instrumentation of the real experiment with metrics. The derivation of metrics can be done in a goal-oriented way (as described, for instance, in (Briand et al., 1996)). In addition, the development of a simulation model reveals what kind of data is available and what is missing. A next step towards the design of a real experiment is to adjust the context and determine the level of control.

3.3. Online Simulation

Real experiments and process simulation can be combined in such a way that they are performed in parallel. This so-called online simulation offers the following additional capabilities to the advanced software engineering laboratory:

- Real processes and virtual processes can be performed in parallel and influence each other. The scale-up of real experiments can be performed in real time. This allows for examining effects in a real setting without neglecting the global effects.
- Data from real-life experiments can be validated in real time. This offers additional data validation capabilities to traditional validation techniques (as described in (Basili and Weiss, 1984)).
- Simulators can be used as a means for training. Simulation allows for practice-oriented and scenario-oriented training of software developers, managers or other target groups. The learner can, for example, act as if he/she was playing his role in the real world and see the consequences.

4. Case Examples

The following case examples are presented to illustrate how SPS models, the core element of a VSEL, can become a valuable tool in software trade-off analysis. The selection of the case example is not meant to completely cover all types of product-focused trade-off analyses that possibly can be supported by VSELs, but it does provide some insights on how to use different SPS models for various purposes and in various contexts for typical decision problems associated with product-focused trade-off analyses. Each example briefly describes the following aspects:

- Definition of simulation goal defined in terms of role of SPS user, SPS model scope, dynamic focus of SPS model, purpose of SPS model application, environment of SPS model application.
- Characterization of SPS model in terms of modeling language/paradigm, model size, key input and output variables of the model, empirical base for model development, and benefit for focusing real experiments.
- Example results of SPS model application, i.e., examples of trade-off effects analyzed

For the definition of simulation goals, the IMMoS (Integrated Measurement, Modeling, and Simulation) method (Pfahl and Ruhe, 2002) provides a template that has been adapted from the goal definition template typically used in goal-

oriented measurement programs (Briand et al., 1996). The SPS goal definition template answers the following questions:

- Who are the SPS model's intended users? (Role)
- What are the SPS model's boundary and granularity? (Scope)
- What dynamic behavior is in the focus of interest (Dynamic focus)
- Why is the SPS model developed? (Purpose)
- For which and in which organizational environment is the SDM developed and applied?

4.1. Process SIMulator (PSIM)

PSIM (Process SIMulator) is an industrial SPS model that was developed for one of the former Siemens telecom divisions to support process and project managers. Table 2 summarizes the goal definition.

Table 2. PSIM goal definition template.

Role	Process and project managers
Scope	Single project (from phases high-level design to system test)
Dynamic Focus	Impact of milestone planning, product size, product complexity, manpower, skills, and quality assurance on project duration, project effort, and product quality
Purpose	Project Planning and Analysis
Environment	Software development organization within Siemens Private Networks

The PSIM model was developed using the System Dynamics approach (Forrester, 1961), an instance of the continuous simulation modeling paradigm. The empirical base of the model consists of results from focused interviews with software development experts, process handbooks, and experience data bases from similar previous projects (Pfahl and Lebsanft, 2000b).

The PSIM model can be used by project managers for project planning and control, and by process managers to analyze process improvement potentials. In all cases, trade-offs between project duration, effort consumption (or effort constraints), and product quality play an important role.

In the following sub-sections, the application of the PSIM model for project

planning, project control, and process improvement is illustrated through three different fictitious examples.

Project Planning

Under the assumption that PSIM is a valid predictive model, a project manager can 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.) he/she predicts the duration of the individual project phases and the defect detection dynamics.

Figure 1 shows the output of a simulation run generated from a project specific set of input parameters. For each of the phases high-level design (HLD), low-level design (LLD), implementation (IMP) and unit test (TST), the expected defect detection over time is shown. The total project duration is 298 workdays. The dynamics of defect detection is expressed in terms of the cumulative number of defects (model variables: "Defects Detected") detected in the set of related artefacts, i.e. the set of high-level design documents, the set of low-level design documents, and the set of code modules.

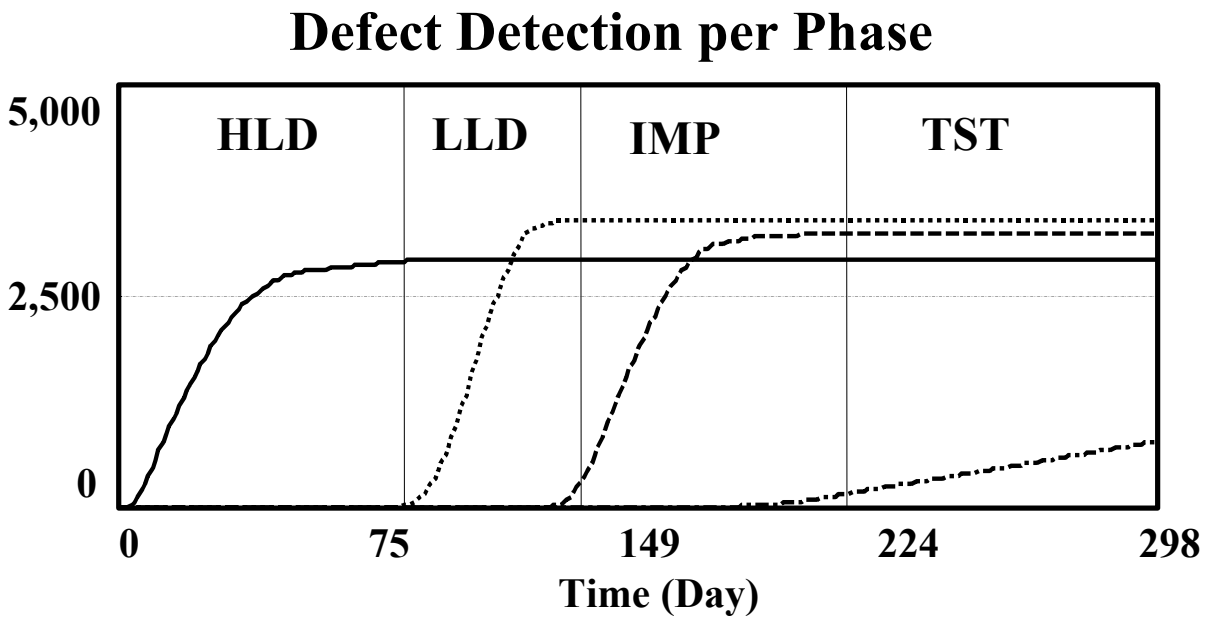

Figure 1. Simulation results for product quality and project duration (all phases).

Project Control and Re-Planning

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?

The curves in Figure 2 show the variation of product quality ("Defects Undetected") during implementation and unit test phases, as well as the overall project duration until the end of the unit 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/she 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. The goal of the project manager is to adjust the control parameters such that the trade-offs between duration, effort, and quality become optimal. In the given context, optimality is equivalent to achieving (almost) the same product quality as in the BASE case without spending too much extra effort and/or time.

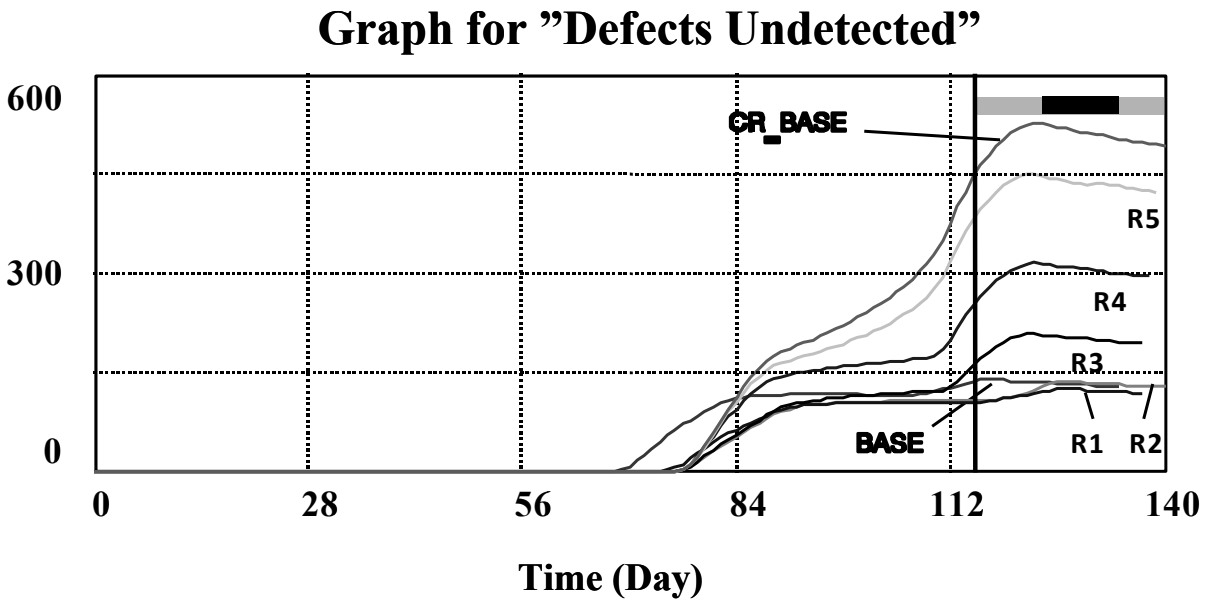

Figure 2. Simulation results for product quality and project duration (implementation and unit test).

Table 3 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, i.e. KLOC), which is a direct consequence of the change request, as well as the overall project duration, project effort consumption, and the absolute and relative product quality are clearly visible in simulation run CR-BASE. By a step-wise alteration of the control parameters, the project manager can use the simulation results to find an optimal policy.

Table 3. Simulation results for project re-planning.

Simulation Run	**Input (controlled by management)**		**Output (at end of phase TST / project end)**			
	Project		**Project**	**Product**		
	Workforce [persons] // Effort [person weeks]	Milestone Planning [calendar weeks]	Duration [calendar weeks]	Quality (absolute) [undetected defects]	Size [KLOC]	Quality (relative) [undetected defects / KLOC]
BASE	40 // 5360	(45, 80, 115, 134)	134	127	151	0.84
CR_BASE	40 // 5600	(45, 80, 115, 134)	140	492	171	2.88
R1	*48 // 6576*	*(60, 87, 125,* 134)	137	118	171	0.69
R2	*45 // 6300*	*(60, 86, 125, 137)*	140	126	171	0.74
R3	*45 // 6165*	*(60, 86, 117,* 134)	137	193	171	1.13
R4	*42 // 5796*	*(60, 80, 116, 135)*	138	293	171	1.71
R5	*41 // 6699*	*(60, 80,* 115*, 136)*	139	421	171	2.46

The spider plot in Figure 3 shows how the relative differences to the original base case (run BASE) first tend to balance out (runs R1 to R3) but start to generate quality loss (R4 and R5), if the goals schedule and/or budget adherence become too important. If the model variables representing project duration and product quality are appropriately combined in a weighted utility function, the project manager could even use the built-in optimisation feature of the SD modelling tool Vensim (Ventana Systems, 2004) to perform an automated search for the best value assignment for the control parameters.

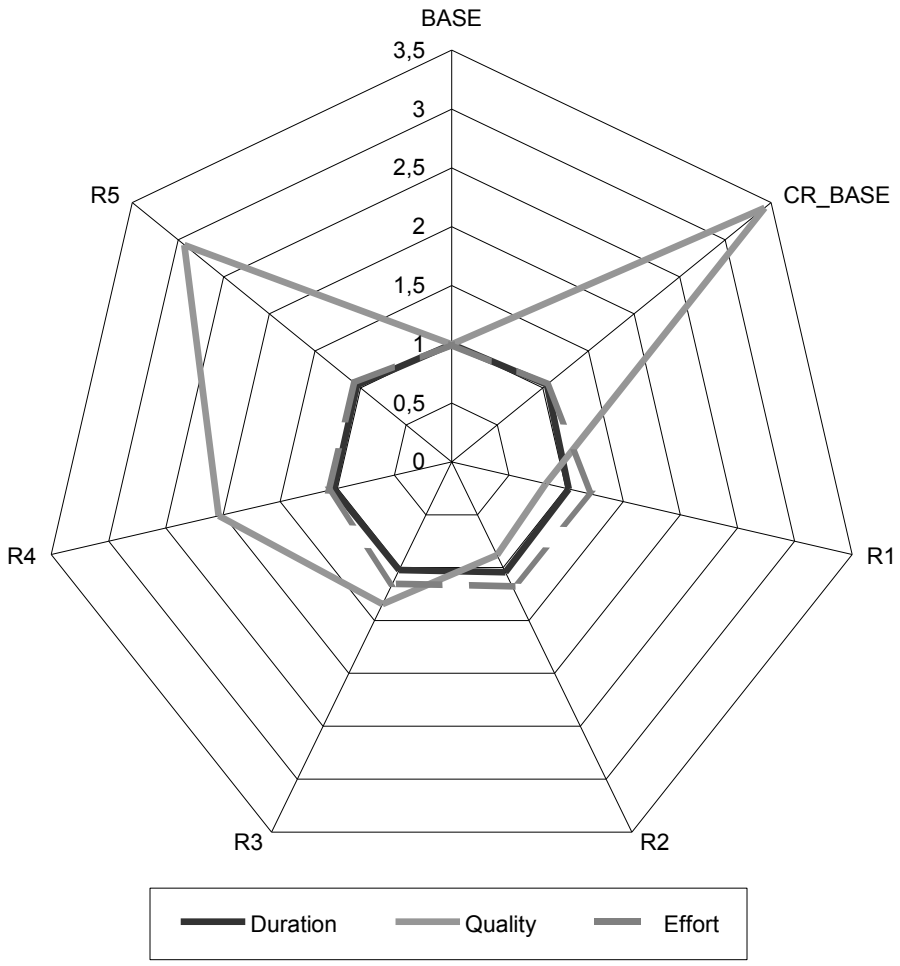

Figure 3. Trade-offs between product quality, project duration, and effort consumption.

Process Improvement

PSIM can also be used to explore the impact of candidate process improvement strategies on project performance (cf. Figure 4). 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).

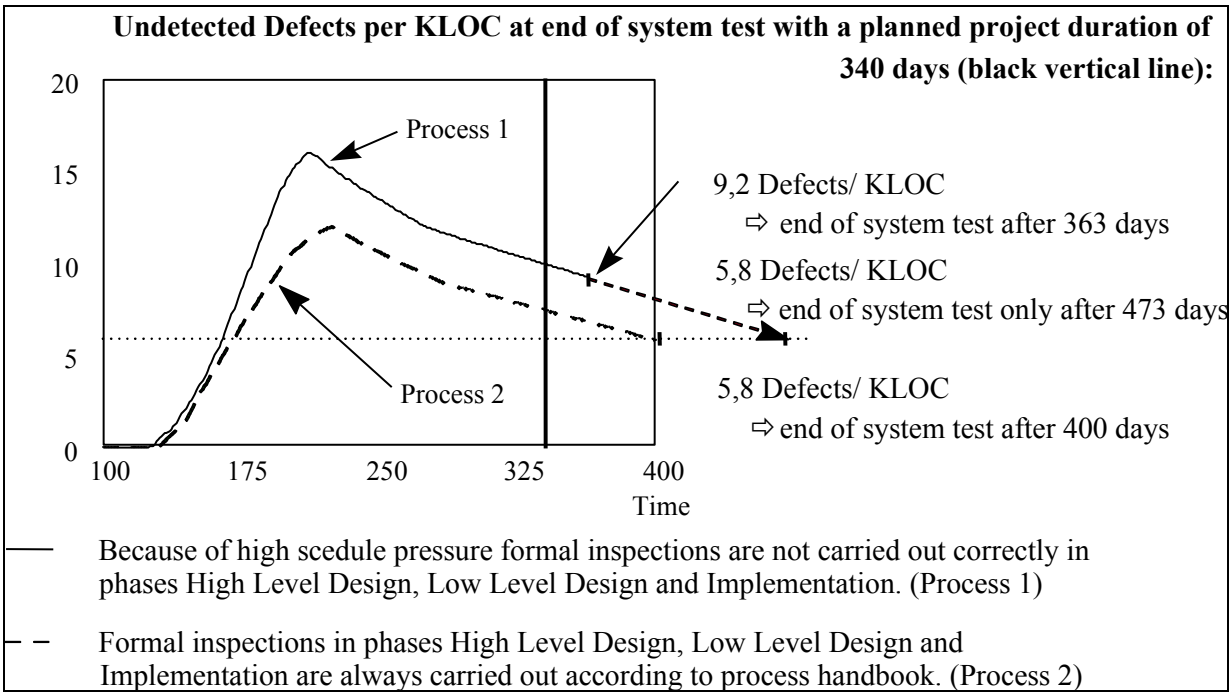

Figure 4. Simulation results for process improvement.

Table 4 shows the simulation results of a fictitious example. 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 defects 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 defects 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.

Table 4. Simulation results for process improvement.

Simulation Run	**Input (at start of phase HLD / project start)**			**Output (at end of phase TST / project end)**			
	Work-force [persons]	Planned project duration [days]	Planned product quality [undet. defects / KLOC]	Actual project duration [days]	Actual product quality [undet. defects / 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 %

4.2. GENeric SIMulator (GENSIM)

GENSIM (GENeric SIMulator) was derived from PSIM. It is a simplified didactical SPS model that was developed for the usage in academic environments for the training of students on aspects of software project management (Pfahl et al., 2001). Table 5 summarizes the goal definition.

Table 5. GENSIM goal definition template.

Role	Software Engineering Students
Scope	Single Project (phases design, implementation, test)
Dynamic Focus	Impact of project targets (time quality), product size, product complexity, manpower, skills, and quality assurance on project duration, project effort, and product quality
Purpose	Understanding and Training
Environment	University (e.g., Technical University Kaiserslautern, University of Oulu, University of Calgary, University of Reading, The Open

	University, National College of Ireland)

The GENSIM model is very flexible and can be used for various virtual experiments, e.g., investigating the impact of budget or time constraints, requirements volatility, skill, product size or complexity, quality goals, etc., on project duration, effort consumption, and product quality. Moreover, it is possible to investigate the impact of verification and validation techniques on the distribution of detected and undetected defects according to defect type. A fictitious example is discussed in the following paragraph.

Assume that – based on empirical evidence – two different types of design and code inspection techniques, i.e., INSP1 and INSP2, are known to have the following profiles with regards to detecting defects of severity class S1 and S2 in both design and code:

- INSP1 detects in design and code on average 80% of all S1 and 60% of all S2 defects;
- INSP2 detects in design and code on average 90% of all S1 and 50% of all S2 defects.

Assuming that a typical defect distribution in design and code is 20% defects of type S1 and 80% defects of type S2, then a comparison of applying INSP1 versus INSP2 during design and code results in a reduction of product quality (measured in terms of residual defect density) by 22% when using INSP2, while effort consumption and project duration slightly increase by 2% (cf. Figure 5). When looking at the defect distribution per severity class (cf. Figure 6) one can observe that INSP2 reduces the residual defects of class S1 by factor 2.7, while the number of class S2 residual defects increases by 26%. In other words, there are several types of trade-offs induced by applying verification techniques with varying defect detection profiles. One trade-off relates to the effectiveness in finding defects of a certain type throughout the project phases (and in combination with other techniques). Another trade-off relates to the overall performance with regards to product quality as compared to project performance, i.e. effort consumption and project duration.

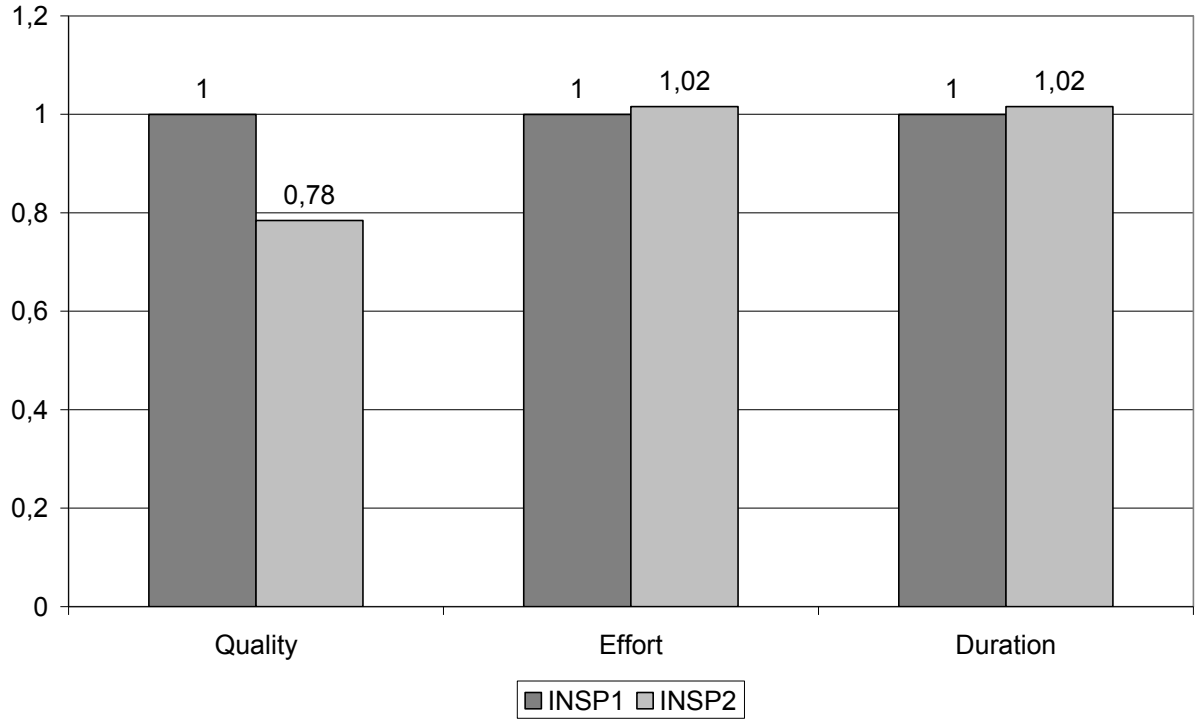

Figure 5. Comparison of product quality, effort consumption, and project duration (normalized data).

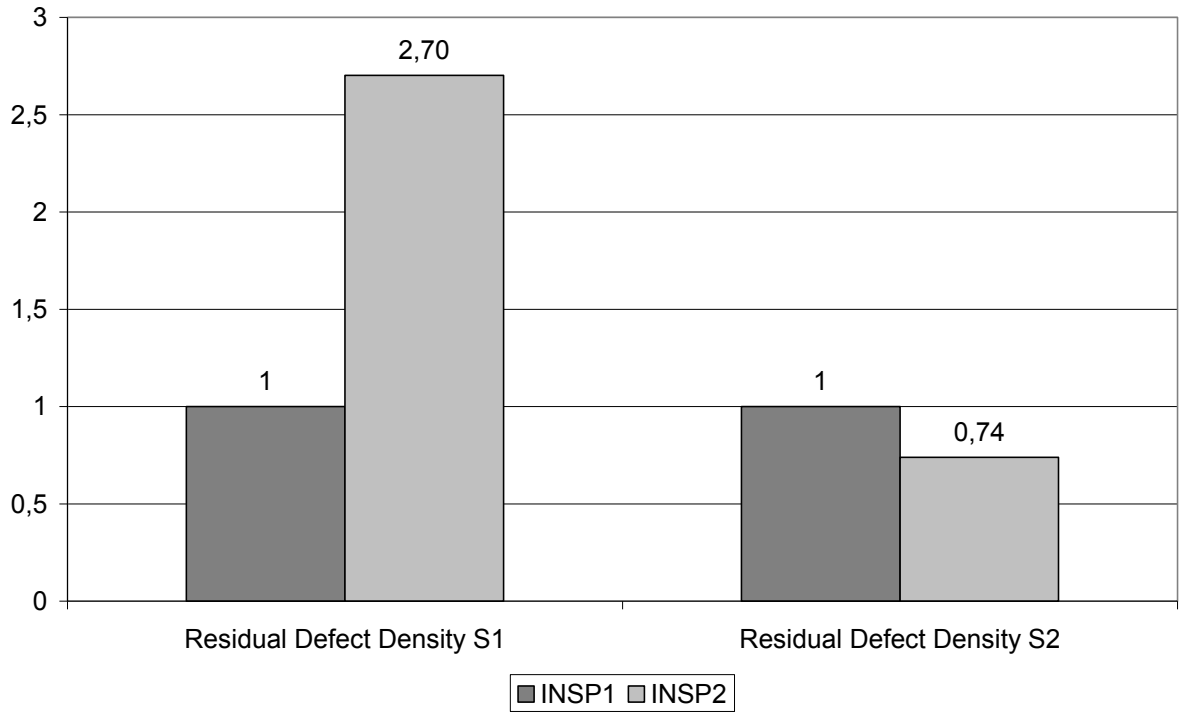

Figure 6. Distribution of residual defects per type (normalized data).

Depending on the availability of empirical data – which is needed to calibrate the SPS model – much more detailed analyses can be conducted. For example, the distribution of defect severity classes could be more refined and vary between work products, i.e. design and code documents. Moreover, more complex combinations of verification (and validation) techniques, each having specific defect detection profiles, could be analyzed.

Apart from serving as a means for student education and training, models such as GENSIM could serve as a means for preparing empirical experiments and focusing pilot projects. Under the assumption that the SPS is empirically valid, simulation experiments can, firstly, help to detect optimal combinations of techniques and, secondly, help explore the sensitivity of specific properties of a particular development activity with regards to defined project goals. This can help direct the development and empirical evaluation of new techniques towards those which are of (potentially) highest positive impact.

4.3. The SEV-RI Model

The SEV-RI model is a software requirements inspection process model that aims

at determining the defect detection efficiency of a requirements inspection process under varying contexts (Münch and Armbrust, 2003). It was developed within the scope of the German national research project SEV (Simulation-based Evaluation and Improvement of Software Development Processes) that aimed at developing a simulation environment for software development processes and experiments.

The model was developed according to a systematic method for modeling discrete-event simulation models of software development processes (Rus et al., 2003). The model is empirically based in results from an experiment conducted at the NASA/GSFC Software Engineering Laboratory (Basili et al., 1996) and two replications conducted at the Technical University Kaiserslautern (Ciolkowski et al., 1997). The experiment at NASA/GSFC SEL was conducted with professional developers whereas the replications were conducted with students. The purpose of the model is to support project managers and process engineers in adjusting the process such that a defined target defect detection rate is achieved. It is assumed that this is especially important during the project planning phase (in order to tailor the process to project goals and characteristics) and during improvement programs (in or to change the process and justify the change). The model allows for trade-off analysis between different context factors such as inspector experience and document size. Table 6 summarizes the goal definition.

Table 6. SEV-RI goal definition template.

Role	Project managers, process engineers
Scope	Inspection process in requirements definition phase (techniques: Perspective-based Reading technique and ad-hoc inspection); the model can be integrated in an overall lifecycle model
Dynamic Focus	Impact of document size, document complexity, experience, expertise, and type of inspection technique on defect detection efficiency
Purpose	Decision support for project planning and improvement
Environment	NASA/GSFC, Technical University Kaiserslautern

The SPS model itself was implemented as a discrete-event model using simulation modeling tool Extend (Krahl, 2000). The model does not model isolated defects, but relies on average values for defect detection because no complete data from all three experiment runs was available. An extended version of the model allows

for analyzing situations in which more than one document is inspected and several groups of reviewers are involved.

The model is validated with a separate set of data stemming from another requirements inspection experiment in a classroom setting. Although the contexts slightly differed (e.g., the document size was different), simulations result in document defect detection rates that are very close to the values observed in the real-life experiment.

The scientific focus of the model development was to better understand the use of quantitative empirical knowledge for SPS model development. During the development of SEV-RI, several lessons were learned about creating synergies between SPS modeling and real-life experimentation: As the SEV-RI model development was initially based on empirical knowledge gained from experiments that were not designed to support (discrete-event) simulation modeling, some information was missing or not fully suitable. Also, deriving valid SPS model equations from discrete data points is difficult because of incomplete knowledge about the interrelations between influence factors in different contexts. In the SEV project, a method was developed that describes how focused real-life experiments can be designed in a way that necessary data for SPS model-building can be collected efficiently.

4.4. The SimODC Model

With this model we focused on trade-offs for achievement of different dependability attributes. Usually dependability is considered a sub-set of quality, consisting of attributes such as reliability, safety, availability, security, and robustness. These are attributes related to the behavior of a system and are closely related to the perception of the user during system's operation. This constitutes the "external" perspective on dependability/quality. The "internal" perspective of dependability, that is a developer's view of software dependability (designed and predicted during development) focuses more on properties of software artifacts and their quality, expressed in defects. Defects that are introduced and not detected during development, if activated in the delivered product, might impact on the behavior of the system with respect to dependability properties. For that reason, these defects must be identified and removed. Different defects can impact the same, or different dependability attributes. Different methods for defect reduction might work better on some type

of defects than on others. For example, detecting and eliminating security issues does not necessarily improve reliability.

Figure 7 presents (for a hypothetical project) the profile of defects detected during development, during different phases. Defects are characterized by their severity and dependability attribute that they influence.

Elicitation and analysis of dependability attributes requirements for stakeholders' satisfaction is outside the scope of this work. Assuming that such an analysis had been performed, a manager has to achieve the required values for these attributes, dealing with schedule and cost constraints. By finding the right allocation of effort and resources to different activities in a project, in the specific context of an organization and project, the manager can reach the balance for achieving multiple goals. How to make these decisions is, however, difficult and relies to a great extend on personal experience. To support this decision-making task, we developed a process model and simulator that uses historical data from the organization under consideration, some of them provided by an Orthogonal Defect Classification (ODC) based defect data collection (Rus et al., 1999). Table 7 shows the characteristics of this model.

Table 7. SimODC goal definition template.

Role	Project managers, process engineers
Scope	Verification and validation activities throughout development life cycle
Dynamic Focus	Impact of effort and resource allocation for different verification and validation activities on dependability attributes
Purpose	Decision support for project planning, for achieving the desired balance between dependability attributes
Environment	Generic software development project

Using discrete event modeling of a software development process, we focused on the evolution of defects, i.e., their introduction, detection, and removal (Rus et al., 2002). This modeling approach allows different attributes to be attached to the entities modeled - the defects in our case. For example, each defect can be characterized by its type, severity, and effort to be detected or fixed. The values of these attributes are set or changed during the simulation, when a related event happens (e.g. when a defect is discovered the type attribute is set to the appropriate value). This model also captures the structure of the process, activities, resources, flows of artifacts and defects, parameters such as

resource allocation, duration, effort and cost for each activity, and relationships between these parameters. The values for the product and process parameters can be deterministic or follow probabilistic distributions.

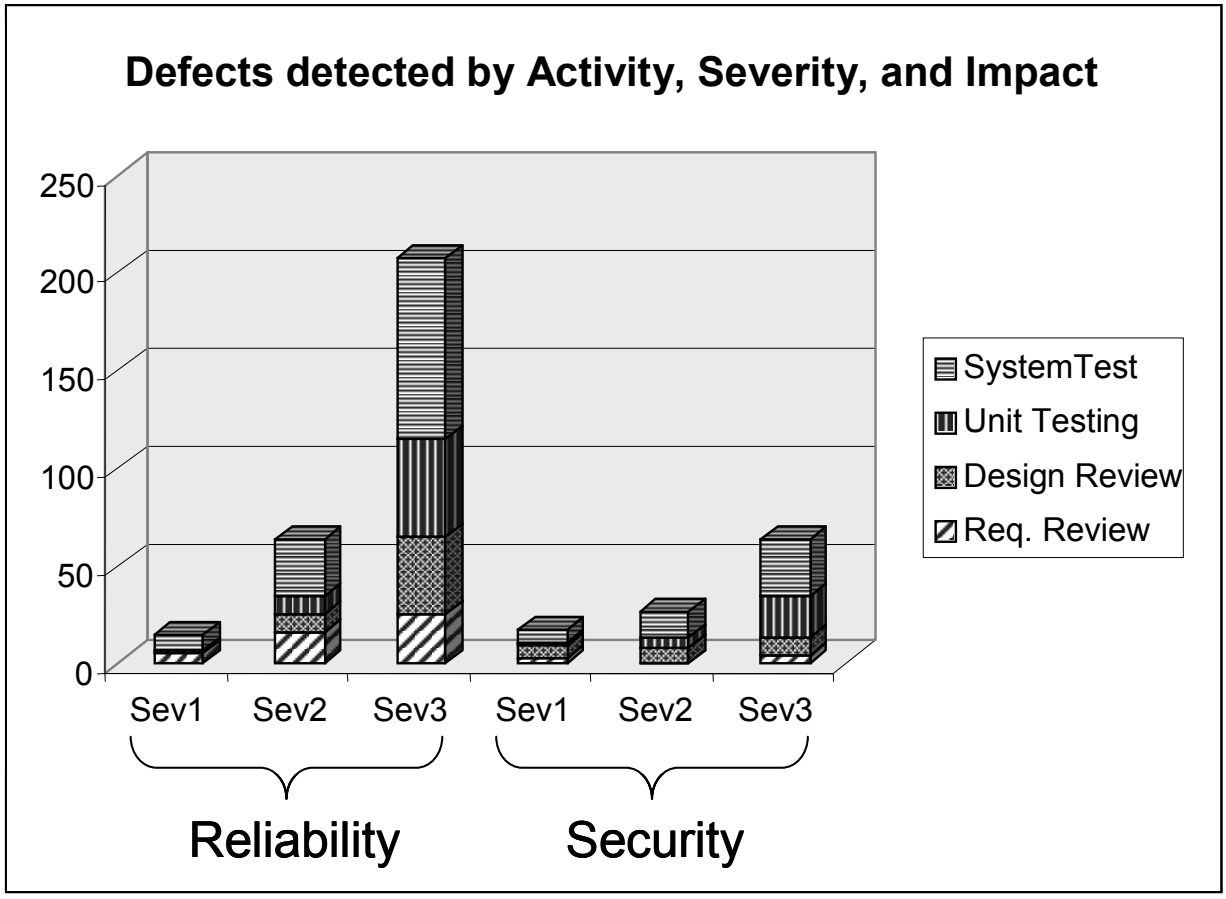

Figure 7. Profile of detected defects and their contribution to dependability properties

The simulator has to be first calibrated with values specific for an organization (such as expected productivity, quality of work expressed by defect generation, and effort for detecting and fixing a defect) and then executed with values specific for the project (such as size of the task and available resources). The outputs are cost, effort, schedule, and remaining defects, giving an indication about how dependable the software will be and how costly it is to reach that level of dependability.

By using a virtual lab approach and empirical studies (experimentation) for obtaining ODC-like data, as well as data regarding project parameters as

calibration and input parameters, this simulator will help answer questions such as: What trade-offs should be considered and what changes in the process should be made in order to efficiently achieve the desired dependability objectives? More specifically: Should more resources or time be allocated to design reviews, to code inspections, or to unit testing? Should developers be trained for using new reading or testing techniques? Should test automation be introduced or increased? Should developer's coding skills be improved, or new approaches be used so that fewer defects of a specific type would be generated? How would the dependability attributes be affected by these changes? How much each of these changes would cost, in terms of money and delays in product release? What changes would have the greatest impact on achieving the goal, or the best return on investment? What improvements are worth making?

5. Discussion

The case examples presented in this paper have illustrated how SPS models as the core elements of VSELs provide helpful support to managers in product-focused trade-off analysis and decision-making situations.

5.1. Efficiency

To be efficient, a VSEL needs to be more than an ad-hoc set of SPS models. It must provide a simulation environment comprising the following elements:

- A systematically developed, modular simulation-based evaluation platform that facilitates reuse and efficient application of SPS models.
- A method for preparing, conducting, packaging, and enriching the results of simulation-based studies.
- An integrated tool environment for the effective application of the method.

Such an VSEL efficiently supports project managers and planners, experimenters, process engineers (in the context of process improvement), and trainers (in the context of practice-oriented teaching). Using an efficient VSEL promises the following advantages in the context of product-focused trade-off analyses:

- Cost reduction by simulating software development processes and human behavior.
- Cost reduction by better selecting and focusing the scope of real-life experiments.

- Coupling with optimization methods promises results that could not be obtained by varying the impact factors in real or simulated experiments.
- The demonstration of the benefits of new methods in the context of an industrial development environment.
- Practice-oriented learning with respect to project planning and management can be performed in a scenario-oriented way.
- The simulation environment with the corresponding method can be integrated into process improvement programs.

5.2. Effectiveness

To be effective, the SPS models contained in a VSEL need to be sufficiently valid. Since software development is a people-based activity, which does not rely on physical laws, and thus cannot be treated mechanistically like a production process, SPS models representing realistic industrial processes will hardly ever achieve 100% predictive validity. Full predictive validity would require that the values of all model variables are predicted with very high accuracy for each point in time within the simulated time interval. Fortunately, such a high degree of validity is not always required. To be effective, it is sufficient that a SPS model achieves a degree of validity that is adequate with regards to the model purpose.

SPS models can have various purposes, i.e. not only prediction (in the sense of point estimates). Often, they are used to analyze effects of current processes or explore the effects of process changes. For these purposes, trends or the (qualitative) distinction of positive versus negative impacts might be sufficient. Also, it might not always be necessary to have measurement data for all parameters/variables at hand; to a certain extent, expert opinion can also be used to calibrate SPS models.

In order to make sure that the right purpose is identified during simulation modeling, and the resulting SPS model actually fulfils this purpose, a goal-oriented approach is recommended. IMMoS, which we already mentioned in the previous section, is an example of a goal-oriented SPS modeling approach (Pfahl and Ruhe, 2002). Inspired by goal-oriented measurement Basili and Weiss, 1984, Briand et al., 1996), IMMoS advocates the active involvement of process experts and prospective SPS model users. The involvement of end users assures that the SPS model focuses on the right problem(s), while the involvement of experts assures that the SPS model is built right, i.e., the actual processes are

modeled, not the official ones. In addition, IMMoS provides guidance on how to integrate static process modeling and measurement with simulation modeling. This helps to ground the SPS model on a solid empirical base.

6. Future Developments

As stated in the introduction, a comprehensive VSEL consists of a systematically developed set of reusable SPS modules, which can be composed and tailored. The long-term goal is to have customizable SPS modules that are sufficiently validated for various purposes in typical contexts and, when suitably combined, cover all key software processes. To achieve this is not a trivial task. Currently, several research groups worldwide are working on methods that aim at making the development of dependable SPS models faster and cheaper, e.g. by applying principles of agile development and process patterns (Angkasaputra and Pfahl, 2004).

An important next step would be the seamless integration of VSELs and traditional SELs, i.e., the creation of Advanced SELs (ASELs). ASELs provide an organizational and technological framework that facilitates and actually implements the synergetic integration of empirical-based and simulation-based experimentation (Münch et al., 2003). Much work is still needed to create ASELs. For example, the complex relationship between empirical studies and SPS is still relatively unexplored, even though first results exist. The vision of ASELs leads to several new research questions such as: How to effectively combine real-life and virtual (i.e., simulation-based) experiments? How to validate hypotheses for real experiments with the help of SPS? How to scale results from real-life experiments without significantly lowering validity? Industrial use of ASELs will yield further challenges, e.g., customer-oriented visualization of results, mapping of observed effects on business goals, and identifying market demands for simulation-based training.

7. Conclusions

Based on the presented case studies and subsequent discussion, we conclude that VSELs are practical means in support of product-focused trade-off analyses in the context of software development. Furthermore, there exist clear requirements that need to be fulfilled by a VSEL in order to become efficient in an industrial context. Finally, if we manage to seamlessly integrate VSELs with traditional

SELs, we expect significant synergies for product-focused trade-off analyses by systematically combining simulation-based with empirical-based experiments.

Acknowledgements

This work was supported in part by the German Federal Ministry of Education and Research (SEV Project) and the Stiftung Rheinland-Pfalz für Innovation (ProSim Project, no.: 559). The development of the SEV-RI model was mainly conducted by Ove Armbrust.

References

Abdel-Hamid, T. K. and Madnick, S. E. 1991, Software Project Dynamics – an Integrated Approach. Prentice-Hall.

Angkasaputra, N. and Pfahl, D. 2004. Making Software Process Simulation Modeling Agile and Pattern-based. Proc. 5th International Workshop on Software Process Simulation Modeling, Edinburgh, Scotland, 222-227.

Basili, V. R., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sørumgård, S., and Zelkowitz, M. V. 1996. The empirical investigation of perspective-based reading, Empirical Software Engineering 1(2): 133–164.

Basili, V. R., McGarry, F., Pajerski, R., and Zelkowitz, M. V. 2002. Lessons learned from 25 years of process improvement: The rise and fall of the NASA Software Engineering Laboratory. Proc. 24th International Conference on Software Engineering, Orlando, Florida, USA, 69-79.

Basili, V. R., Selby, R., and Hutchens, D. 1986. Experimentation in Software Engineering. IEEE Transactions on Software Engineering 12(7): 733-743.

Basili, V. R. and Weiss, D. M. 1984. A Methodology for Collecting Valid Software Engineering Data. Transactions on Software Engineering 10(6): 728-738.

Briand, L. C., Differding, C., and Rombach, D. 1996. Practical Guidelines for Measurement-Based Process Improvement. Software Process Improvement and Practice 2: 253-280.

Cartwright, M. and Shepperd, M. 1999. 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.

Ciolkowski, M., Differding, C., Laitenberger, O., and Münch, J. 1997. Empirical investigation of perspective-based reading: A replicated experiment. Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany, Technical Report 048.97/E.

Christie, A. M. 1999a. Simulation: An Enabling Technology in Software Engineering. CROSSTALK – The Journal of Defense Software Engineering, 2-7.

Christie, A. M. 1999b. Simulation in support of CMM-based process improvement. Journal of Systems and Software 46(2/3): 107-112.

Drappa, A. and Ludewig, J. 1999. Quantitative modeling for the interactive simulation of software projects. Journal of Systems and Software 46(2/3): 113-122.

Forrester, J. W. 1961. Industrial Dynamics. Productivity Press, Cambridge.

Höst, M., Regnell, B., Dag, J., Nedstam, J., and Nyberg, C. 2001. Exploring Bootlenecks in Market-Driven Requirements Management Processes with Discrete Event Simulation. Journal of Systems and Software 59(3): 323-332.

Houston, D. X., Mackulak, G. T., and Collofello, J. S. 2001. Stochastic simulation of risk factor potential effects for software development risk management. Journal of Systems and Software 59(3): 247-257.

Jensen, C. and Scacchi, W. 2003. Simulating an Automated Approach to Discovery and Modeling of Open Source Software Development Processes. Proc. 4th Process Simulation Modeling Workshop, Portland, USA.

Judd, C., Smith, E. R., and Kidder, L. 1991. Research Methods in Social Relations. Harcourt Brace Jovanovich College Publishers, sixth edition.

Kellner, M. I., Madachy, R. J., and Raffo, D. M. 1999. Software process simulation modeling: Why? What? How? Journal of Systems and Software 46(2/3): 91-105.

Krahl, D. 2000. The Extend simulation environment. Proc. 2000 Winter Simulation Conference, Orlando, Florida, USA, IEEE Press, 280–289.

Lee, B. and Miller, J. 2004. Multi-Project Management in Software Engineering Using Simulation Modeling. Software Quality Journal 12: 59-82.

Lin, C. Y., Abdel-Hamid, T. K., and Sherif, J. 1997. Software-Engineering Process Simulation Model (SEPS). Journal of Systems and Software 38: 263-277.

Madachy, R. J. 1996. System Dynamics Modeling of an Inspection-Based Process. Proc. 18th International Conference on Software Engineering, Berlin, Germany, IEEE Computer Society Press, 376-386.

Madachy, R. J. and Tarbet, D. 2000. Case Studies in Software Process Modeling with System Dynamics. Software Process Improvement and Practice 5: 133-146.

Mišic, V. B., Gevaert, H., and Rennie, M. 2004. Extreme Dynamics: Towards a System Dynamics Model of the Extreme Programming Software Development Process. Proc. 5th International Workshop on Software Process Simulation Modeling, Edinburgh, Scotland, 237-242.

Münch, J. and Armbrust, O. 2003. Using Empirical Knowledge from Replicated Experiments for Software Process Simulation: A Practical Example. Proceedings of 2nd ACM-IEEE International Symposium on Empirical Software Engineering, Rome, Italy, 18-27.

Münch, J., Rombach, D., and Rus, I. 2003. Creating an Advanced Software Engineering Laboratory by Combining Empirical Studies with Process Simulation. Proc. 4th International Workshop on Software Process Simulation Modeling, Portland, Oregon, USA.

Neu, N., Hanne, T., Münch, J., Nickel, S., and Wirsen, A. 2002. Simulation-Based Risk Reduction for Planning Inspections. Proc. 4th International Conference on Product Focused Software Process Improvement. Lecture Notes in Computer Science 2559, Springer, 78-93.

Pfahl, D. and Birk, A. 2000. Using Simulation to Visualise and Analyse Product-Process Dependencies in Software Development Projects. Proc. 2nd International Conference on Product Focused Software Process Improvement, Oulu, Finland, 88-102.

Pfahl, D., Klemm, M., and Ruhe, G. 2001. A CBT module with integrated simulation component for software project management education and training. Journal of Systems and Software 59(3): 283-298.

Pfahl, D. and Lebsanft, K. 2000a. Using Simulation to Analyse the Impact of Software Requirement Volatility on Project Performance. Information and Software Technology 42(14): 1001-1008.

Pfahl, D. and Lebsanft, K. 2000b. 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(4): 487-510.

Pfahl, D. and Ruhe, G. 2002. IMMoS - A Methodology for Integrated Measurement, Modelling, and Simulation. Software Process Improvement and Practice 7: 189-210.

Powell, A., Mander, K., and Brown, D. 1999. Strategies for lifecycle concurrency and iteration: A system dynamics approach. Journal of Systems and Software 46(2/3): 151-162.

Raffo, D. M., Kaltio, T., Partridge, D., Phalp, K., and Ramil, J. F. 1999. Empirical Studies Applied to Software Process Models. Empirical Software Engineering 4(4): 353-369.

Raffo, D. M., Nayak, U., Setamanit, S., Sullivan, P., and Wakeland, W. 2004. Using Software Process Simulation to Assess the Impact of IV&V Activities. Proc. 5th International Workshop on Software Process Simulation Modeling, Edinburgh, Scotland, 197-205.

Raffo, D. M., Setamanit, S., and Wakeland, W. 2003. Towards a Software Process Simulation Model of Globally Distributed Software Development Projects. Proc. 4th Process Simulation Modeling Workshop, Portland, Oregon, USA.

Raffo, D. M., Vandeville, J. V., and Martin, R. H. 1999. Software process simulation to achieve higher CMM levels. Journal of Systems and Software 46(2/3): 163-172.

Roehling, S. T., Collofello, J. S., Hermann, B. G., and Smith-Daniels, D. E. 2000. System Dynamics Modeling Applied to Software Outsourcing Decision Support. Software Process Improvement and Practice 5: 169-182.

Rombach, H. D. 1999. Experimentation – Engine for Applied Research and Technology in Software Engineering. Proc. NASA's 24th Annual Software Engineering Workshop, Software Engineering Laboratory, Greenbelt, Maryland, USA.

Rombach, H. D., Basili, V. R., and Selby, R. W. 1993. Experimental Software Engineering Issues: Critical Assessment and Future Directions, Lecture Notes in Computer Science, Springer.

Ruiz, M., Ramos, I., and Toro, M. 2004. Using Dynamic Modeling and Simulation to Improve the COTS Software Process. Proc. 5th International Conference on Product Focused Software Process Improvement, Kyoto, Japan, 568-581.

Rus, I. 2002. Combining Process Simulation and Orthogonal Defect Classification for Improving Software Dependability. Proc. 13th International Symposium on Software Reliability Engineering, Annapolis, USA.

Rus, I., Biffl, S., and Halling, M. 2002. Systematically Combining Process Simulation and Empirical Data in Support of Decision Analysis in Software Development. Proc. 1st International Workshop on Software Engineering Decision Support, Ischia, Italy.

Rus, I., Collofello, J. S., and Lakey, P. 1999. Software process simulation for reliability management. Journal of Systems and Software 46(2/3): 173-182.

Rus, I., Neu, H., and Münch, J. 2003. A Systematic Methodology for Developing Discrete Event Simulation Models of Software Development Processes. Proc. 4th International Workshop on Software Process Simulation and Modeling, Portland, Oregon, USA.

Stallinger, F. and Grünbacher, P. 2001. System dynamics modelling and simulation of collaborative requirements engineering. Journal of Systems and Software 59: 311-321.

Ventana Systems. 2004. http://www.vensim.com (last visited on 6 Sep 2004)

Waeselynck, H. and Pfahl, D. 1994. System Dynamics Applied to the Modelling of Software Projects. Software Concepts and Tools 15(4): 162-176.

Wernick, P. and Hall, T. 2004. A Policy Investigation Model for Long-term Software Evolution Processes. Proc. 5th International Workshop on Software Process Simulation Modeling, Edinburgh, Scotland, 149-158.

Williford, J. and Chang, A. 1999. Modelling the FedEx IT Division: A System Dynamics Approach to Strategic IT Planning. Journal of Systems and Software 46(2/3): 203-211.

Biographies

Dr. Jürgen Münch is department head for quality and process engineering at the Fraunhofer Institute for Experimental Software Engineering (IESE), Kaiserslautern. Before that, Dr. Münch was an executive board member of the temporary research institute SFB 501 "Development of Large Systems with Generic Methods" funded by the German Research Foundation (DFG). Dr. Münch received his PhD degree (Dr. rer. nat.) in Computer Science from the University of Kaiserslautern, Germany. Dr. Münch's research interests in software engineering include: (1) modeling and measurement of software processes and products, (2) software quality assurance and control, (3) technology evaluation through experimental means and simulation, (4) generic methods for the development of large systems, (5) technology transfer methods.

Dr. Dietmar Pfahl is department head with the Fraunhofer Institute for Experimental Software Engineering (IESE) in Kaiserslautern, Germany. He has more than 15 years of experience in conducting and leading national and international research and transfer projects with software industry, including organizations such as Bosch, DaimlerChrysler, Dräger, Ericsson, and Siemens. Dr. Pfahl received his PhD degree (Dr. rer. nat.) in Computer Science from the University of Kaiserslautern. He has more than 50 refereed publications. His current research interests include quantitative software project management, software process analysis and improvement, and simulation-based learning and decision support.

Dr. Ioana Rus is a scientist at the Fraunhofer Center for Empirical Software Engineering Maryland. She has experience in research, teaching, and software development. Her current research interests include software process modeling and simulation, dependability engineering, experience and knowledge management, process improvement, measurement and empirical studies in software development and technology transfer. She has a Ph.D. in Computer Science and Engineering and is a member of the IEEE Computer Society.