Dissertationen

Oliver Laitenberger: **Cost-effective Detection of Software Defects through Perspective-based Inspections**

Promotion: Universität Kaiserslautern
Erstgutachter: Prof. Dr. H. Dieter Rombach (Universtität Kaiserslautern)
Zweitgutachter: Prof. Dr. Victor R. Basili, University of Maryland, USA
Datum der Prüfung: 16.05.2000
Veröffentlichung: Stuttgart: Fraunhofer IRB Verlag, 2000 (PhD Theses in Experimental Software Engineering 1). Zugl.: Kaiserslautern, Univ., Diss., 2000; ISBN 3-8167-5583-6

Kurzfassung:

Software organizations must deliver high quality products on time and within budget to remain competitive in the marketplace. However, the reliable and predictable development of high quality software continues to be a major problem, largely due to the inadequate and late removal of defects.

One of the proposed solutions for early detection and removal of defects is software inspection. However, despite the large body of inspection experience accumulated over the last 20 years, there is still a need for further work on the question of how managers can organize a software inspection with respect to the examined documentation and, once it has been organized, how developers can perform the defect detection activity in a systematic manner.

This thesis presents a new approach to inspection organization. It argues for organizing inspections around logical entities from the software architecture rather than around particular (types of) documents. Although this approach, dubbed architecture-centric inspection organization, is generally applicable, we consider it particularly valuable for inspections in object-oriented development projects.

To provide systematic support for detecting defects in the documentation of any logical entity, this thesis introduces the Perspective-based Reading (PBR) technique. The basic idea underlying PBR is to focus each inspector on a specific viewpoint and to provide active guidelines for scrutinizing the documentation of one or several logical entities for defects. To elaborate this idea into a generally applicable technique, the thesis provides an in-depth description of PBR. Moreover, it explains how to tailor PBR to various inspection situations.

To quantify the potential benefits of PBR in conjunction with the architecture-centric inspection organization, this thesis describes three empirical studies. The primary goal of these studies was the evaluation of PBR in comparison with more traditional reading techniques, that is, adhoc or checklist-based reading. The study results suggest that PBR increases the defect detection effectiveness as well as the cost-effectiveness of inspections.

By providing a detailed explanation of PBR, we expect that researchers will be able to extend the scope of the technique, perform replications of our empirical studies, and set up new ones. The reported experiences gathered in the design, execution, and analysis of our studies may be helpful for these purposes. However, we strongly believe that the lessons learned in those studies are beneficial for the empirical software engineering field in general. Practitioners not only receive concrete advice about how to apply the architecture-centric approach for inspection organization and how to utilize PBR for their specific inspection implementation, but also how to evaluate software inspections in their projects and their organizations.

Carsten Tautz: **Customizing Software Engineering Experience Management Systems to Organizational Needs**

Promotion: Universität Kaiserslautern
Erstgutachter: Prof. Dr. H. Dieter Rombach (Universität Kaiserslautern)
Zweitgutachter: Prof. Dr. Michael M. Richter
Datum der Prüfung: 11.09.2000
Veröffentlichung: Stuttgart: Fraunhofer IRB Verlag, 2001(PhD Theses in Experimental Software Engineering 4). Zugl.: Kaiserslautern, Univ., Diss., 2000; ISBN 3-8167-5881-9

Kurzfassung:

The increasing internationalization of world trade intensifies the global competition forcing organizations to accelerate their technology changes and product innovations. Software development organizations are not exempted from this trend as software plays an increasingly important role for the value of products and services. The systematic acquisition, utilization, and maintenance of software engineering experience accelerates goal-oriented changes in software development practices. Thus, it has become a major competitive advantage.

Explicit software engineering experience is stored in a repository called the experience base. This dissertation presents a methodology based on the principle "learning from examples" for designing and implementing an experience base that contains all kinds of experience. The methodology consists of:

- A new knowledge representation formalism for experience base schemas. The formalism fulfills many practical needs (e.g., rationalized schemas and support for uncertain, imprecise, incomplete, and context-dependent information) for the first time. As such, it uses many features typically associated with case-based reasoning systems as well as ideas from other areas such as database systems.

- A generic architecture for the comprehensive tool support of an experience base. The architecture considers many practical needs such as the integration of already existing tools and learning in distributed environments.

- A generic task framework. The framework is the most comprehensive description for incremental, continuous learning (of all kinds of software engineering experience) based on a repository to date. Its description is based on the knowledge-level approach and, hence, implementation-independent.

- A systematic, goal-oriented method for setting up an experience base. The method describes how to engineer experience base schemas as well as how to tailor the generic architecture and instantiate the task framework to organization- specific needs. It is unique in the sense that its application does not only provide a schema, but also matching methods for populating (and updating) the experience base on the basis of the developed schema.

Systems designed using this methodology will provide faster and more effective access to software engineering experience available in an organization compared to existing solutions (usually a combination of "asking colleagues" and consulting lists and databases accessible over the intranet).

The methodology is complemented by a goal-oriented method for improving the technical infrastructure of a running experience base from a user's point of view.

The components of the methodology have been validated in several industrial- strength projects including Fraunhofer IESE's Corporate Information Network, which is presented in detail. A prototype for the tool architecture has been built using a commercially available case-based reasoning system as its modeling tool and retrieval engine.

Andreas Birk: A Knowledge Management Infrastructure for Systematic Improvement in Software Engineering

Promotion: Universität Kaiserslautern
Erstgutachter: Prof. Dr. H. Dieter Rombach (Universität Kaiserslautern)
Zweitgutachter: Prof. Dr. Werner Mellis (Universität Köln)
Datum der Prüfung: 15.09.2000
Veröffentlichung: Stuttgart: Fraunhofer IRB Verlag, 2001 (PhD Theses in Experimental Software Engineering 3). Zugl.: Kaiserslautern, Univ., Diss., 2000; ISBN 3-8167-5880-0

Kurzfassung:

Present improvement approaches in software engineering offer very little guidance for informed decision making regarding the identification and selection of concrete improvement actions. Decisions about improvement actions are often either based on the personal experience of software professionals, who are likely to be biased, or triggered by general trends in the software domain (so-called silver bullet solutions or technology hypes). This limits the effectiveness of improvement programmes and even can make them fail.

The objective of this thesis is to develop a knowledge management infrastructure through which experience about improvement actions can be made explicit, accumulated, and offered for informed decision making. The infrastructure shall help establishing systematic improvement processes and overcoming ad hoc practices and decisions.

Based on the Experience Factory / Quality Improvement Paradigm (EF/QIP) [BCR94a], a knowledge management framework is to be set up for managing experience about software engineering technologies and about the effects and context dependencies of technology application. This shall provide the empirically validated information needed for supporting the identification and selection of improvementactions for given improvement goals and context characteristics of a software organisation.

Contributions

The EF/QIP makes software development and software engineering improvement more systematic by supporting the learning and reuse of relevant experience. This thesis refines and extends the EF/QIP for the specific case of technology-related experience reuse in systematic improvement. In particular, the thesis provides:

- An extended approach to software process modelling that includes an explicit concept of software engineering technologies, relates the process with the developed products, and describes the impact that the context of software development has on technologies, processes, and products.
- A knowledge representation schema for modelling technology experience packages (TEPs) and for organising them in the form of technology experience bases (TEBs). TEPs represent the core knowledge items of the approach. They store experience relevant for the application of software engineering technologies. TEBs are knowledge repositories that contain and organise collections of technology experience packages.
- Knowledge management processes for the acquisition, deployment, and evolution of TEPs:
- A knowledge acquisition method collects and models distributed experience about the application of technologies.
- A systematic technology selection method deploys TEBs (and the contained TEPs) for supporting the planning of software projects and improvement programmes.
- A TEP evolution method analyses software projects in which a technology of interest is applied and suggests modifications or refinements of the technology's TEPs.

These contributions establish a refinement of the EF/QIP for the specific case of experience about the effects and context dependencies of software engineering technologies. In addition, concrete patterns of the effects and context dependencies of software engineering technologies have been identified. These patterns can be used for supporting decision making for the identification of improvement actions.

Scope and Emphasis

The required first steps for setting up the knowledge management infrastructure in a software organisation are the installation of a TEB and its population with organisation-specific TEPs. In order to support this bootstrapping of TEBs, the thesis places particular emphasis on the development of the knowledge representation schema and the knowledge acquisition method.

Application and validation

The approach has been applied several times in the context of industrial software projects. The experience from these applications support that the approach is well applicable in industrial settings and that it effectively contributes to the enhancement of systematic improvement practices.

Christian Bunse: **Pattern-Based Refinement and Translation of Object-Oriented Models to Code**

Promotion: Universität Kaiserslautern
Erstgutachter: Prof. Dr. H. Dieter Rombach (Universität Kaiserslautern)
Zweitgutachter: Prof. Dr. Colin Atkinson (Universität Kaiserslautern)
Datum der Prüfung: 15.09.2000
Veröffentlichung: Stuttgart: Fraunhofer IRB Verlag, 2001 (PhD Theses in Experimental Software Engineering 2). Zugl.: Kaiserslautern, Univ., Diss., 2000; ISBN 3-8167-5613-1

Kurzfassung:

This thesis addresses the problem of systematically implementing object-oriented models by identifying the basic nature of the transformation activities involved in the implementation of software systems (i.e., refinement and translation), and defining a method which separates and explicitly distinguishes between refinement and translation, known as SORT. Refinement is viewed as the process of moving from a model presented in some language (e.g. UML) to a more detailed model presented in the same language, while translation is viewed as the process of moving from a description presented in some language (e.g., UML) to a description at the same level of detail presented in another language (e.g., C++). To help developers move from design to code, the SORT method, defined in this thesis, provides a set of patterns (both refinement and translation patterns) organized according to a multi-dimension classification scheme.

In general, the SORT method defines a process for the stepwise refinement of models to a level of abstraction, known as the implementation level, from which models can be translated to source-code. The method also supports continuous quality assurance by means of a systematic visualization and verification technique, as well as by a tool developed in the context of the thesis. In specific projects, developers choose the refinement patterns that best fit their needs (e.g., non-functional requirements) and apply them repeatedly to their models until the models are at the implementation level. Translation patterns are then applied to translate the refined models to code. To exactly define when refinement should end and translation can start, this thesis introduces the concept of the implementation set. An implementation set is composed of modeling constructs which are semantically close to code, and which can thus be directly mapped onto code. In other words, the implementation set defines when a model provides a sufficient level of detail for implementation.

As experienced developers are well aware, the underlying concepts of object-oriented programming are basically the same whatever language or implementation vehicle is used to apply them. Therefore, this thesis defines a specific implementation set, known as the Normal Object Form (NOF), aimed at exploiting that commonality to provide a language independent subset of object-oriented constructs. NOF constructs are semantically close to source-code but are independent of a specific language, thus allowing the translation of NOF models into various target languages.

The effectiveness of the SORT approach has been tested in the context of this thesis by a controlled student experiment. The experiment results show that SORT technology is beneficial and has a positive impact on the traceability, understandability, and verifiability of a design due to the establishment of precise links between models and source-code.

In summary, the SORT method addresses a weak link in modern object-oriented development methods, namely the problem of moving from design to code in a systematic way. In this sense, SORT follows the tradition of one of the most systematic processes defined to date: Cleanroom Software Engineering.

Erik Kamsties: Surfacing Ambiguity in Natural Language Requirements

Promotion: Universität Kaiserslautern
Erstgutachter: Prof. Dr. H. Dieter Rombach (Universität Kaiserslautern)
Zweitgutachter: Prof. Dr. D. M. Berry (University of Waterloo, Kanada)
Datum der Prüfung: 16.03.2001
Veröffentlichung: Stuttgart: Fraunhofer IRB Verlag, 2001 (PhD Theses in Experimental Software Engineering 5). Zugl.: Kaiserslautern, Univ., Diss., 2001; ISBN 3-8167-5903-3

Kurzfassung:

In industrial requirements engineering (RE), natural language is the most frequently used representation to state the requirements that are to be met by information technology products or services. Natural language is universal, flexible, and wide-spread, but unfortunately also inherently ambiguous. Even worse, often neither customers nor software developers recognize an ambiguity, and each derives an interpretation that differs from that of others without noticing this difference. Consequently, software developers design and implement a system that does not behave as intended by the customers. Semi-formal and formal representation techniques, such as UML or SCR, have been proposed to overcome the weaknesses of natural language. However, they only shift the problem; an ambiguity simply becomes an unambiguously wrong specification statement, which must be detected by the customers in reviews or simulations of the specification.

This thesis addresses the problem of detecting ambiguities in natural language requirements. We distinguish between linguistic and RE-specific ambiguities. Linguistic ambiguities are the commonly known ambiguities of natural language, such as ambiguous pronoun references. RE-specific ambiguities arise from the RE context, which comprises the application, system, and development domain. The requirement "The system injects coolant to [...] when the pressure falls below a 'low' threshold" is ambiguous because of the system domain, i.e., it is not clear whether injecting coolant is a discrete action or continuous activity. An experiment was performed to understand the problems that the different types of ambiguity cause during the formalization of informal requirements.

This thesis presents the ADTD approach, which allows developing techniques for detecting ambiguities in natural language requirements on the basis of existing and industrially proven techniques, namely checklists, scenario-based reading, and agendas. A technique for detecting ambiguities must be tailored to a particular RE context to be effective, because RE-specific ambiguities depend heavily on the context in which an RE process takes place. For this purpose, the ADTD approach provides heuristics to investigate metamodels, which capture information about an RE context. These heuristics help identify the RE-specific types of ambiguities that are typical for the particular RE context.

A set of ambiguity detection techniques for the domain of embedded systems was developed using the ADTD approach to show its feasibility. To quantify the benefits of the ADTD approach, the resulting techniques were empirically validated in five experiments. The goals were (1) to validate that the investigation of metamodels pays off in terms of higher efficiency of the resulting technique and (2) to show that our ambiguity detection techniques are superior to existing ones.

From a practitioner's perspective, a set of validated techniques for detecting ambiguities is provided, which allow improving current RE processes of different maturity. The ADTD approach is available for developing additional techniques if the provided ones do not fit the situation at hand. From a researcher's perspective, new insights are provided on the problems that defects, and ambiguities in particular, cause when semi-formal or formal specifications are developed from an informal statement of requirements. Also, the thesis provides a comprehensive understanding of the various facets of ambiguity in RE. This understanding clarifies the currently ambiguous use of the term in the literature.

Christiane Differding: Adaptive Measurement Plans for Software Development

Promotion: Universität Kaiserslautern
Erstgutachter: Prof. Dr. H. Dieter Rombach (Universität Kaiserslautern)
Zweitgutachter: Prof. Dr. Michael M. Richter (Universität Kaiserslautern)
Datum der Prüfung: 20.04.2001
Veröffentlichung: Stuttgart: Fraunhofer IRB Verlag, 2001 (PhD Theses in Experimental Software Engineering 6). Zugl.: Kaiserslautern, Univ., Diss., 2001; ISBN 3-8167-5908-4

Kurzfassung:

Goal-oriented measurement is a key technology for planning, tracking, and controlling software development products and processes, and the Goal/Question/Metric paradigm (GQM) is its most promising approach. Measurement principles have been understood and documented quite well through experience gathered in many measurement programs. However, detailed descriptions of systematic techniques for measurement planning are still missing. Most techniques are applied rather intuitively, which implies that the experience of the measurement planner plays an important role. Several critiques have been formulated about the GQM approach, e.g., difficulty of goal formulation, lack of consistency of the resulting set of measures, lack of models, lack of consideration of the software development process being measured, and difficulties with modification and reuse of measurement plans. Responding to these critiques, models have been integrated into GQM plans, templates and guidelines for goal-setting have been defined, and a reuse approach for measurement plans based on case-based reasoning has been developed. Nevertheless, the difficulty of goal-setting still exists, the problem of inconsistent sets of measures has not yet been solved, the software development process is not sufficiently taken into account during measurement planning, and modification and reuse of measurement plans are not yet sufficiently supported.

The goal of this thesis is to improve the GQM approach with a focus on modification and reuse. It investigates the following research hypothesis: Measurement plans that are consistent with project information can be developed with lower effort and fewer defects using the proposed reuse-oriented process than with an intuitive process based on the literature.

The approach to measurement planning presented in this thesis is based on reuse of project information, i.e., software development process definitions and quality models, during the development of measurement plans. The relationships that are captured between measurement plans and project information are used to modify a measurement plan based on typical modification requests originating from project information. The approach is developed in an iterative way and consists of several parts: a formal conceptual model for measurement plans integrated with software development process definitions and quality models, a measurement plan development process with a supporting tool, a measurement plan modification process with a supporting tool, and results from case studies and a controlled experiment that validate the approach. First ideas for a conceptual model for reuse and a reuse process are proposed.

Isabella Wieczorek: **Improved Software Cost Estimation - A Robust and Interpretable Modeling Method and a Comprehensive Investigation**

Promotion: Universität Kaiserslautern
Erstgutachter: Prof. Dr. H. Dieter Rombach (Universität Kaiserslautern)
Zweitgutachter: Dr. Lionel Briand (Carleton University, Kanada)
Datum der Prüfung: 11.06.2001
Veröffentlichung: Stuttgart: Fraunhofer IRB Verlag, 2001 (PhD Theses in Experimental Software Engineering 7). Zugl.: Kaiserslautern, Univ., Diss., 2001; ISBN 3-8167-6033-3

Kurzfassung:

Delivering software on time, within budget, and to an agreed level of quality is crucial for many organizations. To ensure their business reputation and competitiveness organizations have recognized the importance of software cost estimation. However, it is not well understood in practice how to obtain and interpret software cost predictions. This is because the estimation of cost for software projects leads to many practical measurement and modeling difficulties.

These difficulties are not satisfactory addressed by common estimation methods. Classical statistical methods have limited model building capabilities regarding software engineering data. Other methods produce models with low interpretability and limitations in capturing uncertainty. This restricts the acceptability and usefulness of estimation models.

To overcome these limitations, there is a need to derive methods that make less assumptions about the underlying data sets and whose models are easy to interpret. This thesis proposes an enhancement of the method Optimized Set Reduction (OSR) to solve estimation problems. This method is based on machine learning and robust statistics. OSR exploits the rigor of statistical modeling and still generates easy-to-interpret, rule-based models.

To evaluate the benefits of OSR, this thesis describes a subjective assessment and a comprehensive accuracy evaluation. OSR and commonly used cost estimation methods are applied to large software cost data sets from different application domains. The data allow for a comparison of estimation methods across organizations, as well as within one company.

Local, company-specific data are widely believed to provide a better basis for accurate estimates. On the other hand, multi-organizational databases provide an opportunity for fast data accumulation and shared information benefits. Therefore, this thesis also trades off the potential advantage and drawbacks of using local data as compared to multi-organizational databases.

The results from the comparative evaluation suggest that OSR performs relatively accurate and is more robust to outlying predictions than other methods considered. In the context of missing data, OSR optimally utilizes the available data and performs better than comparable methods.

The results investigating the feasibility of multi-organizational data sets suggest that to really benefit from collecting company-specific cost data, one should focus on collecting the important factors in an organization through a tailored measurement program. In case few local data are available, common data repositories across companies within homogenous application domains could be considered as a beneficial alternative.

Beside prediction accuracy, a broader range of issues should be considered when selecting an estimation method. OSR may be favored as it has the potential in providing easy-to-interpret predictive models. These allow for a better understanding of the factors impacting software costs and the uncertainties associated with software projects.

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

Promotion: Universität Kaiserslautern
Erstgutachter: Prof. Dr. H. Dieter Rombach (Universität Kaiserslautern)
Zweitgutachter: Prof. Dr. D. Ehrenberg (Universität Leipzig)
Datum der Prüfung: 01.10.2001
Veröffentlichung: Stuttgart: Fraunhofer IRB Verlag, 2001 (PhD Theses in Experimental Software Engineering 8). Zugl.: Kaiserslautern, Univ., Diss., 2001; ISBN 3-8167-6066-X

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

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

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

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

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