Analysis of Software Evolvability in Quality Models

Hongyu Pei Breivold[1], Ivica Crnkovic[2]
[1]*ABB Corporate Research, Industrial Software Systems, 721 78 Västerås, Sweden*
hongyu.pei-breivold@se.abb.com
[2]*Mälardalen University, 721 23 Västerås, Sweden*
ivica.crnkovic@mdh.se

Abstract

For long-lived systems, there is a need to address evolvability explicitly. For this purpose, we have in our earlier work developed a software evolvability framework based on industrial case studies. With this as input in this paper we analyze several existing quality models for the purpose of evaluating how software evolvability is addressed in these models. The goal of the analysis is to investigate if the elements of the evolvability framework can be systematically managed or integrated into different existing quality models. Our conclusion is that although none of the existing quality models is dedicated to the analysis of software evolvability, we can enrich respective quality model through integrating the missing elements, and adapt each quality model for software evolvability analysis purpose.

1. Introduction

For long-lived industrial software, the largest part of lifecycle costs is concerned with the evolution of software to meet changing requirements [2]. In this context, software evolvability has been recognized as a fundamental element for increasing strategic decisions, characteristics, and economic value of the software [16]. It describes "*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*" [15]. We have also observed the need for greater system evolvability from various cases in industrial context [5, 7], where evolvability was identified as a very important quality attribute that must be maintained.

This paper first recaps briefly the software evolvability framework from our earlier work [5, 6]. Using this as input, we explore some well-known quality models in coping with software architecture evolution and analyzing software evolvability. Different quality models have been studied: McCall [12], Boehm [3], FURPS [9], ISO 9126 [10] and Dromey [8] in an attempt to identify in these models the aspects that are deemed important for software evolvability, as well as the aspects that are missing for managing software evolvability. The rest of this paper is structured as follows. Section 2 presents briefly our software evolvability framework. Section 3 describes the well-known quality models and discusses their relevance to software evolvability analysis. Section 4 concludes the paper.

2. Software Evolvability Framework

We have seen at ABB examples of different industrial systems that often have a lifetime of 10-30 years. These systems are subject to and may undergo a substantial amount of evolutionary changes. For such long-lived systems, there is a need to address evolvability explicitly during the entire lifecycle, and to prolong the productive lifetime of the software systems. We have in our earlier work [4-6], established a software evolvability framework illustrated in Figure 1.

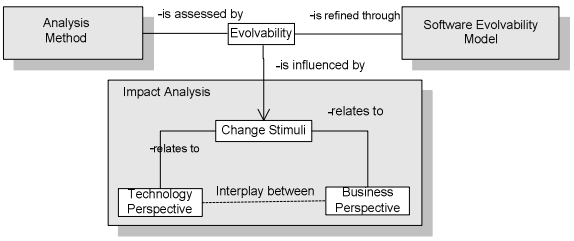

Figure 1. Software Evolvability Framework

The framework is used for software evolvability analysis through the establishment of a software evolvability model [5] and a method for assessing evolvability [6]. In addition, software evolvability is affected by a set of change stimuli. These stimuli are domain-dependent and have influence to different characteristics of evolvability. We characterize the stimuli in relation to technological and business perspectives; while technological perspective is mostly related to the system architecture, business-type of change stimuli can be very different and can significantly change the system requirements, and influence the technological stimuli.

2.1 Software Evolvability Model

We outline in [5] a software evolvability model that provides a basis for analyzing and evaluating software evolvability, as illustrated in Figure 2. This model regards software evolvability to be a multifaceted quality attribute [15], and refines software evolvability into a collection of subcharacteristics that can be measured through a number of corresponding measuring attributes.

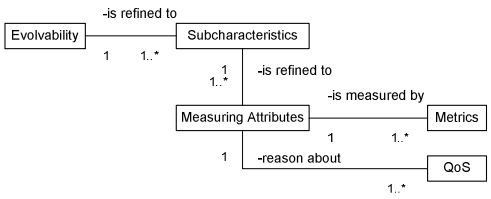

Figure 2. Software Evolvability Model

The evolvability model and identified evolvability subcharacteristics are the results from case studies [5, 6] and are valid for a class of long-lived industrial software-intensive systems that often are exposed to many, but in most cases evolutionary changes. Such systems are characterized by a number of dependability requirements (such as reliability, availability, possibly safety), compliance to different standards, complexity, and a combination of software and systems requirements. For these types of systems we have identified the following subcharacteristics: **Analyzability** describes the capability of the software system to enable the identification of influenced parts due to change stimuli; **Architectural Integrity** describes the non-occurrence of improper alteration of architectural information; **Changeability** describes the capability of the software system to enable a specified modification to be implemented and avoid unexpected effects; **Extensibility** describes the capability of the software system to enable the implementations of extensions to expand or enhance the system with new features; **Portability** describes the capability of the software system to be transferred from one environment to another; **Testability** describes the capability of the software system to validate the modified software; **Domain-specific Attributes** are the additional quality subcharacteristics that are required by specific domains.

3. Software Evolvability in Quality Models

In quality models, quality attributes are decomposed into various factors, leading to various quality factor hierarchies. Some well-known quality models are McCall [12], Dromey [8], Boehm [3], ISO 9126 [10] and FURPS [9]. These models are intended to evaluate the quality of software in general; none of them is specialized in or dedicated for evolvability analysis. It is thus likely that certain evolvability subcharacteristics are disregarded or not explicitly addressed in these models. We discuss each quality model regarding their relevance to the software evolvability analysis, and investigate if elements of the evolvability framework can be integrated into existing quality models.

3.1 McCall's Quality Model

McCall's quality model [12] defines and identifies the quality of a software product through addressing three perspectives: (i) *Product operation* is the product's ability to be quickly understood, operated and capable of providing the results required by the user. It covers correctness, reliability, efficiency, integrity and usability criteria. (ii) *Product revision* is the ability to undergo changes, including error correction and system adaptation. It covers maintainability, flexibility and testability criteria. (iii) *Product transition* is the adaptability to new environments, distributed processing together with rapidly changing hardware. It covers portability, reusability and interoperability criteria.

Not all the software evolvability subcharacteristics are explicitly addressed in this model. Analyzability is not explicitly included as one of the perceived aspects of quality. However, as the model is further detailed into a hierarchy of factors, criteria and metrics, some of the measurable properties and metrics are related to the achievement of analyzability, e.g. simplicity and modularity. Architectural integrity is not covered in the model. The integrity mentioned in the model describes the protection of the program from unauthorized access, and does not have the same essence as what we mean by architectural integrity. Moreover, none of the factors or quality criteria in the model is related to architectural integrity with respect to the understanding and coherence to the architectural decisions. This model is proposed for general application systems, and thus the domain-specific attributes are not explicitly addressed in the scope of the model.

3.2 Boehm's Quality Model

Boehm's quality model [3] represents a hierarchical structure of characteristics, each of which contributes to the total quality. The model begins with the software's general utility, i.e. the high level characteristics that represent basic high-level requirements of actual use. The general utility is refined into a set of factors and each factor is composed of several criteria which contribute to it in a structured manner. The factors include: (i) *portability*; (ii) *utility* which is further refined into reliability, efficiency and human engineering; and (iii) *maintainability* which is further refined into testability, understandability and modifiability.

Neither in the Boehm quality model is all the software evolvability subcharacteristics explicitly addressed. Analyzability is partially addressed through the characteristic *understandability*, which describes that the purpose of the code is clear to the inspector. However, none of the factors or measurable properties describes the capability to analyze the impact at the software architecture level due to a change stimulus. Architectural integrity is not covered in the model.

Extensibility is not perceived as an explicit quality aspect, but is instead aggregated within the scope of characteristic *modifiability*, which describes that the code facilitates the incorporation of changes, once the nature of the desired change has been determined. Domain-specific attributes are not explicitly addressed in the model.

3.3 FURPS Quality Model

The characteristics that are taken into consideration in FURPS model [9] are: (i) *Functionality* includes feature sets, capabilities and security; (ii) *Usability* includes human factors, consistency in the user interface, online and context-sensitive help, wizards, user documentation, and training materials; (iii) *Reliability* includes frequency and severity of failure, recoverability, predictability, accuracy, and mean time between failure (MTBF); (iv) *Performance* prescribes conditions on functional requirements such as speed, efficiency, availability, accuracy, throughput, response time, recovery time, and resource usage; (v) *Supportability* includes testability, extensibility, adaptability, maintainability, compatibility, configurability, serviceability, installability, and localizability/internationalization.

Architectural integrity is not covered in the model. None of the characteristics or subcharacteristics in the model is related to architectural integrity with respect to the understanding and coherence to the architectural decisions. Moreover, one disadvantage of this model is that it fails to take account of the software portability [13]. Domain-specific attributes are not addressed either in the model.

3.4 ISO 9126 Quality Model

ISO 9126 [10] specifies and evaluates the quality of a software product in terms of internal and external software qualities and their connection to attributes. The model follows the factor-criteria-metric model [12] and categorizes software quality attributes into six independent high-level quality characteristics: functionality, reliability, usability, efficiency, maintainability and portability. Each of these is broken down into secondary quality attributes, e.g. maintainability is refined into analyzability, changeability, stability, testability and compliance to standards, conventions or regulations.

The ISO 9126 quality model does not explicitly address all the software evolvability subcharacteristics. For instance, architectural integrity is not considered. None of the quality characteristics or subcharacteristics in the model is related to architectural integrity with respect to the understanding and coherence to the architectural decisions or strategies. Moreover, extensibility is not addressed as an explicit characteristic to represent future growths in this model. One may also argue if the enhancement-with-new-features type of change is embedded within the types of modifications defined in the quality model, i.e. corrections, improvements or adaptations of the software to changes in environment, requirements and functional specifications.

3.5 Dromey's Quality Model

Dromey [8] proposes a working framework for evaluating requirement determination, design and implementation phases. The framework consists of three models, i.e. *Requirement quality model*, *Design quality model* and *Implementation quality model*. The high-level product properties for the implementation quality model include: (i) *Correctness* evaluates if some basic principles are violated, with functionality and reliability as software quality attributes; (ii) *Internal* measures how well a component has been deployed according to its intended use, with maintainability, efficiency and reliability as software quality attributes; (iii) *Contextual* deals with the external influences on the use of a component, with software quality attributes in maintainability, reusability, portability and reliability; (iv) *Descriptive* measures the descriptiveness of a component, with software quality attributes in maintainability, reusability, portability and usability.

In this model, characteristics with regard to process maturity and reusability are more explicit in comparison with the other quality models. However, not all the evolvability subcharacteristics are explicitly addressed in this model. Analyzability is only partially covered within the *contextual* and *descriptive* product properties at individual component level, though none of these product properties describes the capability to analyze the impact at the software architecture level due to a change stimulus. Architectural integrity is not fully addressed despite the *design quality model* takes into account explicitly the early stages (analysis and design) of the development process. The focus of the *design quality model* is that a design must accurately satisfy the requirements, and be *understandable*, *adaptable* in terms of supporting changes and developed using a mature process. However, it is not sufficient for capturing architectural design decisions. Extensibility is not addressed as an explicit characteristic to represent future growths. Testability is implicitly embedded in the *internal* product property. Domain-specific attributes are not addressed. Moreover, one disadvantage of the Dromey model is associated with reliability and maintainability, as it is not feasible to judge them before the software system is actually operational in the production area [14].

3.6 Evolvability Analysis in Quality Models

None of the available quality models explicitly considers all the subcharacteristics of software evolvability. Table 1 summarizes the evolvability subcharacteristics coverage in each quality model, indicating if a certain evolvability subcharacteristic is partially, implicitly, explicitly addressed or not addressed at all in a quality model.

Table 1. Evolvability Coverage in Quality Models

Evolvability Subcharacteristics	McCall	Boehm	FURPS	ISO9126	Dromey
Analyzability	partially	partially	explicit	explicit	partially
Architectural Integrity	not present	not present	not present	not present	partially
Changeability	explicit	explicit	explicit	explicit	explicit
Extensibility	explicit	Implicit	explicit	implicit	implicit
Portability	explicit	explicit	not present	explicit	explicit
Testability	explicit	explicit	explicit	explicit	implicit
Domain-specific Attributes	not present	not present	not present	not present	not present

In general, we can claim that when using these quality models, additional attention should be given in the analysis of architectural integrity and domain-specific attributes. Most of the quality models, except the Dromey's quality model, are more driven towards the final coded software product, and do not take into account explicitly the analysis and design stage [11]. Thus, these models are not focused on the capability of capturing architectural design decisions for consistency of architectural integrity. Additionally, most of these models are generic models and are proposed for general application systems [1]. Thus, the domain-specific attributes are outside the scope of the quality models since they cannot be generalized. However, as various quality models cover a wide range of different quality characteristics, some of these quality characteristics might become domain-specific attributes in a certain context. Therefore, our claim is that domain-specific attributes need to be explicitly identified and considered in the evolvability analysis in relation to change stimuli. Changeability, portability and testability are addressed explicitly in most of the quality models. Extensibility is explicitly addressed in McCall and FURPS models, but not in the others.

By analyzing the evolvability subcharacteristics coverage in the quality models, we can recognize the subcharacteristics that are overlooked in a quality model. In this way, we can enrich respective quality model through integrating the missing elements, and adapt each quality model for software evolvability analysis purpose.

4. Conclusions

In order to address evolvability explicitly, we have, in our earlier work, developed a software evolvability framework based on industrial case studies. From the analysis of several quality models we can conclude that although none of these quality models is dedicated to the analysis of software evolvability, we can enrich respective quality model through integrating the missing elements, and adapt each quality model for software evolvability analysis purpose. This means that several evolvability subcharacteristics need to be explicitly considered, and that several additional system properties must be analyzed, e.g. architectural integrity and domain-specific attributes in relation to possible change stimuli.

References

[1] Arun, S., Rajesh, K., and Grover, P.S.: 'Estimation of quality for software components: an empirical approach', SIGSOFT Softw. Eng. Notes, 2008, 33, (6), pp. 1-10

[2] Bennett, K.: 'Software evolution: past, present and future', Information and Software Technology, 1996.

[3] Boehm, B.W., et al.: 'Characteristics of software quality' (North-Holland, 1978.)

[4] Breivold, H.P., and Crnkovic, I.: 'Using Software Evolvability Model for Evolvability Analysis', Mälardalen University, 2008.

[5] Breivold, H.P., Crnkovic, I., and Eriksson, P.J.: 'Analyzing Software Evolvability', COMPSAC 2008.

[6] Breivold, H.P., Crnkovic, I., Land, R., and Larsson, M.: 'Analyzing Software Evolvability of an Industrial Automation Control System: A Case Study', ICSEA2008.

[7] Christian, D.R.: 'Continuous evolution through software architecture evaluation: a case study', J. Softw. Maint. Evol.: Res. Pract, 2006, 18, pp. 351-383

[8] Dromey, R.G.: 'Cornering the Chimera', IEEE Software, 1996, 13, (1), pp. 33-43

[9] Grady, R.B., and Caswell, D.L.: 'Software metrics: establishing a company-wide program' (Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1987.)

[10] ISO9126: 'ISO/IEC 9126-1, International Standard, Software Engineering. Product Quality – Quality Model'

[11] Losavio, F., Chirinos, L., and Perez, M.A.: 'Quality models to design software architectures', Technology of Object-Oriented Languages and Systems 2001.

[12] McCall, J.A. et al.: 'Factors in Software Quality' (NTIS, 1977.)

[13] Ortega, M., Pérez, M., and Rojas, T.: 'Construction of a Systemic Quality Model for Evaluating a Software Product', Software Quality Journal, 2003.

[14] Rawashdeh, A., and Matalkah, B.: 'A New Software Quality Model for Evaluating COTS Components', Journal of Computer Science, 2006, 2, (4), pp. 373-381

[15] Rowe, D., Leaney, J., and Lowe, D.: 'Defining systems evolvability-a taxonomy of change', Change, 1994.

[16] Weiderman, N.H., Bergey, J.K., Smith, D.B., and Tilley, S.R.: 'Approaches to Legacy System Evolution', 1997.