Quality Attribute Techniques Framework

Yin Kia Chiam[1,2,3], Liming Zhu[1,2], and Mark Staples[1,2]

[1] NICTA, Locked Bag 9013, Alexandria NSW 1435, Australia
{yinkia.chiam,liming.zhu,mark.staples}@nicta.com.au
[2] School of Computer Science and Engineering, K17, University of New South Wales, Sydney, NSW 2052, Australia
[3] School of Computer Science, University of Malaya, 50603 Kuala Lumpur, Malaysia

Abstract. The quality of software is achieved during its development. Development teams use various techniques to investigate, evaluate and control potential quality problems in their systems. These "Quality Attribute Techniques" target specific product qualities such as safety or security. This paper proposes a framework to capture important characteristics of these techniques. The framework is intended to support process tailoring, by facilitating the selection of techniques for inclusion into process models that target specific product qualities. We use risk management as a theory to accommodate techniques for many product qualities and lifecycle phases. Safety techniques have motivated the framework, and safety and performance techniques have been used to evaluate the framework. The evaluation demonstrates the ability of quality risk management to cover the development lifecycle and to accommodate two different product qualities. We identify advantages and limitations of the framework, and discuss future research on the framework.

Keywords: Quality Attribute Techniques, Product Quality, Software Process Improvement, Process Tailoring.

1 Introduction

The process research framework presented by SEI's IPRC states that "In an ideal future state, the use of processes is part of accepted practice to ensure that acceptable levels of product qualities are in place during all stages of the software and system development life cycle" [1, p.24]. It is during software development that product qualities such as safety, performance, reliability and security are determined. It is costly and time consuming to fix quality problems at later development stages if a system fails to meet specified levels of product quality. Research questions identified by the IPRC [1, p.27] in this area highlight the importance of understanding how software processes can be created to target product quality goals: "How do we select processes to meet specific product quality requirements?" and "What process steps significantly influence the achievement of a specified level of product quality?".

Software engineers use a variety of specific techniques to investigate, evaluate, and control potential quality problems throughout the development of a system.

R.V. O'Connor et al. (Eds.): EuroSPI 2009, CCIS 42, pp. 173–184, 2009.

In this paper, we call these "Quality Attribute Techniques" (QAT). These QATs are usually technical engineering techniques [2] that are specific to individual product quality issues. Examples of QATs for safety include hazard analysis techniques such as Failure Mode and Effect Analysis (FMEA) and Fault Tree Analysis (FTA). QATs may be specific to a single phase of the development lifecycle, or span multiple phases. However, QATs are usually not explicitly detailed in software process models, and the relationship between QATs and other process elements are not usually clearly shown. In order to create software process models that target specific quality attributes, it is important to first understand the important characteristics of QATs and how they relate to the development process. If quality procedures are left implicit in software process models, then tasks related to quality problems can be forgotten when individuals leave development teams [3, p.2].

Most software process tailoring methodologies are designed to address variations in project context such as customer characteristics or the size of the product or development team [4,5,6]. The research literature has not normally regarded product quality as an important characteristic for software process tailoring. So, although QATs are used in practice by software engineers, they are not currently represented in detail or incorporated well in software development process models [2,7]. In practice, such information is usually informally described in process documentation. The existence of a repository of codified knowledge about QATs could help development teams to better understand the potential effect of using various QATs to target key product qualities across all phases of the software development process.

This paper proposes and evaluates a Quality Attribute Technique Framework (QATF) for capturing important information about QATs. The QATF is intended to provide a basis for creating a catalogue of QATs to support software process tailoring to target a specific product quality attribute. Elements of the QATF focus on information required for decision making during QAT selection and integration with development processes. We have used safety techniques to motivate the framework, and evaluate the framework using safety and performance techniques. We use risk management as a general theory to encompass a variety of product qualities. The following two research questions are the focus of this paper:

1. What characteristics of QATs are useful to select QATs for inclusion in a tailored software process that targets a specific product quality attribute?
2. What characteristics of QATs are useful to integrate QATs into software development process models?

The outline of this report is as follows. Section 2 discusses work related to this research. Section 3 describes the QATF. Section 4 presents an evaluation of the QATF using safety and performance techniques. Section 5 discusses limitations and advantages of the QATF arising from the evaluation. Section 6 presents conclusions and discusses future research.

2 Related Work

Most earlier research on helping development teams to achieve specific product qualities has focused on techniques and guidelines for specific quality attributes (e.g. [8,9,10,11,12]) or specific lifecycle phases (e.g. [13]). It is important to address quality throughout the entire development process. Process engineers select appropriate techniques and incorporate them into development processes created or tailored for new projects. The selection and integration of techniques into defined process models requires relevant information about those techniques to be available and presented to process engineers.

There are some efforts in the safety area to describe information about safety techniques. The EWICS TC7 Software Sub-group [14] and Leveson [15, p.313-358] provide general descriptions of safety techniques and highlight advantages and disadvantages of using each technique. Characteristics related to process information have been discussed by Alberico et al. [11]) and Stephans [16]. Zurich Risk Engineering [17] has compared hazard analysis techniques from the resource perspective (team approach, documentation, time required and team leader expertise) and the scope perspective (result, analysis approach, depth of analysis, emphasize single or multiple failures). These (especially the resource perspective) are closely related to project characteristics. Some approaches in software performance engineering (SPE) such as [18,19] discuss integration of performance activities into the software development process. Vegas [20] has proposed a characterisation schema to identify the relevant information for testing techniques.

Previous approaches do not attempt to systematically capture and document the important information about QATs and their relationship with other process elements. Most of the safety guidelines and approaches still lack information which is important for QAT integration and process tailoring. Some attributes identified in [20] and [14] are only suitable for testing techniques and are not relevant to other types of QATs. Development teams need appropriate information to understand the characteristics of QATs, how they are incorporated into process models, and how they function to identify, analyse or control potential quality problems.

QATs for safety-critical system have been selected to motivate this initial framework because the area of system safety is well-established. There are many existing procedures, handbooks, standards, books and other references. In safety-critical systems, techniques are available to perform hazard evaluation, hazard control and hazard analysis. This does not affect the validity of the whole framework as all extracted characteristics are non-safety specific.

3 QATF

The ultimate goal of this research is to improve product quality by better integrating appropriate QATs into software process models. Information about QATs can support decision making during QAT selection and integration with development processes. The QAT framework (QATF) captures and presents information about QATs in a format intended to be suitable for process engineers

to understand QATs and to highlight the relationship between QATs and other process elements. This is to support development teams select appropriate QATs and incorporate them into process models and related process guides. The framework is intended to encompass QATs from many quality domains.

3.1 QAT Overview

QATs are used to identify, analyze, and control potential quality problems in the development of critical systems. For examples, safety critical systems are concerned with hazards to life, property or the environment, security-critical systems focus on resistance to external threats and malicious actions against integrity [15], and performance-critical systems emphasize response time or throughput [18]. However, despite their importance in practice, QATs are not usually represented in detail in software development process models. They are not well integrated with other process elements such as tasks, roles and work products across the different phases of process models.

3.2 Identifying Important Characteristics of QATs

The QATF was constructed using the following approach. An initial review of the software safety literature identified the QATs in the safety area. The review also provided information about characteristics of these QATs, such as aims, description, benefits, limitations and expertise required [15,21,16,22,17]. By referring to the software process modeling literature and simulation literature [23,24][23] and the safety literature [11,16], process characteristics such as input, output and performer have been identified and populated.

We analysed the differences and similarities between various types of safety techniques. Based on the purpose of selecting QATs and integrating them into process models, characteristics which are generic for all the QATs have been selected. These characteristics have been grouped into three perspectives: General Information, Process Tailoring and QAT selection. Metamodels such as SPEM [23] can be used to define software processes and their components. The Process Tailoring characteristics in our framework have been selected based on the basic process entities defined in SPEM: roles, activities, work products and guidance. According to SPEM, a software development process is a collaboration between multiple roles that execute operations called activities and have work products as inputs and outputs [23]. Guidance elements such as tools, guidelines and examples can be used to support or automate the execution of an activity. The QATF and the three perspectives are as follows.

General Information. The General Information perspective provides an overview of the functionality of a QAT.

- Technique Name: *Short and full name of the QAT.*
- Aims: *What the QAT helps or enables us to do.*
- Description: *A brief overview of the QAT.*

Process Tailoring Characteristics. The Process Tailoring perspective highlights the relationship of a QAT with elements in software process models.

- Main performer(s): *The roles of people who typically perform the QAT.*
- Optional Performer(s): *Other roles which can optionally perform or assist the QAT.*
- Phase(s): *The development process phase(s) in which the QAT is applied.*
- Input: *The work products (information or artefacts) needed to apply the QAT.*
- Output: *The temporary, intermediate or final products created or modified during the performance of the QAT.*
- Guidance Documents: *Additional documents (e.g. guidelines, templates or examples) that can be used to assist the performer to execute the QAT.*
- As Source Data for (optional): *Other techniques or process activities that rely on the outputs from the QAT.*

QAT Selection Characteristics. The QAT Selection perspective provides a more detailed and structured view of a QAT, including costs, benefits, and quality impact in terms of our risk-based theory of quality management.

- Category: *The quality risk to which the QAT belongs (refer to Section 3.3).*
- Benefits: *Principal benefits claimed for the QAT.*
- Limitations: *Specific difficulties or limitations associated with the QAT.*
- Cost of Application: *The level of effort and resources needed to perform the QAT.*
- Expertise: *The level of expertise or training required to perform this QAT.*
- Team/Individual approach: *Whether the QAT is a team approach or is performed by an individual.*
- Single/Multiple Failures Analysis: *Whether the QAT emphasizes single failures in isolation or is geared toward multiple failures in combination.*
- Tool(s): *Tool(s) that can be used to support this QAT.*

3.3 QAT Categorisation Based on Risk Management Process

Most prior research has focused on techniques for individual quality attributes or specific lifecycle phases. We are attempting to provide a more general framework for QATs, using risk management as a general theory for managing quality during development. We have found risk management to be useful in understanding how QATs function to affect quality by identifying, analyzing, and controlling potential quality problems. In the QATF, categories classify QATs according to the method by which they address quality risks. We believe this will help process engineers to incorporate appropriate QATs to better manage quality throughout the development process. For example, FMEA is useful for hazard analysis (safety risk analysis). Process engineers may include FMEA into process activities which require hazard analysis during the design phase.

According to [25, p.7], the risk management process is "a continuous process for systematically addressing risk throughout the life cycle of a product or

service". The risk-driven quality management process is inter-related with the normal software development processes. After defining quality objectives, potential quality risks can be identified. The impact of each potential risk can be analysed and ranked according to its probability of occurrence and severity of damage. The amount of effort to monitor, eliminate or prevent specific risks can be determined by the level of risks.

Based on the studies that discusses software risk management processes [26,25,15], QATs have been grouped into two main categories: Quality Assessment and Quality Control (see Fig. 1). Below, safety techniques are used as case examples in the description of these categories.

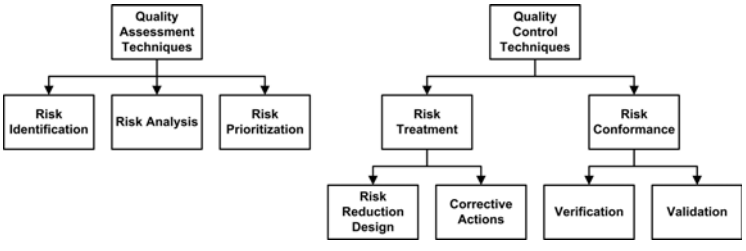

Fig. 1. Categorisation of QATs: Quality Assessment and Quality Control

Quality Assessment Techniques

1. Risk Identification - Involve QATs which produce lists of the project-specific quality risk items may compromise a project's satisfactory outcome. Typical QATs for safety include hazard identification techniques such as Hazard and Operability Study (HAZOP), "What if" Checklist.
2. Risk Analysis - Involve QATs which produce assessments of the probability and magnitude of losses associated with each of the identified quality risk items, and assessments of compound risks involved in risk-item interactions. Typical QATs for safety include hazard analysis techniques such as Failure Mode and Effect Analysis (FMEA), Fault Tree Analysis (FTA).
3. Risk Prioritisation - Involve QATs which produce a prioritised ordering of the quality risk items identified and analysed. Typical QATs for safety include techniques used to rank the impact of identified hazards such as Consequence Analysis, Criticality Analysis.

Quality Control Techniques

1. Risk Treatment - Involve QATs which resolve, reduce or eliminate risk items and take corrective action when appropriate. Typical QATs for safety include hazard reduction design such as simplification and decoupling or corrective actions such as improve error recovery (e.g. feedback, checking procedures, treating system failures and supervision).

2. Risk Conformance - Involve process of determining (verification) and confirming (validation) the quality specification of either a phase or that the complete system is fulfilled and is consistent with the quality requirements. Typical QATs for safety include verification and validation techniques such as Sneak Circuit Analysis, Control Flow Analysis and Boundary Value Analysis to ensure that the software product meet precise safety objectives.

4 Evaluation of QATF

This section provides an initial evaluation of the QATF in assessing its support for the integration of QATs into software development process models. Various types of QATs are available to identify, analyze, and control potential quality problems during software development. In this evaluation, the QATF has been used to capture information for some safety and performance techniques.

Table 1. Organising Safety QATs into Different Software Development Phases

Development Phase	**Safety Activities**	**Safety Techniques**
Requirements	Preliminary Hazard Identification (PHI)	ETBA, HAZOP, Checklist
	Preliminary Hazard Analysis (PHA)	ETBA, HAZOP
Architecture	Hazard Analysis (SSHA, SHA)	FMEA, FMECA, FTA, ETA
	Design Pattern	Homogeneous Redundancy Pattern, Diverse Redundancy Pattern, Monitor-Actuator Pattern
Design	Hazard Analysis (SSHA, SHA)	FMEA, FMECA, FTA, ETA
	Hazard Analysis (O&SHA)	PET, Procedural audits
	Safety Design	Design for controllability, Barriers (Lockouts, Lockins, Interlocks), fail-safe design
	Safety Design Review	Walkthroughs, Checklists, Fagan inspection, State transition diagrams, Time Petri nets
Coding	Safety Code Design	Error prevention (e.g. interlock); Error deduction (e.g. stepladder); Error recovery (e.g. warning)
	Safety Code Review	Emulation Analysis, Symbolic execution
	Design Patterns	Homogeneous Redundancy, Diverse Redundancy, Monitor-Actuator
Testing	Safety Testing	Sneak circuit analysis, Software common work analysis
	Hazard Analysis	FMEA, FMECA, FTA, ETA
	Independent Safety Audit	Safety Management Organisation Review Technique (SMORT)

4.1 Methodology

The first part of the evaluation organises these QATs into different development phases. The process tailoring information (e.g. phase, artifacts) captured by QATF is used to incorporate QATs into relevant development process phases. Safety and performance activities are used to describe the common purpose of using these QATs. The second part of evaluation organises QATs into different risk categories in Section 3.3 according to their aims in risk management. Safety and performance techniques are again used as examples for this evaluation.

4.2 Results of Evaluation

Table 1 shows some examples of the safety QATs and their fit into development process phases. For example, FMEA is suitable for subsystem hazard analysis (SSHA) and system hazard analysis (SHA) during the architecture and design phases. This QAT is not appropriate for hazard identification or preliminary hazard analysis (PHA) in earlier phases because it is intended to help to analyse potential failure causes and their effects. Corrective actions will be recommended to the potential hazards based on an assessment of their criticality. Detailed system information and descriptions are needed in order to perform this QAT.

Table 2. Organising Performance QATs into Different Software Development Phases

Development Phase	**Performance Activities**	**Performance Techniques**
Requirements	Define and Analyse Performance Requirements	Execution graphs (EG), Use Case Maps (UCM), Layered Queueing Network (LQN)
Architecture	Performance Prediction	Performance Assessment for Software Architecture (PASA), LQN, Performance Evaluation Process Algebra (PEPA), Stochastic Petri Nets (SPN)
	Performance-oriented design Principles and Patterns	Principles (e.g. Centering Principles, Shared Resource Principle), Patterns (e.g. Fast Path, Batching)
	Identify Performance Antipatterns	Antipatterns (e.g. Excessive Dynamic Allocation)
Design	Performance Prediction	LQN, Markov Chain
	Performance Principles	Principles (e.g. Locality Principle, Parallel Processing Principles)
	Identify Performance Antipatterns	Antipatterns (e.g. Circuitous Treasure Hunt)
Coding	Performance Solutions	Performance Patterns (e.g. Fast Path, Batching)
Testing	Performance Testing and Measurement	Load Test, Instrumentation (e.g. ARM, Paradyn), Benchmark
	Performance Enhancement	Performance Tuning

Table 3. Organising QATs into Different Risk Management Categories

Risk Category	**Safety Techniques**	**Performance Techniques**
Risk Identification	HAZOP, Checklist, ETBA	EG, UCM, Interactive Tree Algorithm, Performance Antipatterns
Risk Analysis	FMEA, FMECA, FTA, ETA	Software Architecture Analysis Method (SAAM), PASA; Layered queuing network (LQN), Stochastic Petri Nets, Markov Chains
Risk Prioritisation	FMEA, FMECA, Criticality Analysis, Consequence Analysis	Layered queueing network (LQN), Markov Chains
Risk Treatment - Risk Reduction Design	Hazard Reduction Design (e.g. Simplification and decoupling)	Principles (e.g. Locality, Parallel Processing)
Risk Treatment - Corrective Actions	Error Recovery (e.g. feedback, checking procedures)	Performance Tuning, Performance Patterns (e.g. Fast Path Speed-Up)
Risk Conformance - Verification	Sneak Circuit Analysis, Control Flow Analysis and Boundary Value Analysis	Load Testing, Stress Testing, Instrumentation
Risk Conformance - Validation	SMORT, Safety Review, Fagan Inspection	Benchmark, Profilers

Development teams can use the QATF to help them compare FMEA with other safety QATs such as FTA and ETA, to determine a sequence of using QATs by referring to the process information captured by the QATF. The most appropriate QATs can be selected to execute specific safety tasks. For example, FTA and ETA can be used when the design is completed. FTA begins with all hazards identified from other QATs such as FMEA and HAZOP and works backwards to determine their possible causes until reaching a base event. ETA uses inputs from QATs such as FTA to analyse all possible consequences and determine the percentage of consequences which lead to the desired result.

As with safety, there are various performance techniques available to identify and address performance problems throught development processes. These QATs include performance estimation techniques, performance modelling techniques, performance evaluation techniques to ensure that the implementation meets performance objectives and also some principles and patterns for performance design. Table 2 organises some performance QATs into different development phases. For example, there are a set of performance-oriented principles to identify design alternatives that help to meet specific performance objectives. Design engineers can use the QATF to help choose the most suitable principle by referring to the definition and examples of applying these principles.

Table 3 organises some of the safety and performance QATs into different quality risk management types (refer to Section 3.3). QATs are categorised based on

their aims and description captured by QATF. Development teams can choose QATs based on their action in quality risk management process, and then integrate them into their software processes.

5 Discussion

The section discusses some of the benefits and limitations of using QATF that we have observed to date. The QATF provides a systematic way to capture important information about QATs. The template provides some information about how QATs impact quality during development, and how QATs can be related to software development processes. The first part of our evaluation suggests that QATs can be integrated into process models by referring to the process tailoring characteristics in the QATF. Although the QATF was initially motivated and developed using safety techniques, our evaluation has shown that the framework is also relevant for other quality attributes (performance).

The risk management categories provide a way for development team to choose QATs based on the means by which they impact quality risks. Table 3 shows some examples of QATs which are been organised into different risk management categories. We expect that process improvements to identify, analyse, and control quality risks could be undertaken by integrating corresponding QATs into the appropriate development phases or activities. These tables also indicate that quality management processes are iterative and ongoing. For example, FMEA and FTA not only can be used for hazard analysis in the early development phases but also can been applied later during testing. New hazards will have been identified during the testing phase. These hazards can be analysed to decide on suitable risk treatments to control the identified hazards according to their severity and frequency of occurrence. However, the selection of QATs can be determined in practice by other considerations. For example, available expertise may be a limiting condition in adopting a new QAT. The framework contains elements to describe the resources and expertise required to use each QAT.

In the development of the QATF, we have catalogued some QATs for safety and performance, but the catalogue is incomplete. Joint efforts will be required between between researchers, process engineers, and quality experts in development teams to obtain a more complete catalogue of QATs across a wider range of product qualities. We intend that QAT selection strategies and process tailoring methods will be supported by this framework, but they are outside of the scope of the framework itself. Some selection strategies (e.g. [20,13]) and tailoring methods may be able to be extended to select and integrate QATs more effectively into development processes. Process modelling tools such as EPF or WAGNER may be able to be used to represent QAT information and tailor development process models using selected QATs.

6 Conclusions and Future Work

Quality Attribute Techniques (QATs) are used by development teams to create software with specific qualities. Potential quality problems can be identified,

analyzed and controlled by using appropriate QATs throughout the development process. Development teams need appropriate information about QATs to better understand their impact on quality and to better integrate QATs into process models. Previous process tailoring approaches do not attempt to systematically capture and document how QATs can be incorporated into process models. This study has investigated important characteristics of QATs and has proposed a framework to capture and present significant QAT information to support QAT selection and process tailoring.

On the basis of this characterisation, development teams can use the QATF to help identify important information about QATs and to place QATs into development phases. We have used risk management theory as a basis to characterise QATs according to the means by which they impact potential quality risks. This has let us develop a framework that addresses a variety product qualities. Although the QATF was motivated and developed using safety techniques, our initial evaluation has showed that the framework can be used for performance, and we expect that other product qualities will also be able to be treated within the framework. This framework has been generated according to our own view, further theoretical and empirical evaluation is required for this initial framework.

Our future work will develop process tailoring methods to select appropriate QATs according to the product quality goals in a development project, and to incorporate those QATs into software development process models. The SPEM metamodel and EPF Composer will be investigated in terms of their ability to support the representation of QAT information captured by QATF and also integration of QATs.

Acknowledgements. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. Forrester, E.: A Process Research Framework. The International Process Research Consortium (IPRC) (2006)
2. Zhu, L., Jeffery, D.R., Staples, M., Huo, M., Tran, T.T.: Effects of Architecture and Technical Development Process on Micro-process. In: Wang, Q., Pfahl, D., Raffo, D.M. (eds.) ICSP 2007. LNCS, vol. 4470, pp. 49–60. Springer, Heidelberg (2007)
3. Smith, C.U., Williams, L.G.: Best Practices for Software Performance Engineering. Technical report, Performance Engineering Services and Software Engineering Research (2003)
4. Basili, V.R., Rombach, H.D.: Tailoring the Software Process to Project Goals and Environments. In: International Conference on Software Engineering (ICSP), pp. 345–357 (1987)
5. Bowers, J., May, J., Melander, E., Baarman, M., Ayoob, A.: Tailoring XP for Large System Mission Critical Software Development. In: Wells, D., Williams, L. (eds.) XP 2002. LNCS, vol. 2418, pp. 100–111. Springer, Heidelberg (2002)

6. Pedreira, O., Piattini, M., Luaces, M.R., Brisaboa, N.R.: A Systematic Review of Software Process Tailoring. SIGSOFT Software Engineering Notes 32(3), 1–6 (2007)
7. Zhu, L., Tran, T.T., Staples, M., Jeffery, D.R.: Technical Development Process in the XML Domain. In: International Conference of Software Process, ICSP (2009)
8. Juristo, N., Ferre, X.: How to Integrate Usability into The Software Development Process. In: International Conference on Software engineering (ICSE 2006), pp. 1079–1080. ACM, New York (2006)
9. Lutz, R.R.: Targeting Safety-related Errors During Software Requirements Analysis. SIGSOFT Softw. Eng. Notes 18(5), 99–106 (1993)
10. Lawrence, J. D.: Software Safety Hazard Analysis Version 2.0. Technical report, Lawrence Livermore National Laboratory (1995)
11. Alberico, D., Bozarth, J., Brown, M., Gill, J., Mattern, S., McKinlay VI, A.: Software System Safety Handbook. A Technical and Managerial Team Approach (1999)
12. Borcsok, J., Schaefer, S.: Software Development for Safety-related Systems. In: International Conference on Systems (ICONS 2007), pp. 38–42 (2007)
13. Wojcicki, M.A., Strooper, P.: An Iterative Empirical Strategy for the Systematic Selection of a Combination of Verification and Validation Technologies. In: International Workshop on Software Quality (WoSQ 2007), p. 9 (2007)
14. EWICS TC7 Software Sub-group: Techniques for Verification and Validation of Safety-related Software. Computers and Standards 4(2), 101–112 (1985)
15. Leveson, N.: Safeware: System Safety and Computers. Addison-Wesley, Reading (1995)
16. Stephans, R.A.: System Safety for the 21st Century. Wiley, Chichester (2004)
17. Zurich Risk Engineering: Which Hazard Analysis? - A Selection Guide (1998)
18. Smith, C., Williams, L.: Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software. Addison-Wesley, Reading (2002)
19. Fox, G.: Performance Engineering as A Part of The Development Life Cycle for Large-Scale Software Systems. In: International Conference on Software Engineering (ICSP), pp. 85–94. ACM Press, New York (1989)
20. Vegas, S.: Identifying The Relevant Information for Software Testing Technique Selection. In: International Symposium on Empirical Software Engineering (2004)
21. Storey, N.: Safety Critical Computer Systems. Addison Wesley, Reading (1996)
22. Vincoli, J.W.: Basic Guide to System Safety. Wiley, Chichester (2006)
23. OMG: Software Process Engineering Metamodel (SPEM) Version 2.0 (2008)
24. Pfahl, D., Ruhe, G., Lebsanft, K., Stupperich, M.: Software Process Simulation with System Dynamics - A Tool for Learning and Decision Support. New Trends in Software Process Modelling. World Scientific 18, 57–90 (2006)
25. AS/NZS ISO/IEC 16085:2007: Risk Management (2007)
26. Boehm, B.W.: Software Risk Management. IEEE Computer Society, Los Alamitos (1989)