

Software Architecture Evolution through Evolvability Analysis

Ph.D. Thesis Proposal, March 10th, 2011

Hongyu Pei Breivold
ABB AB, Corporate Research
hongyu.pei-breivold@se.abb.com

Mälardalen University
School of Innovation, Design and Engineering
hongyu.pei-breivold@mdh.se

Abstract

Software evolution is characterized by inevitable changes of software and increasing software complexities, which in turn may lead to huge costs unless rigorously taking into account change accommodations. This is in particular true for long-lived systems. For such systems, there is a need to address evolvability explicitly during the entire lifecycle, carry out software evolution efficiently and reliably, and prolong the productive life of the software systems.

Motivated by the need to understand software architecture evolution and to investigate ways to analyze software evolvability to support this evolution, the central theme of this research focuses on three particular aspects: (i) identify software characteristics that are necessary to constitute an evolvable software system; (ii) assess software evolvability in a systematic manner; and (iii) investigate how evolvability is addressed in open source software evolution. In this research, we have proposed the software architecture evolvability analysis process which provides several repeatable techniques for supporting software architecture evolution: (i) software evolvability model that provides a basis for analyzing and evaluating software evolvability, and a check point for evolvability evaluation and improvement; (ii) qualitative architecture-level evolvability analysis method that focuses on improving the capability of being able to understand and analyze systematically the impact of change stimuli on software architecture evolution; and (iii) quantitative evolvability analysis method that provides quantifications of stakeholders' evolvability concerns and potential architectural solutions' impacts on evolvability. Moreover, we also conducted a systematic review to obtain an overview of the existing studies in open source software evolution, with the intention to achieve an understanding of how software evolvability is addressed during development and evolution of open source software.

Table of Contents

1. Background and Motivation	3
2. Research Description	3
2.1 Purpose	3
2.2 Research Questions and Results	3
2.3 Methodology	5
2.4 Contribution	5
3. Thesis Contents	5
3.1 Introduction	6
3.2 Understanding Software Architecture and Evolution	6
3.3 Architecting for Software Evolvability	7
3.4 Analyzing Software Evolvability	7
3.5 Studying Proprietary Systems	7
3.6 Discussing Open Source Software Evolution	7
3.7 Summary and Conclusions	7
4. Related Work	8
5. Progress and Time Plan	9
6. Publications	10
6.1 Publications Fundamental to the Thesis Contributions	10
6.2 Publications Related to the Thesis	10
7. References	12

1. Background and Motivation

Software evolution is characterized by inevitable changes of software and increasing software complexities, which in turn may lead to huge costs unless rigorously taking into account change accommodations. This is in particular true for long-lived systems. For such systems, there is a need to address evolvability explicitly during the entire lifecycle, carry out software evolution efficiently and reliably, and prolong the productive life of the software systems. As software architecture holds a key to the possibility to implement changes in an efficient manner [1], software architecture evolution has become an integral part of the software lifecycle.

As software evolvability is a fundamental element for increasing strategic and economic value of the software [19], the need for greater systems evolvability is becoming recognized. We have observed this need from various cases in industrial context, where evolvability was identified as a very important quality attribute that must be maintained. Despite this need, there are no systematic means for systematically evaluating the evolvability of a software system. This leads to the reasons for this research: to find out what characteristics are necessary for a software system to be evolvable, how to assess and achieve software evolvability in a systematic manner.

2. Research Description

Lehman [16] describes two views on software evolution: the “*what and why*” versus the “*how*” perspectives. The former perspective studies the nature of software evolution phenomenon and investigates its driving factor and impact. The latter perspective studies the pragmatic aspects, i.e. technology, methods and tools that provide the means to control software evolution. In this research, we focus on the *how* perspective of software evolution. In addition, we refer to the definition of *Software Evolvability* in [18]:

“An attribute that bears on the ability of a system to accommodate changes in its requirements throughout the system’s lifespan with the least possible cost while maintaining architectural integrity”

2.1 Purpose

The purpose of this research is to build up an evolvability model and to investigate ways to systematically analyze and improve the ability to evolve software to the ever-changing requirements, and cope with stakeholders’ concerns with respect to business, technology, process and organizational perspectives. The focus of the research is primarily aimed at improving software evolvability for industrial software systems that often have a lifetime of 10-30 years. These systems are subject to and may undergo a substantial amount of evolutionary changes, e.g. software technology changes, system migration to product line architecture, ever-changing managerial issues such as demands for distributed development, and ever-changing business decisions driven by marketing, etc.

2.2 Research Questions and Results

This section presents the topic of the thesis in terms of research questions and summary of answers. The overall research question is how to analyze and improve software evolvability.

Software evolvability is a multifaceted quality attribute [18]. Before we can determine how to analyze software evolvability, we need to understand what subcharacteristics constitute the evolvability of a software system. To this end, we formulated the following research question which provides a starting point for further research:

What are subcharacteristics that are of primary importance for the evolvability of a software system?

(Q1)

Results based on our research: Based on the definition in [18], the software quality challenges and assessment [9], the types of change stimuli and evolution [7], and experiences we gained through industrial case studies, we have discovered that only having a collection of the subcharacteristics of maintainability as defined in the ISO software quality standard [12] is not sufficient for a software system to be evolvable. Therefore, we have (i) complimented and identified subcharacteristics that are of primary importance for an evolvable software system, and (ii) outlined a software evolvability model that provides a basis for analyzing and evaluating software evolvability.

The subcharacteristics that are of primary importance for the evolvability of a software system are: *Analyzability, Architectural Integrity, Changeability, Extensibility, Portability, Testability* and *Domain-specific Attributes*. A software evolvability model is outlined, with necessary subcharacteristics of software evolvability and corresponding measuring attributes identified. This model is established as a first step towards analyzing and quantifying evolvability, a base and check points for evolvability evaluation and improvement.

Additionally, the software evolvability model has been validated in two different industrial settings, i.e., an automation control system and mobile network node system.

Given this set of subcharacteristics of evolvability, the next question relates to the assessment of the evolvability subcharacteristics:

How to assess software evolvability of long-lived proprietary systems in a systematic manner? (Q2)

Results based on our research: A software evolvability analysis process has been proposed along with qualitative and quantitative software evolvability analysis methods.

- The qualitative architecture-level evolvability analysis method focuses on improving the capability of being able to understand and analyze systematically the impact of change stimuli on software architecture evolution;
- The quantitative evolvability analysis method provides quantifications of stakeholders' evolvability concerns and potential architectural solutions' impacts on evolvability.

These techniques have been validated through our participation in two large-scale industrial projects driven by the need of improving software evolvability. Based on our experiences, both the qualitative and quantitative analysis methods can be used as an integral part of software development and evolution process. Throughout the process of evolvability analysis at ABB, the architecture requirements and corresponding design decisions for the transition of architecture became more explicit, better founded and documented. The resulting analysis results were well accepted by the stakeholders involved in the analysis process, and became a blueprint for further implementation improvement. Throughout the process of evolvability analysis at Ericsson, the importance of various quality attributes perceived among different stakeholders was quantified and became more explicit. This quantification also served as a communication vehicle for further discussions among stakeholders. In both cases, by analyzing architectural improvement proposals with respect to their implications on evolvability subcharacteristics, we further avoided an ad hoc choice of potential evolution paths of software architecture.

So far, we have been focusing much on industrial software systems. On the other hand, with the emergence of the open source paradigm, researchers are also provided with a wealth of data for open source software (OSS) evolution analysis. Therefore, the next question relates to studying how evolvability is addressed in OSS evolution:

How is software evolvability addressed in the development and evolution of open source software? (Q3)

Results based on our research: A systematic review was conducted, with 41 primary studies identified. Based on the research topics of those studies, we have classified them into four main categories of themes: software trends and patterns, evolution process support, evolvability characteristics, and examining OSS at software architecture level. The first category is further refined into three sub-categories: software growth, software maintenance and evolution economics, and prediction of software evolution. A comprehensive overview of these categories, corresponding sub-categories and related studies is discussed. The main findings from this systematic review are:

- Regarding the category of software trends and patterns, most papers focus on using different metrics to analyze OSS evolution over time. Few papers have looked into the economic perspective, e.g., maintenance effort, and few papers utilize the historical evolution data for prediction of OSS evolution and development.
- Regarding the category of evolution process support, different aspects that appear to have impact on the OSS evolution process are covered; these aspect include commenting practice, OSS evolution and maintenance evaluation model, structures and quality characteristics of resources such as repositories, mails, bug tracking systems, as well as tools that support data retrieval for evolution analysis.
- Regarding the category of evolvability characteristics, determinism, understandability, modularity and complexity are addressed in the included studies. However, there are more evolvability characteristics that are not covered such as changeability, extensibility, testability, and modifiability. This might also explain the findings in the analysis of OSS evolution trends category that focuses on the evolution history instead of predicting the OSS evolution, because when there is a lack of analysis on OSS evolvability characteristics, it also becomes harder to predict its evolution.

- Regarding the category of examining OSS evolution at software architecture level, we have noticed from the review that few papers address OSS evolution at architectural level. Most papers address OSS evolution at source code level. However, software architectures are inevitably subject to evolution. Therefore, it is of major importance to put more focus on managing OSS evolution and assessing OSS evolvability at the software architecture level besides the code-level evolution.

2.3 Methodology

The methodology that has been used in the research consists of the following steps:

1. Analysis of the state-of-the-art and state-of-the-practice of the existing quality models for software evolution
2. Analysis of the state-of-the-art of the existing process models for software evolution
3. Based on the knowledge from the quality models and process models, a software evolvability model was outlined, with case studies performed to validate subcharacteristics of the evolvability of a software system
4. Systematic review of architecting for software evolvability
5. Case studies have been performed to assess software evolvability
6. Systematic review of the open source software evolution

Through the first two steps, a thorough investigation of the well-known quality models was made and the idea of a characterization of software architecture evolution was outlined. Besides, a characterization of a studied system was created in the third step. The third step includes also case studies with two development organizations from two different domains to address the issues with software architecture evolution.

A systematic review on architecting for software evolvability was performed in the fourth step with the intention to obtain an overview of the existing approaches in analyzing and improving software evolvability at architectural level.

Through the fifth step, the qualitative and quantitative evolvability analysis was validated in industrial settings.

In the last step, a systematic review was performed with the intention to obtain an overview of the existing studies in open source software evolution, and on how software evolvability is addressed during development and evolution of open source software.

2.4 Contribution

Motivated by the need to understand software architecture evolution and to investigate ways to analyze software evolvability to support this evolution, the central theme of this thesis focuses on three particular aspects: (i) identify software characteristics that are necessary to constitute an evolvable software system; (ii) assess evolvability in a systematic manner; and (iii) obtain an overview of the existing studies in open source software evolution in addition to our industrial studies.

The main contributions of the research include:

- *Software evolvability model* refines evolvability into a collection of subcharacteristics that can be measured through a number of measuring attributes, and is established as a first step towards analyzing and quantifying evolvability; This model provides a basis for analyzing and evaluating software evolvability, and a check point for evolvability evaluation and improvement;
- The *software architecture evolvability analysis (AREA) process* which provides several repeatable techniques for supporting software architecture evolution. These techniques have been validated through our participation in two industrial projects driven by the need of improving software evolvability.
 - *Qualitative architecture-level evolvability analysis method* focuses on improving the capability of being able to understand and analyze systematically the impact of change stimuli on software architecture evolution;
 - *Quantitative evolvability analysis method* provides quantifications of stakeholders' evolvability concerns and potential architectural solutions' impacts on evolvability.
- *A holistic overview of the existing studies on OSS evolution*, and find out how software evolvability is addressed during development and evolution of OSS.

3. Thesis Contents

The thesis will be written as a monograph. The following outlines the sections of the thesis:

1. Introduction
 - a. Problem statement
 - b. Scope
 - c. Research questions
 - d. Research methodology
 - e. Contributions
 - f. Thesis outline
2. Understanding Software Architecture and Evolution
 - a. Software architecture
 - b. Software evolution
 - c. Software quality models
 - d. Software process models
3. Architecting for Software Evolvability
 - a. Quality consideration during design
 - b. Quality evaluation at architectural level
 - c. Economic valuation
 - d. Architectural knowledge management
 - e. Modeling techniques
4. Analyzing Software Evolvability
 - a. Software evolvability model
 - b. Software evolvability analysis process
 - c. Qualitative software evolvability analysis
 - d. Quantitative software evolvability analysis
5. Studying Proprietary Systems
 - a. Study I – Qualitative Software Evolvability Analysis
 - b. Study II – Quantitative Software Evolvability Analysis
6. Discussing Open Source Software Evolution
 - a. OSS evolution trends and patterns
 - b. Evolution process support
 - c. Evolvability characteristics
 - d. Examining OSS evolution at software architecture level
7. Summary and Conclusions

3.1 Introduction

The introduction will present general descriptions of software evolution and describe the background and motivation to the research, including problem statement and research questions. A general discussion on methodology will also be included, and a more thorough description of the specific methods used for the different parts of the research as outlined in section 2.3 of this proposal. Thesis structure and contributions are stated as well.

3.2 Understanding Software Architecture and Evolution

This section presents relevant fields of research and practice.

3.3 Architecting for Software Evolvability

This section presents the results from a systematic review in architecting for software evolvability. The objective of this section is to describe an overview of the existing approaches in analyzing and improving software evolvability at architectural level, as well as their impacts on research and practice.

3.4 Analyzing Software Evolvability

This section will describe the software architecture evolution characterization, and propose an architecture evolvability analysis process that provides repeatable techniques for performing the activities to understand and support software architecture evolution. The activities are embedded in: (i) the application of a software evolvability model; (ii) a structured qualitative method for analyzing evolvability at the architectural level; and (iii) a quantitative evolvability analysis method with explicit and quantitative treatment of stakeholders' evolvability concerns and potential architectural solutions' impacts on evolvability.

3.5 Studying Proprietary Systems

This section describes the industrial case studies in which we applied the qualitative and quantitative software evolvability analysis methods at ABB and Ericsson.

3.6 Discussing Open Source Software Evolution

This section presents the results from a systematic review of open source software (OSS) evolution. The objective of this section is to describe an overview of the existing studies in open source software evolution, and how software evolvability is addressed during development and evolution of open source software.

3.7 Summary and Conclusions

This section will summarize the thesis and propose areas for future research.

4. Related Work

This section describes work that has been done related to software quality and software architecture evolution.

The seminal papers within software evolution are [13, 14], in which Lehman formulated eight laws of software evolution based on the observations of the IBM OS/360 operating system, and deliberately used the term software evolution to address the difference with the post-deployment activity of software maintenance.

Several process models have been proposed since the late seventies. For instance, Yau [20] proposed in the seventies the evolutionary process model with change mini-cycle, in which important new activities such as change impact analysis and change propagation are identified to accommodate the fact that software changes are rarely isolated. In the nineties, the term software evolution gained widespread acceptance. Several process models have emerged, e.g. Gilb's evolutionary development [10], Boehm's spiral model [3] and Bennett and Rajlich's staged model [2].

In quality models, quality attributes are decomposed into various factors, leading to various quality factor hierarchies. Some well-known quality models are McCall [17], Dromey [8], Boehm [4], ISO 9126 [12] and FURPS [11].

The foundation for any software system is its architecture, which allows or precludes nearly all of the quality attributes of the system. Accordingly, a spectrum of approaches with specific perspective or focus on a particular architecture-centric activity in software lifecycle has emerged. These approaches belong to five main categories of themes [6]:

- Quality considerations during design. The approaches in this category are further refined into three sub-categories: quality attribute requirement focused, quality attribute scenario focused, and influencing factor focused. Most of the techniques that support quality considerations during software architecture design help identify key quality attribute requirements early in the software design phase.
- Architectural quality evaluation. In the subsequent iteration when an architecture starts to take form, architectural quality evaluations help elicit and refine additional quality attribute requirements and scenarios. The approaches in this category are further refined into three sub-categories: experienced-based, scenario-based and metric-based.
- Economic valuation. Economic valuation approaches provide more details on architectural decisions' business consequences, and assist development teams in choosing among architectural options.
- Architectural knowledge management. Architectural knowledge management approaches improve architectural integrity by enriching architecture documentation with architectural knowledge captured from different information sources.
- Modeling techniques. Modeling techniques add value by modeling software artefacts along with their traceability, and visualizing corresponding impact of the evolution of software architecture artifacts.

With the emergence of the Open Source Software (OSS) paradigm, researchers have access to the code bases of a large number of evolving software systems along with their release histories and change logs. There have been a many studies published on OSS characteristics and evolution patterns by examining sequences of code versions or releases using statistical analysis. Meanwhile, the easily accessible data about different aspects of OSS projects also provides researchers with immense number of opportunities to validate the prior studies of proprietary software evolution [15] and to study how evolvability has been addressed in OSS evolution. The studies that are related to software evolution belong to different categories [5]:

- OSS evolution trends and patterns that focus on investigating OSS evolution trends and patterns. Based on their focus, the studies are further classified into three sub-categories: (i) software growth; (ii) software maintenance and evolution economics; and (iii) prediction of software evolution.
- OSS evolution process support that focus on OSS evolution support from various perspectives of software development process.
- Evolvability characteristics that focus on characteristics that can be considered important for software evolvability.
- Examining OSS evolution at software architecture level.

5. Progress and Time Plan

The progress of the current research is:

- **Publications.** Remaining publications include two journal papers – one is under revision (Journal of Information Software and Technology). The other one is to be submitted in March 2011. Notification to the second journal paper is expected to be in May 2011.
- **Courses.** I already fulfill the other main part required for the Ph.D. degree (75 credits for courses), namely completed courses with 76.5 credits as shown in the table below.

Courses	Credits	Status
UML, Object-oriented analysis and design with UML	4.5	Completed
Advanced C++ programming	1.5	Completed
COM and ActiveX	1.5	Completed
Software Architecture for industrial systems	4.5	Completed
ABB project design and analysis work	10.5	Completed
Research methodology	7.5	Completed
PROGRESS: techniques and technologies	7.5	Completed
Legacy issues in industrial software development	7.5	Completed
Advanced CBSE	7.5	Completed
Formal languages, automata and theory of computation	7.5	Completed
Research planning	4.5	Completed
Software architecture and process relation	7.5	Completed
Empirical software engineering research methodology	4.5	Completed
Total	76.5	

The goal is to have the Ph.D. defense in November 25, 2011. To complete the thesis, the main remaining activities are: (i) revise the IST paper; (ii) submit the second journal paper; (iii) convert relevant publications into chapters of the thesis; and (iv) write and compile the thesis.

A preliminary time plan is as follows:

- PhD. Thesis proposal presentation 2011-03-10
- Complete first draft of the thesis 2011-06-25
- Complete thesis and send to opponents 2011-08-25
- Finish camera-ready copy of the thesis 2011-10-15
- PhD. Defense 2011-11-25

Potential opponents and grading committee members are Patricia Lago, Claes Wohlin, Philippe Kruchten.

6. Publications

The publications are divided into two categories: (i) papers that are fundamental for the thesis contributions; and (ii) papers that are related to the thesis, but not directly to the thesis contribution.

6.1 Publications Fundamental to the Thesis Contributions

Journals

- Software Architecture Evolution through Evolvability Analysis
Hongyu Pei Breivold, Ivica Crnkovic, Magnus Larsson
(To be submitted to Elsevier SCICO - Special issue on Software Evolution, Adaptability and Maintenance, or Journal of Systems and Software)
- Architecting Software for Evolvability: A Systematic Review
Hongyu Pei Breivold, Ivica Crnkovic, Magnus Larsson
(Submitted to Journal of Information Software and Technology on 2010-10-26, accepted and is under revision)

Thesis

- Software architecture evolution and software evolvability, Hongyu Pei Breivold, Licentiate Thesis, Mälardalen University Press, January, 2009

Conferences and workshops

- A Systematic Review of Studies of Open Source Software Evolution
Hongyu Pei Breivold, Muhammad Aueef Chauhan, Muhammad Ali Babar
17th Asia Pacific Software Engineering Conference (APSEC), IEEE, Sydney, Australia, November, 2010
- A Systematic Review on Architecting for Software Evolvability
Hongyu Pei Breivold, Ivica Crnkovic
21st Australian Software Engineering Conference (ASWEC), IEEE, Auckland, New Zealand, April, 2010
- An Extended Quantitative Analysis Approach for Architecting Evolvable Software Systems
Hongyu Pei Breivold, Ivica Crnkovic
Computing Professionals Conference Workshop on Industrial Software Evolution and Maintenance Processes (WISEMP), IEEE, Montréal, Québec, Canada, April, 2010
- Analyzing Software Evolvability
Hongyu Pei Breivold, Ivica Crnkovic, Peter Eriksson
32nd IEEE International computer Software and Applications Conference (COMPSAC), Turku, Finland, July, 2008
- Analyzing Software Evolvability of an Industrial Automation Control System: A Case Study
Hongyu Pei Breivold, Ivica Crnkovic, Rikard Land, Magnus Larsson
3rd International Conference on Software Engineering Advances (ICSEA), IEEE, Sliema, Malta, October, 2008

6.2 Publications Related to the Thesis

Conferences and workshops

- What Does Research Say About Agile and Architecture?
Hongyu Pei Breivold, Daniel Sundmark, Peter Wallin, Stig Larsson
5th International Conference on Software Engineering Advances (ICSEA), IEEE, Nice, France, August, 2010
- Software Architecture Evolution – An Integrated Approach in Industry (Extended Abstract)
Hongyu Pei Breivold, Ivica Crnkovic

21st Australian Software Engineering Conference (ASWEC), IEEE, Auckland, New Zealand, April, 2010

- A Systematic Review of Software Evolvability
Hongyu Pei Breivold
Mälardalen University Workshop on Software Engineering, Västerås, Sweden, November, 2009
- Analysis of Software Evolvability in Quality Models
Hongyu Pei Breivold, Ivica Crnkovic
35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Software Process and Product Improvement (SPPI) Track, IEEE, Patras, Greece, August, 2009
- Migrating Industrial Systems towards Software Product Lines: Experiences and Observations through Case Studies
Hongyu Pei Breivold, Stig Larsson, Rikard Land
34th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Software Process and Product Improvement (SPPI) Track, IEEE, Parma, Italy, September, 2008
- Using Dependency Model to Support Software Architecture Evolution
Hongyu Pei Breivold, Ivica Crnkovic, Rikard Land, Stig Larsson
4th International ERCIM Workshop on Software Evolution and Evolvability (Evol'08), IEEE, L'Aquila, Italy, September, 2008
- Component-Based and Service-Oriented Software Engineering: Key Concepts and Principles
Hongyu Pei Breivold, Magnus Larsson
33rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Component-Based Software Engineering (CBSE) Track, IEEE, Lübeck, Germany, August, 2007
- Evaluating Software Evolvability
Hongyu Pei Breivold, Ivica Crnkovic, Peter Eriksson
7th Conference on Software Engineering and Practice in Sweden (SERPS), Göteborg, Sweden, October, 2007

MRTC Report

- A Survey of Software Architecture Evolvability
Hongyu Pei Breivold, Ivica Crnkovic
MRTC report ISSN 1404-3041 ISRN MDH-MRTC-239/2009-1-SE, Mälardalen Real-Time Research Center, Mälardalen University, September, 2009
- Using Software Evolvability Model for Evolvability Analysis
Hongyu Pei Breivold, Ivica Crnkovic
MRTC report ISSN 1404-3041 ISRN MDH-MRTC-222/2008-1-SE, Mälardalen Real-Time Research Center, Mälardalen University, February, 2008

Tutorial

- Emerging Technologies in Industrial Context – Component-Based and Service-Oriented Software Engineering
Ivica Crnkovic, Hongyu Pei Breivold
31st IEEE International computer Software and Applications Conference (COMPSAC), Beijing, China, July, 2007

7. References

- [1] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, ISBN 0-321-15495-9, Addison-Wesley Professional, 2003.
- [2] Bennett and Rajlich, Software maintenance and evolution: a roadmap, Conference on The Future of Software Engineering, ACM, 2000.
- [3] B. W. Boehm, A spiral model of software development and enhancement, *Computer*, vol. 21, pp. 61-72, 1988.
- [4] B. W. Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. J. MacLeod, and M. J. Merritt, Characteristics of software quality, North-Holland Publication Co., 1978.
- [5] H. P. Breivold, M. A. Chauhan, and M. A. Babar, A systematic review of studies of open source software evolution, Asia Pacific Software Engineering Conference, 2010.
- [6] H. P. Breivold and I. Crnkovic, A systematic review on architecting for software evolvability, Australian Software Engineering Conference, 2010.
- [7] N. Chapin, J. E. Hale, K. M. Khan, J. F. Ramil, and W. G. Tan, Types of software evolution and software maintenance, *Journal of Software Maintenance and Evolution Research and Practice*, vol. 13, pp. 3-30, 2001.
- [8] R. G. Dromey, Cornering the chimera, *IEEE Software*, vol. 13, pp. 33-43, 1996.
- [9] R. Fitzpatrick, P. Smith, and B. O'Shea, Software Quality Challenges, Proceedings of the Second Workshop on Software Quality at the 26th International Conference on Software Engineering, 2004.
- [10] T. Gilb, Evolutionary development [software], SIGSOFT Software Engineering Notes, vol. 6, p. 17, 1981.
- [11] R. B. Grady and D. L. Caswell, *Software metrics: establishing a company-wide program*. Prentice-Hall, Inc. Upper Saddle River, 1987.
- [12] ISO/IEC 9126-1, International Standard, Software Engineering. Product Quality – Part 1: Quality Model.
- [13] M. M. Lehman, On understanding laws, evolution, and conservation in the large-program life cycle, *Journal of Systems and Software*, vol. 1, pp. 213-221, 1980.
- [14] M. M. Lehman, Programs, life cycles, and laws of software evolution, *IEEE Software*, vol. 68, pp. 1060-1076, 2005.
- [15] M. M. Lehman, D. E. Perry, and J. F. Ramil, On evidence supporting the FEAST hypothesis and the laws of software evolution, 1998, pp. 84-88.
- [16] M. M. Lehman, J. F. Ramil, and G. Kahen, Evolution as a noun and evolution as a verb, SOCE 2000 Workshop on Software and Organisation Co-evolution, 2000.
- [17] J. A. McCall, P. K. Richards, G. F. Walters, *Factors in Software Quality*: NTIS, 1977.
- [18] D. Rowe, J. Leaney, and D. Lowe, Defining systems evolvability - a taxonomy of change, IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, 1994.
- [19] N. H. Weiderman, J. K. Bergey, D. B. Smith, and S. R. Tilley, Approaches to legacy system evolution, Carnegie Mellon University, Software Engineering Institute, 1997.
- [20] S. S. Yau, J. S. Collofello, and T. MacGregor, Ripple effect analysis of software maintenance, IEEE International computer Software and Applications Conference (COMPSAC), 2002.