

Effects of Architecture and Technical Development Process on Micro-Process

Liming Zhu, Ross Jeffery, Mark Staples, Ming Huo, Tu Tak Tran

NICTA, Australian Technology Park, Eveleigh, NSW, Australia
School of Computer Science and Engineering, University of New South Wales, Australia
{Liming.Zhu, Ross.Jeffery, Mark.Staples, Ming.Huo, TuTak.Tran}@nicta.com.au

Abstract. Current software development methodologies (such as agile and RUP) are largely management-centred, macro-process life-cycle models. While they may include some fine-grained micro-process development practices, they usually provide little concrete guidance on appropriate micro-process level day-to-day development activities. The major factors that affect such micro-process activities are not well understood. We propose that software architecture and technical development processes are two major factors. We describe how these two factors affect micro-process activities. We validate our claim by mining micro-processes from two commercial projects and investigating relationships with software architecture and technical development processes.

Keywords: micro-process, macro-process, architecture, process mining

1 Introduction

The increasing demands imposed on software-intensive systems require more rigorous engineering and management of integration among the products being developed, the technology being used and software development processes [8, 16].

In process engineering, a macro-process describes the high-level behaviours of process while micro-process describes the fine-grained internal workings of processes [15]. Current software development methodologies (such as Agile and RUP) are largely project management-centred, macro-process life cycle models. While they may include some fine-grained micro-process development practices, they usually provide little concrete guidance on appropriate micro-process level day-to-day development activities. This has hindered the wide adoption of rigorous development processes by developers because they do not usually find macro-processes useful for their immediate needs at a micro-level. This gap between macro-processes and micro-processes has been recognized previously [15], [13]. In this paper, we suggest that software architecture and technical development are two major factors that affect fine-grained micro-processes:

- 1) Software architecture is important for decomposing a system into functional modules. This can be used to support task allocation when planning development. However, architecture has other influences on development. We observe that module dependency, degree of such dependency, and architectural refactoring play major roles for micro-processes.

2) A technical development process is a development process for a particular technology, such as XML, service orientation, object orientation or a programming language. Technical development processes are composed of technical steps, best practices, and checklists for different types of technology-specific components at different stages. However, these are not necessarily aligned with the normal phases of software development or project iterations. Companies consider technical development processes an important addition to their macro-process, bringing competitive advantage [10]. Some technologies, such as programming languages and object orientation, have not been considered to have a major impact on the normal flow of a development process, but have rather been associated with development efficiency and product quality. We observe that some new technologies, such as XML and service orientation, do have major effects on the development process. This is largely because such technologies are not confined to the design and implementation phases. XML has been used to directly define business level requirements and communication standards. Service governance has become a normal business activity, since it gives direct control over service development beyond the phases of initial development.

We have conducted micro-level process mining in two commercial projects. After excluding the “normal” micro-process activities, such as detailed designing, implementing, testing a single module, integration and system testing of a particular sub-system, we find most of the micro-processes and activities that are significant in terms of effort and recurring patterns are indeed affected by software architecture and technical development processes. These effects are reflected in micro-process activities themselves, cost of the micro-process activities and recurring process patterns. We use effort and recurring patterns as indicators of importance among hundreds of activities. As later revealed in structured interviews, explicitly considering these factors may increase the efficiency of the process and the quality of the project and can give more concrete guidance for developers at a micro-level.

The main contribution of this work is to provide a better understanding of the major factors that can influence micro-processes. It will be valuable in offering further assistance to actively planning micro-processes and bridging the gap between the macro-process and micro-process.

2 Related Work

Connections between macro-processes and micro-processes are usually created through organization- or project-specific process tailoring, which can be either top down or bottom up [4, 7, 14, 18]

In the top down approach, a macro-process is instantiated by considering both project characteristics and organizational factors. It has been recognized that the main problem here is the low level of reuse during tailoring; additional context information used in the tailoring is not systematically reused [7]. Moreover, the resulting process is not fine-grained enough to provide concrete guidance to developers. The tailoring process is usually about making or selecting optional process elements, or modifying variant process elements [1]. Since such process elements usually do not include

technology- and product-specific information, the resulting process is not fine-grained and “micro” enough. This paper addresses these challenges by aiming to develop a new understanding of major factors in fine-grained tailoring, such as product architecture and technology-specific development processes.

In the bottom-up approach, an organization usually has a large number of process assets. However, the composibility of these assets is often poor and the suitability of a composed process hard to verify [7]. The research in this paper does not address this issue directly; however, we believe that explicitly considering the additional factors has the potential to significantly improve the composibility of these assets and to provide further criteria for assessing the suitability of the composed process.

Both approaches are used to produce development processes for a particular type of project or technology, such as embedded systems [10] and COTS [12]. However, both the top-down and bottom-up approaches are not “micro” enough and the resulting processes are hard to verify.

Traditionally, software architecture affects the software development process in two different ways. First, architecture design and evaluation methods constitute part of the whole life-cycle macro-process. Currently there is active research into streamlining these methods by using additional phases [11]. Such streamlining does consider the architecture itself, but only the method for producing it. Secondly, architecture is used to decompose systems into functional modules, which can then be used in task allocation. This reflects some finer-grained processes, but is confined to the early planning phase and is fairly simplistic, so that the full potential of the architecture is not exploited. In this study, we look into taking more advantage of architecture in the development process.

We believe the missing link between micro-process and macro-process can be further bridged by considering the architecture and technical development processes.

3 Effects of Architecture and Technical Development Process on Micro-Process

3.1 A Framework of Factors that Affect Micro-process

We first propose a framework that includes all the factors that may have influence on micro-process, as shown in Figure 1. Some of them, such as macro-process methodologies, are understood at least in terms of the existence of their influence, although the exact nature and size of their influence is not completely understood. Our recent work has explored the existence of other factors, such as the effect of business processes and business data on software development processes [9].

1. **Macro-process methodologies:** As mentioned earlier, the generic software development process is the foundation for any fine-grained micro process through top-down or bottom-up process tailoring.
2. **Functional Requirements:** Functional requirements, especially functional scoping in iteration planning define what needs to be developed at a fine-grained level. However, they provide information only on ‘what’ to do but not ‘how’ to do it.

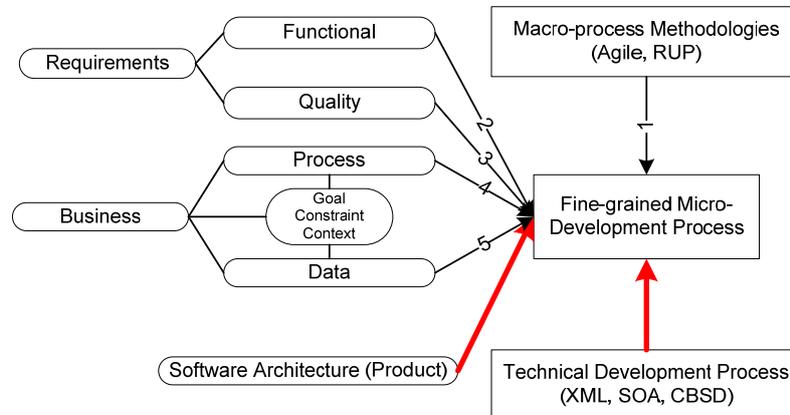


Figure 1. Factors that affect fine-grained micro-process

3. **Quality requirements:** Quality requirements (or non-functional requirements) guide additional quality-related processes. For example, a high-reliability system may require more testing to be done, or the use of a particular testing technique.
4. **Business Process:** The need for mechanisms to support the analysis and tracing of relationships between the business process and the software process is discussed in [9]. It is critical for instantiating elements of that business process in software.
5. **Business Data:** Industries have developed electronic business data “standards” to improve business efficiency and business-to-business interoperability. Such standards inevitably have to map to technology infrastructures such as service-oriented architecture. The design decisions embodied in the business data standards often affect the development process at a fine-grained level.

In this paper, we propose two additional factors: software architecture and the technical development process.

3.2 Software Architecture Factor

Architectures provide a wide range of information that can benefit micro-process planning and monitoring. Architectural models of inter-module dependency are particularly relevant for micro-process. We propose that architectural dependency models will influence micro-processes in ways including but not limited to the following:

Claim 1: A micro-process activity usually concerns multiple inter-dependent architecture modules at the same time.

For example:

- All modules involved in a single micro-process activity when designing, implementing and testing a module with high dependency.

This is important because coupling information between modules has not been explicitly used in fine-grained process planning. Tightly coupled modules may be

only suitable for a close team to develop while loosely coupled modules can be developed in a distributed and parallel manner.

Claim 2: The cost of a micro-process activity on a module will be affected by the architectural dependency characteristics of the module.

Examples of this include:

- Cost of code understanding may be high when the module has high dependency.
- Cost of integration testing will be higher between highly cross-dependent modules

This is important because cost estimation models usually only concern with sizes of features and function points. Dependencies between functions are not recognised as a cost factor. Some generic complexity metrics have been used in cost estimation but are not useful for fine-grained activity costing.

Claim 3: Micro-process patterns can often be better explained by the influence of the various development stages of different inter-dependent modules rather than by macro-process phases.

For example:

- During stub creation for unit and integration testing when one unit relies on the existence of another yet-to-be developed module.

This is important because increasingly, large-scale software is developed in a concurrent manner. The nature of interactions between these parallel development processes is important, and architecture is an important factor.

We realize that process planning usually starts at an early stage, sometimes even before a contract is awarded and the architecture is known. However, we should have a plan about how fine-grained processes will respond to architecture definitions.

3.3 Technical Development Process Factor

We propose that a technical development process will affect micro-processes by imposing technology-specific activities, sequences and best practices through process interactions with macro-processes:

Claim 1: A micro-process activity usually “is” an activity described by a technical development process rather than an instantiated or tailored macro-process activity.

For example:

- XML schema development processes are technical processes and are also micro-processes.

This is important because limitations exist in instantiating or parameterizing macro-processes. A complete top-down approach will never be able to cover the richness of technical development process activities.

Claim 2: The cost of a micro-process activity will be affected by its technical characteristics rather than macro-process activities. For example:

- Designing, implementing and testing unique types of technology modules. In the case of XML development, data update module, up translate/down translate and cross-translate modules are developed very differently with different cost implications.

This is important because the technical characteristics of an activity have not been used in cost estimation models. However, they may have definitive cost implications.

Claim 3: Micro-process patterns are often determined directly by a technical development process or interactions between a technical development process and a macro-process rather than a tailored macro-process. For example:

- Designing schema in XML development has a unique sequence of activities. It little resembles traditional macro-processes.
- A technical development process requires compliance with certain best practices. A micro-process pattern emerges from interactions between the compliance processes and the normal development process.

This is important because most process patterns are currently related to macro-process methodology. Factors affecting micro-process patterns need to be investigated and eventually used in process planning.

4 Case Study

The primary goal of this case study is to validate the existence of strong influence of architecture and technical development process in micro-process, through process mining, architecture reconstruction and structured interviews. Using these factors and measuring their effects is beyond the scope of this work.

4.1 Project Selection

The details of the projects will be described in section 4.3 and 4.4, along with the analysis. The general reasons for selecting these two projects are as follows:

- These are two typical and representative projects within the company, not pilot projects for trialling a new technology, nor instrumented with any particular process measurement techniques other than what the company is already doing.
- Time sheets for fine-grained process measurement (in addition to billing) are recorded for all projects. They directly record individual micro-level activities, and also enable us to mine process patterns from recurring sequences of activities.
- The company has explicitly used software or system architecture in their process planning. However, only functional module decomposition is used. In one project, we helped them reconstruct additional architectural views to investigate the influence of architecture on micro-processes. The influence of technical development processes (concerning Java) is also evident in this project.
- The company considers that their major process competitive advantage is their technical development process, in this case related to XML technologies. These processes are actively used, but are not systematically integrated with their macro-process definition. In the XML project, we mainly investigate the technical development processes, although the architecture factor also has some influence.

To build their competitive advantage, the company has focussed on their micro-processes in XML development. Historically, these are materialized in technical checklists, best practices, metrics and governance, loosely grouped around macro processes in an EPG system. However, technical development processes are not used in their process planning and monitoring.

4.2 Data Source and Techniques

The evidence for this case study is collected from multiple sources to avoid any single-source bias. The data includes source code (for architecture reconstruction), documentation, time sheets (for micro-process mining) and interviews.

Statistical Mining

To understand how a micro-process is affected by certain factors, we need information about activities at the micro-level. The time sheets recorded at the company directly provide us with this information. For example:

	A	B	C	D	E	F	G	H
1	Project	TaskNo	Employee	Hours	Comments	module	epgType	activityID
2	2307	41	56	7	testing and integration of the profile manager	screeningTool	DD/CODE	253
3	2307	41	56	1.5	design and coding of the User and UserHome classes	screeningTool	DD/CODE	254
4	2307	41	56	4	Modify code accordingly to design changes, wrote impl	screeningTool	DD/CODE	255

From this data, we can extract recurring sequences of activities as micro-process patterns. Our previous work has successfully performed such mining [5]. Essentially, we reconstruct a recurring sequence of activities as a micro-process pattern.

Out of all the activities and recurring sequences of activities that we mined, we excluded the “normal” micro-process activities, such as detailed designing, implementing, testing a single module, integration and system testing of a particular sub-system. We then selected the most effort-wise significant micro-processes and activities and analysed their relationship with architecture and the technical development process. For the mining of the two projects, please refer to [6]. This paper only includes a subset of all the mined activities and micro-process patterns that are relevant to architecture and technical development process.

Reverse Architecting from Source Code

To understand architectural influence on micro-process, we need relevant architecture views of a system. We associate these views with the mined micro-process. Although very high-level architecture views exist for both projects, it was still necessary to reconstruct views that reflected dependency and degree of dependency between modules.

We used two tools, JDepend and Structural Analysis for Java (SA4J), to conduct an architecture reconstruction. The aim was to reconstruct the dependency views between components and see if the degree of dependency influenced micro-process activities. The following is a brief summary of the each tool’s capabilities related to dependency:

JDepend - JDepend is an open source tool which provides design metrics beyond traditional class-level OO metrics by looking at cross-module quantitative inter-dependency. Among the many metrics supported, the following proved to be particularly relevant:

- **Afferent/Efferent Couplings of Modules:** They indicate the outgoing and incoming dependency degree for a particular module.
- **Instability of Modules:** This is an indicator of module’s relative *resilience to change*.

Structural Analysis For Java (SA4J) - SA4J is a tool from IBM for analysing Java dependencies. The uniqueness of this tool is its transitive impact analysis and skeleton diagrams for indirect dependency. They are different from the standard coupling measurement and appear to be more useful in micro-process. The following metrics are the most relevant ones:

- **Global Butterfly:** If the module is changed, it may affect many other components.
- **Global Breakable:** The module is often affected if anything in the system is changed.
- **Global Hub:** The module is both a global butterfly and a global breakable.
- **Skeleton:** This layered view of the system is constructed by putting modules with no dependencies on the bottom layer. Modules that are dependent on the lowest layer appear in the above layer, and so on. In this view, a stable system should have a pyramid shape. An unstable system may look like an upside down pyramid.

Structured Interviews

Structured interviews were used to validate our process mining findings, to avoid any single source bias, and to get further insights. We presented micro-process patterns that were not aligned with macro-processes documented and used through an electronic process guidance system in the company. In the interview we elicited causes of these deviations.

4.3 Project A: the Finance Project

Project Description

This project demonstrates the influence of architecture on micro-process. The code base is an integral part of a series of financial products that are written in Java. The system generates financial data which can be accessed by subscribers to the service. The financial data produced by the application needs to be generated in an extremely flexible manner so that it can be easily tailored to each subscriber's needs. The content delivery mechanism exploits XML formats and XSLT transformations to render tailored views. The code has been refactored on several occasions to progressively improve the design quality and functionality. It has 50k lines of code with 296 classes/interfaces in 27 packages. It took 1600 man hours to produce.

Reconstructed Architecture

Figure 2 shows the reconstructed module dependency view and skeleton view. Drilling down the dependency line can reveal the degree of dependency. The skeleton view also reflects indirect dependency. The skeleton diagram shown here particularly depicts the dependencies of the *util* module. The grey squares represent classes and interfaces in the whole system. The red (black)¹ squares represent the classes/interfaces in the *util* package. The orange (light grey) squares represent the

¹ Colors in braces are for black and white prints of this document.

classes/interfaces that depend on the *util* (red) package. For details of the reconstruction, please refer to [3].

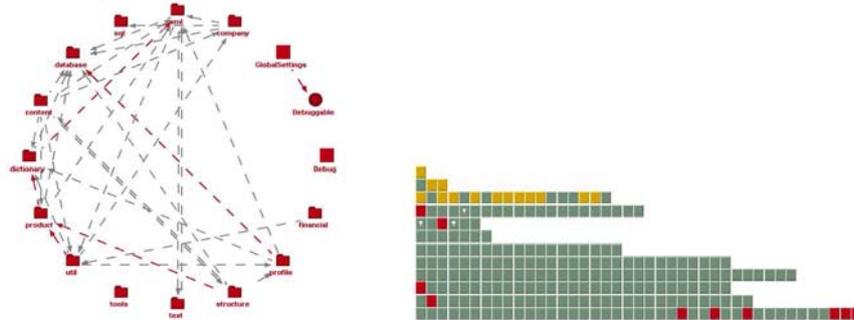


Figure 2. Module Dependency Diagram and Skeleton Diagram

The tool returned a ranked list of global breakables, global butterflies and global hubs. The top 3 modules for each of them are:

- Global Breakable: Content.ViewPortfolio, Content.ViewPortfolioProfile
- Global Butterfly: XMLUtils, Debuggable, XMLSerializable
- Global Hub: HomeFactory, LicenceHome, HomeManufacturable

These highly ranked modules are used in cross-checking with modules involved in unusual micro-process activities and cost.

Data Analysis

According to our criteria for mining, we selected the most effect-wise significant activities. Among the most costly activities (more than 30 man-hours to complete) and recurring process patterns, the following nine have a direct relationship with architecture. Each of them supports claims about architecture influence on micro-processes in section 3.2. We confirmed these findings in follow-on interviews.

- Conducting integration testing between two highly dependent modules [claim 2]
- Designing a specific module when understanding of a highly dependent module is needed [claim 2]
- Understanding and analysing existing modules. These modules are global hubs. [claim 1]
- Refactoring at architectural level, including creating new modules that were not previously used for task allocation [claim 1]
- Cleaning up code for integration of two highly dependent modules [claim 1]
- Making certain part of the UI “smart” (dynamic rather than hand-coded) [claim 1]
- Working on a logical group of dependent modules together [claim 1]
- Producing cross-phase process patterns between design and implementation when detailed design is not available [claim 3]
- Extending existing open-source modules when there is a high dependency between modules [claim 3]

4.4 Project B: the XML Project

Project Description

This project demonstrates the influence of technical development processes. The project developed a Java tool with XML processing capabilities. The project took 2900 man-hours to complete. We did not have access to the source code, and so the architecture recovery tools could not be used, but a high-level system architecture with functional decomposition was made available, and was matched against the modules recorded in the time sheet records.

XML Technical Development Process

The following is a high-level view of the XML technical development process the company has used. For each of the activities and sub-activities there are associated best practices, activities and checklists. The process is largely derived from previous work on XML process models [17].

- Develop (Analyze/Design/Implement/Test) data capture module
 - Create Data
 - Update Data
- Develop data query modules
 - Server-side query module
 - Client-side query module
- Develop data transformation module
 - Up-translate module: transform non-XML documents to XML documents
 - Down-translate module: transform XML documents to non-XML documents
 - Cross-translate module: transform XML documents to XML documents
- Develop intermediary schemas.

Data Analysis

The criteria for the following activities or sequence of activities are the same as before. Among the most costly activities (more than 30 man-hours to complete) and micro-process patterns, the following seven have a direct relationship with the technical development process. Each of them supports claims about technical development processes in section 3.3. We confirmed these findings in follow-on interviews.

- Create and code intermediary XSD [claim 1 and 2]
- Technology investigation and learning (even for the developers, who are experienced XML developers, understanding certain new trends and best practices took a significant amount of time) [claim 1 and 2]
- Set up the technology environment
- Develop a data update module [claim 1 and 2]
- Develop a up-translate module [claim 1 and 2]
- Develop a client side query module [claim 1 and 2]
- Schema re-design during implementation [claim 3]
- Requirement negotiation during development [claim 3]

For example, the cost effects had a distinctive pattern: developing the cross-translate component takes the least amount of time, while developing the up-translate component takes the most amount of time.

5 Discussion

A number of cross macro-process phase activities were found during the mining. They do not follow the macro-process, even considering iterations. This strongly validates the observations on programming rework [2] which sees cross-phase rework are much more complicated than simple redo and iteration. However, we have not yet investigated the factors affecting rework.

In our previous work [6], we thought discrepancies were due to enactment problems or that the macro-process needed to be changed. However, after further study, we have found that it is due to the nature of differences between macro-process and micro-process. Iterations on a macro-process level involve finishing one phase then going to the next one, usually over a period of days, if not weeks. At a more detailed level, we have observed that developers switch between phases in a unusual order for many reasons, including:

- Highly-coupled modules at different development stages
- Requirements maturity for a particular module
- Unique features of some technical development process
- Interactions among different processes (such as generic processes, technical development processes, quality assurance and compliant processes) at different abstraction levels

This is especially evident in the Project B, due to the nature of XML development, which is more about understanding the requirements and making tradeoffs in schema development (sometimes this is even done by the customer), and incorporating technical best practices and strong quality assurance to the normal development in a parallel development fashion.

We realize that there are a number of limitations of this study:

1) There are actually a large number of factors, as identified in Figure 1. We only considered two factors in this paper due to limits to the scope of our investigation. We realize that certain factors may play a more dominant role than others.

2) Using these factors actively in process planning may be different to after-event observation. We are planning a case study on actively investigating the use of these factors in process planning.

6 Conclusion and Future Work

Across the industry, more sophisticated process engineering is needed. This requires increased understanding of fine-grained micro-process and filling the gap between macro- and micro-processes. In this paper, we investigated two major factors: software architecture and the technical development process. We are currently planning a new full-scale case study on using these factors in continuous

micro-process planning and monitoring. We are also developing a technical governance framework to be used by both management and developers for communicating effectively.

References

- [1] J. Bhuta, B. Boehm, and S. Meyers, *Process Elements: Components of Software Process Architectures*, 2005.
- [2] A. Cass and L. Osterweil, "Programming Rework in Software Processes," Department of Computer Science, University of Massachusetts UM-CS-2002-025, 2002.
- [3] I. Gorton and L. Zhu, "Tool Support for Just-in-Time Architecture Reconstruction and Evaluation: An Experience Report," in *27th International Conference on Software Engineering (ICSE)*, 2005, pp. 514 - 523.
- [4] G. K. Hanssen, H. Westerheim, and F. O. Bjornson, "Tailoring RUP to a Defined Project Type: A Case Study," in *Product Focused Software Process Improvement (PROFES)*, 2005, pp. 314-327.
- [5] M. Huo, H. Zhang, and R. Jeffery, "An exploratory study of process enactment as input to software process improvement," in *International Workshop on Software Quality at International Conference on Software Engineering (ICSE)*, Shanghai, 2006.
- [6] M. Huo, H. Zhang, and R. Jeffery, "A Systematic Approach to Process Enactment Analysis as Input to Software Process Improvement or Tailoring," in *Asia-Pacific Software Engineering Conference (APSEC)*, 2006.
- [7] O. Jaufman and J. Munch, "Acquisition of a Project-Specific Process," in *Product Focused Software Process Improvement (PROFES)*, 2005, pp. 328-342.
- [8] R. Jeffery, "Achieving Software Development Performance Improvement Through Process Change," in *Software Process Workshop (SPW)*, Beijing, China, 2005, pp. 43-53.
- [9] R. Jeffery, "Exploring the Business Process-Software Process Relationship," in *Software Process Workshop/Workshop on Software Process Simulation and Modeling (SPW/ProSim)*, 2006.
- [10] E. Johansson, J. Nedstam, F. Wartenberg, and M. Host, *A Qualitative Methodology for Tailoring SPE Activities in Embedded Platform Development*, 2005.
- [11] R. Kazman, H. P. In, and H.-M. Chen, "From requirements negotiation to software architecture decisions," *Information and Software Technology*, vol. 47, pp. 511-520, 2005.
- [12] M. Morisio, C. B. Seaman, V. R. Basili, A. T. Parra, S. E. Kraft, and S. E. Condon, "COTS-based software development: Processes and open issues," *Journal of Systems and Software*, vol. 61, pp. 189-199, 2002.
- [13] J. Münch, "Transformation-based Creation of Custom-tailored Software Process Models," in *International Workshop on Software Process Simulation and Modeling (ProSim)*, Scotland, UK, 2004.
- [14] A. Ocampo, F. Bella, and J. Münch, "Software Process Commonality Analysis," in *International Workshop on Software Process Simulation and Modeling (ProSim)*, Scotland, UK, 2004.
- [15] L. Osterweil, "Unifying Microprocess and Macroprocess Research," in *Software Process Workshop (SPW)*, 2005, pp. 68-74.
- [16] D. Rombach, "Integrated Software Process and Product Lines," in *International Software Process Workshop (SPW)*, 2005, pp. 83-90.
- [17] T. T. Tran, "An Interim Report of XML Process Models," School of Computer Science and Engineering, University of New South Wales UNSW-CSE-TR-0613, 2006.
- [18] H. Washizaki, "Building Software Process Line Architectures from Bottom Up," in *Product-Focused Software Process Improvement (PROFES)*, 2006, pp. 415-421.