Track Coordinators: Gregor Engels, Wilhelm Schäfer
University of Paderborn, Dept. of Computer Science, D-33095 Paderborn, Germany
{engels|wilhelm}@upb.de

Track Introduction

Software Engineering is still on the way to become a true engineering discipline with firm and well-understood foundations. During the last two decades, researchers as well as practitioners worked hard to identify and to develop the right techniques for effective and successful software engineering processes. Due to several reasons, as, e.g., educational background or time and budget resource constraints, academic researchers on the one hand and industrial practitioners on the other hand followed different paths in developing these software engineering techniques.

The objective of the CHASE track is to contrast an industrial perspective with an academic perspective, in order to identify the main achievements of the past, i.e., the state-of-the-art, and to identify the main challenges for the future. This will be the base for drawing a joint road map for future work as result of the CHASE track.

The presentations and discussions focus on two important topics within software engineering, i.e., Process-centered Software Engineering Environments (PSEE) and the area of specification and modeling of software systems. Four distinguished and well-known persons have been invited to present their perspectives. On the topic of PSEEs, Bob Balzer and Volker Gruhn, both with an academic as well as industrial background, present their perspectives. On the topic of specification and modeling, Manfred Broy presents his academic, scientifically based perspective, while Bran Selic reports about an industrial perspective.

The final session in the CHASE track will report about a project which approaches the issue of achievements in particular on a broader scale than the two previous sessions. The IMPACT project is identifying major results in software engineering research which had a broad impact on industrial development processes. It will provide concise summaries of these results and describe the particular improvements which were based on these results. The first areas identified by the IMPACT project will be described in more detail in the last CHASE session.

Process-Centered Software Engineering Environments: Academic and Industrial Perspectives

Robert Balzer¹, Volker Gruhn²
¹ Teknowledge, Marina del Rey, CA 90292, USA
bbalzer@teknowledge.com
² Software Technology, University of Dortmund, D-44221 Dortmund, Germany
Volker.Gruhn@uni-dortmund.de

Abstract

Software engineering environments have a history of about two decades. Early environments provided support for small fragments of the software process (usually focusing on programming-in-the-small). Then there was a trend towards support for more complete software processes (from early phases like requirements analysis and design down to testing and configuration management). Ten years ago the notion of process-centered software engineering environments initiated a new field in software engineering: software process research. The key idea was to use a model of a software process as input parameter for a software engineering environment. The environment was supposed to “behave” in accordance to the process model. Some aspects of this vision became true, others turned out to be of little practicability.

Keywords
Software process, process-centered software engineering environment

1. Introduction

Software engineering environments which support exactly one kind of software process (and which perhaps even prescribe the use of a particular set of tools) have a few important advantages:
• they can enforce a high level of consistency between documents produced,
• they can guide software developers through the supported process,
• they always know in detail the state of the software process, and
• they can automate parts of the process (e.g. in configuration assembly, testing).

But there is also an enormous disadvantage: hardly any software developer wants to use highly integrated and process enforcing environments. Most developers do at least object against guidance and governance in the early activities of software development. In these activities human problem solving is needed most urgently and most supporting tools tend to overburden these activities with guidelines, rules and “syntactical” support. Instead of using such tools, software developers are used to their favourite tools, to certain processes (often only to fragments of overall software processes), and to certain ways of communication with each other and with the supporting infrastructure.

2. The Key Concept of Process-Centered Software Engineering Environments

Process-centered software engineering environments (PSEEs) give up the notion of a predefined process model which has to be applied in every project. In contrast to this production process like notion of a software process, PSEEs support a wider variety of processes. Sometimes this wider variety is based on parameterization of the PSEE (in extremely simple cases by defining the tools to be used). Other PSEEs enable flexible software process models by considering the process model as parameter. Process models used in this way should define, which software development activities are to be executed when and by whom, they should identify tools to be used and the format of documents to be created and/or manipulated. Then, the process model is interpreted at runtime to identify which process behavior has to be supported. In concrete this means, that software developers are reminded of activities which have to be carried out, automatic activities are executed without human interaction and consistency between documents is enforced up to a certain level.

The notion of flexible process support costs an extra price. The more flexible and adaptable the PSEE is (in other words the wider the variety of processes which can be supported), the weaker is the support. If, for example, arbitrary tools can be integrated then the PSEE cannot do much more than just calling the tools in the right order. It can very obviously not enforce more detailed consistency conditions.

A compromise is to let a PSEE support a certain range of processes. This range may be defined in terms of a metamodel which has to be respected by concrete software process models. In this case, a process model has to be expressed in terms of predefined object types, relationships, etc. which are defined in the metamodel. The PSEE is then capable of supporting all process models expressed in terms of the metamodel.

3. State of the Art

In the academic arena many prototypes have been built to illustrate the concepts of flexible PSEEs. Their main differences are the used formalism to define a process model and the underlying communication and database architecture. Formalisms cover, e.g. Petri Net-based operational specification, rule-based Prolog-like specifications, imperative scripting languages or object-oriented approaches.


4. Current Trends

It has been recognized that software processes encompass parts which are similar to production processes, other parts, which are engineering like processes and even some parts which are so closely depending on human creativity that they could be considered as essentially creative. The nature of the overall process is that different process parts demand for different kinds of support. In accordance to this finding, PSEEs have to provide flexible support mechanisms:

• Early software development activities should not be prescribed in detail. Experience from earlier processes should be made accessible. Cooperation between teams should be enabled, but it should not be enforced.
• Specification, design and implementation activities should be handled as engineering processes. Software developers should use proven tools and they should follow abstractions which have proven useful. Good
software engineering principles should be applied wherever possible. Well-proven analysis and design patterns should be offered proactively by the PSEE.

- Other parts, like many configuration management and quality management activities should be described, they should be followed in detail and the PSEE should even enforce them.

If we want to follow the idea to consider process models as parameters for PSEEs, then it has not only to be defined which activities have to be executed, but also which process parts are of which type (creative, engineering production).

Another trend for PSEEs is that software processes become more and more mobile, distributed and federated. Thus, PSEEs should not assume that software processes are central processes to be carried out by a single team at a single site. Instead they have to offer infrastructure for processes which take place at many sites, for software developers who are not statically allocated to a single site and for different ways to carry out processes in a federated way. PSEEs, therefore, have to provide typical middleware services in addition to software process specific services.

In academic research we find various contributions which propose solutions to these challenges. The middleware infrastructure developed in the PIE project, services provided by the APEL platform, mechanisms identified in the scope of JIL and the Process Landscaping approach explicitly deal with problems addressed above.

5. Academic Concepts and Impact on Industrial Software Development

On the first glance, one may conclude that PSEEs did not have an impact on industrial software development. In fact, there are hardly any software engineering environments which could be called process-centered. But there are hardly any closed and process-specific environments left neither. Industry has recognized that highly integrated environments which can hardly be adopted to company specific needs do not keep the promises of higher productivity and better quality software. Instead, the software process to be supported has gained more attention. More and more often, tool support for processes follows a best-of-breed approach. Companies chose a specification tool, a programming environment, a test tool, a configuration management tool (and others) and expect that these can be integrated to enable a company specific software process.

Thus, processes are of interest and the interest has been raised by the notion of PSEEs. PSEEs themselves are not a major success story yet. Considering that many industry software processes do not differ dramatically (at least for information system development most follow object oriented principles based on subsets of UML), there may be room for PSEEs which are capable of supporting a range of software processes. These will not be general purpose PSEEs (as aspired to some ten years ago), but they have the potential to play a beneficial role in industrial software development.

6. Conclusion

The ambitious vision of PSEEs was not overwhelmingly successful as such. But it substantially influenced our understanding of software processes and infrastructure needed to support software processes. A close look at the state of the practice shows that the convergence in actual software processes may be a milestone for a less ambitious goal: PSEEs which are capable of supporting object oriented software processes, which take place at different sites, which require cooperation of many teams and which demand for powerful middleware services.

Specification and Modeling: An Academic Perspective

Manfred Broy
Institut für Informatik, Technische Universität München, D-80290 München, Germany
broy@in.tum.de

Abstract

The specification and modeling of software systems, of their aspects, and their development processes is at the heart of software engineering. Over the years, we have achieved a much deeper and more comprehensive understanding of software and its models as a basis for its specification. However, there is still a way to go to make sure that all we know right now is transferred into practice, and that all we do not understand so far is investigated in depth. The goal is a tractable scientific basis for modeling and specification in programming, software and system engineering and its employment in engineering methods.

Keywords
Modeling, Specification, Requirements Engineering, Quality Assurance, Tool Support