An Extended Quantitative Analysis Approach for Architecting Evolvable Software Systems

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, i.e. a system's ability to easily accommodate changes, explicitly during the entire lifecycle. To improve the capability in being able to understand and analyze systematically software architecture evolution, we introduced in our earlier work a software evolvability model and a structured qualitative method for analyzing evolvability at the architectural level - the ARchitecture Evolvability Analysis (AREA) method. As architecture is influenced by system stakeholders representing different concerns and goals, the business and technical decisions that articulate the architecture tend to exhibit tradeoffs and need to be negotiated and resolved. To avoid intuitive choice of architectural solutions, we propose to extend the AREA method and strengthen its tradeoff analysis with explicit and quantitative treatment of stakeholders' prioritization of evolvability subcharacteristics and their preferences on design solutions. Finally, an example is used to illustrate the concept and applicability of the proposed approach.

1. Introduction

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. Software evolvability has thus been recognized as a fundamental element for increasing strategic and economic value of software [20], as it "*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*" [13]. This is in particular true for long-lived systems. For such systems, there is a need to address evolvability explicitly to prolong the productive life of the software systems. As software architecture holds a key to the possibility to implement changes in an efficient manner [3], software architecture evolution has become an integral part of software lifecycle.

To improve the capability in being able to understand and analyze systematically software architecture evolution, we introduced, in our earlier work, a software evolvability model [5], in which subcharacteristics of software evolvability and corresponding measuring attributes are identified. The evolvability model is a way to articulate subcharacteristics for an evolvable system that an architecture must support. In addition, we also introduced a structured method for analyzing evolvability at the architectural level, i.e. the ARchitecture Evolvability Analysis (AREA) method [6].

The business and technical decisions that articulate an architecture tend to exhibit tradeoffs that need to be negotiated and resolved. In AREA method, the tradeoff analysis is reflected in two constituent steps: (i) during architecture workshops, the stakeholders prioritize potential architectural requirements, which are mapped against evolvability subcharacteristics. By prioritizing the potential architectural requirements based on pre-defined criteria, evolvability subcharacteristics are implicitly prioritized by stakeholders; (ii) after the workshop, the identified architectural refactoring choices are qualitatively analyzed with respect to their impacts and support for evolvability subcharacteristics.

Based on our earlier experiences in evolvability analysis [4], we realize that designing and evolving a software architecture which satisfies a collection of evolvability subcharacteristics is a challenging task. This is mainly due to the fact that architecting for evolvable systems implies a complex decision-making process in which multiple attributes need to be taken into consideration, e.g. stakeholders' needs and goals, multiple quality requirements with competing priorities, various architectural refactoring choices with divergent implications on quality requirements. In AREA method, these multiple attributes and corresponding tradeoffs are treated qualitatively. To avoid intuitive prioritization of evolvability subcharacteristics and intuitive choice of architectural refactoring solutions, we introduce, in this paper, a quantitative extension to the AREA method with the aim to further strengthen its tradeoff analysis of potential architectural requirements, as well as preferences of architectural refactoring solutions in light of multiple evolvability subcharacteristics.

The remainder of this paper is structured as follows. Section 2 describes briefly the software evolvability model and the architecture evolvability analysis (AREA) method, and explains the motivations for extending the AREA method and strengthening its tradeoff analysis with quantitative analysis. Section 3 focuses on two constituent steps in the AREA method in which tradeoff analysis is involved, and presents the extended quantitative approach that would strengthen the tradeoff analysis. Section 4 illustrates the concept and applicability of the proposed approach with an example. Section 5 reviews related work. Section 6 concludes the paper.

2. Background and Motivation

This section describes the software evolvability model and evolvability analysis method, and motivates the need for an extended quantitative analysis for strengthening the tradeoff analysis in AREA method.

2.1 Software Evolvability Model

To improve the capability in being able to understand and analyze systematically software architecture evolution, we introduced a software evolvability model [5]. This model regards software evolvability to be a multifaceted quality attribute [13], and refines software evolvability into a collection of subcharacteristics that can be measured through a number of corresponding measuring attributes.

The evolvability model and identified evolvability subcharacteristics are the results from case studies [5-7] and are valid for a class of long-lived industrial software-intensive systems that often are exposed to many, and in most cases evolutionary changes. For this type 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.

2.2 Architecture Evolvability Analysis

We introduced in [6] a structured method for analyzing evolvability at the architectural level, i.e. the ARchitecture Evolvability Analysis (AREA) method. The evolvability analysis method starts with identification of change stimuli and guides architects through the analysis of potential architectural requirements that the software architecture needs to adapt to, and continues with identification of potential architecture refactoring solutions along with their implications. Through the analysis process, the implications of the potential improvement proposals and evolution path of the software architecture are analyzed with respect to evolvability subcharacteristics. Based on our experience in using the method [6], the result is that the architecture requirements, corresponding architectural decisions, rationale and architecture evolution path become more explicit, better founded and documented. The method consists of three phases as illustrated in Figure 1. The steps with gray color background comprise qualitative tradeoff analysis.

- Phase 1: Analyze the implications of change stimuli on software architecture. The outputs are identified and prioritized potential requirements on software architecture.
- Phase 2: Analyze and prepare the software architecture to accommodate change stimuli and potential future changes. This phase focuses on the identification of potential improvement proposals for the components that need to be refactored.
- Phase 3: Synthesize the previous results and finalize the evaluation.

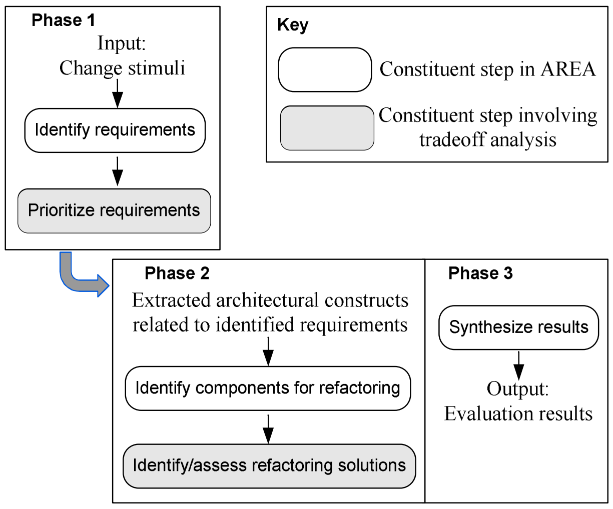
Phase 1
Input: Change stimuli
Identify requirements
Prioritize requirements
Key
Constituent step in AREA
Constituent step involving tradeoff analysis
Phase 2
Extracted architectural constructs related to identified requirements
Identify components for refactoring
Identify/assess refactoring solutions
Phase 3
Synthesize results
Output: Evaluation results

Figure 1. The Phases of ARchitecture Evolvability Analysis (AREA) Method

2.3 Motivations for Extending AREA Method with Quantitative Analysis

This section describes motivations for extending the AREA method and strengthening its existing qualitative tradeoff analysis with quantitative analysis.

2.3.1. Explicit stakeholders' views on prioritization and preferences on evolvability subcharacteristics. Depending on their roles that are involved in the development and evolution of a software system, the stakeholders usually have different concerns, i.e. interests which pertain to the system's development, its operation or evolution. Consequently, architecting for an evolvable software system implies that an architect needs to balance numerous stakeholders' concerns that are reflected in terms of their prioritization and preferences on evolvability subcharacteristics. In AREA method, this is treated implicitly in step *Prioritize requirements* in phase 1; i.e. potential architectural requirements are mapped against evolvability subcharacteristics to justify whether the realization of each requirement would lead to an improvement of any of the subcharacteristics. These potential architectural requirements are then prioritized based on predefined criteria. Consequently, the choice of prioritized architectural requirements implicitly sets priority ranking on evolvability subcharacteristics.

Software architecture is influenced by system stakeholders [3]. In circumstances when there are numerous roles of stakeholders, representing different and sometimes contradictory concerns and goals, explicit quantitative assessment of stakeholders' preferences on evolvability subcharacteristics will strengthen qualitative data and assist architects in making architectural design decisions. Otherwise, when the prioritization and preferences of evolvability subcharacteristics are not explicitly expressed by involved stakeholders, it becomes difficult to determine the dimensions along which a system is expected to evolve.

2.3.2. Quantification of refactoring solution alternatives' impacts on evolvability subcharacteristics. Choosing an architectural refactoring solution that satisfies evolvability requirements is vital to the evolution and success of a software system. Nonetheless, each solution is associated with multiple attributes, as the choice of component refactoring and/or implementation solution alternatives for fulfilling each architectural requirement may probably cause tradeoffs among evolvability subcharacteristics. Hence, it is important to understand how a refactoring alternative supports different evolvability subcharacteristics, especially when there are several refactoring alternatives to choose among, each of which exhibits varied support for evolvability subcharacteristics. Consequently, these alternatives need to be ranked, and meanwhile, can reflect stakeholders' preference information on evolvability subcharacteristics. In AREA method, the determination of potential refactoring solutions along with their impact on evolvability subcharacteristics is qualitatively handled in step *Identify/assess refactoring solutions* in phase 2, by examining the rationale of a solution proposal along with its architectural implications (positive or negative impact) of the deployment of the component on evolvability subcharacteristics.

Architects must often make architectural design decisions and give preference to a certain refactoring solution. In circumstances when there are multiple architectural alternatives to choose among, each of which exhibiting divergent impacts on evolvability subcharacteristics, a quantitative assessment of refactoring alternatives' impacts on evolvability subcharacteristics will guide and support architects to avoid making intuitive decisions in software architecture evolution.

3. Proposed Approach

In this section, we first clarify the notion of multiple attribute decision making process when evolving software architectures and we present briefly the Analytic Hierarchy Process (AHP) method which provides a basis for the extended quantitative analysis approach. This section will then detail the extended quantitative analysis which provides a structured way in quantitatively eliciting stakeholders' preferences for desired evolvability subcharacteristics and in obtaining quantitative understanding of the impacts of refactoring solutions on evolvability.

3.1 Multiple Attribute Decision Making Process

The proposed approach focuses on two constituent steps of the AREA method in which tradeoff analysis is concerned, i.e. step *Prioritize requirements* in phase 1, and step *Identify/assess refactoring solutions* in phase 2. These two steps entail subjective judgments with regard to preferences of architectural requirements, evolvability subcharacteristics, as well as choice of refactoring solutions. These subjective judgments constitute accordingly a multiple-attribute decision making process in architecting for evolvable software systems, as illustrated in Figure 2, i.e. stakeholders' preferences on evolvability subcharacteristics are determined by their different viewpoints, whereas the choice of architectural alternatives is dependent on their respective impacts on evolvability, and is meanwhile constrained by stakeholders' preference information on evolvability subcharacteristics.

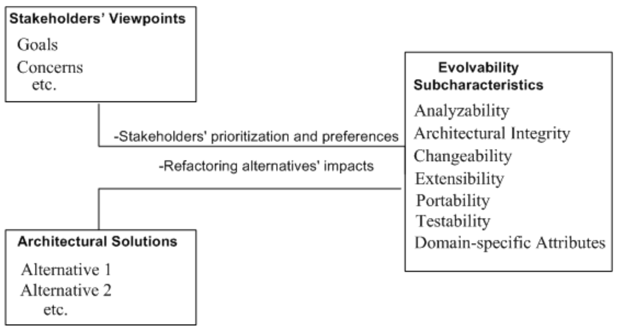

Figure 2. Multiple-Attribute Decision Making Process

3.2 Analytic Hierarchy Process

To obtain quantitative data with regard to stakeholders' preferences on evolvability subcharacteristics and refactoring alternatives' impacts on evolvability, we use Analytic Hierarchy Process (AHP) [14], because it is a multiple-attribute decision making method that enables quantification of subjective judgments. It makes relative assessments through pair-wise variable comparison and consists of five basic steps:

Step 1: Create an *n* x *n* matrix, in which *n* is the number of variables to be compared.
Step 2: Perform pair-wise comparison of the variables with respect to importance. The interpretation of the scales for comparison is shown in Table 1.

Table 1. Scale for Pair-wise Comparison

Scale	Explanation
1	Variable *i* and *j* are of equal importance
3	Variable *i* is slightly more important than *j*
5	Variable *i* is highly more important than *j*
7	Variable *i* is very highly more important than *j*
9	Variable *i* is extremely more important than *j*
2,4,6,8	Intermediate values for compromising between the other numbers

Step 3: Compute eigenvector of the *n* x *n* matrix. In this paper, we apply the '*averaging over normalized columns*' method [14] using the following equations:

a) Calculate sum of the columns;

$$n_j = \sum_{i=1}^{n} (n_{ij})$$

b) Divide each element in a column by the sum of the column, resulting in a new matrix;

$$m_{ij} = n_{ij}/n_j$$

c) Calculate sum of each row in the new matrix;

$$m_i = \sum_{j=1}^{n} (m_{ij})$$

d) Normalize the sum of rows to obtain priority vector *P* by dividing by *n*, which is the number of variables.

$$P = m_i/n$$

Step 4: Assign a relative importance to the variables, each accounts for a certain amount of percent of the importance of the variables.
Step 5: Evaluate consistency of subjective judgment.

3.3 Extended Quantitative Analysis

The application of the AHP method is described in the following subsections, detailing the extended quantitative analysis that is used to strengthen the two tradeoff analysis steps embodied in AREA method.

3.3.1 Stakeholders' prioritization and preferences of evolvability subcharacteristics. In this extension, stakeholders representing different roles provide their preferences on evolvability subcharacteristics by a pair-wise comparison of subcharacteristics (Q_i, Q_j) with respect to their relative importance. The AHP weighting scale shown in Table 1 is used to determine relative importance for each evolvability subcharacteristic pair. Note that the domain-specific attributes might comprise several additional quality characteristics that are required by a specific domain. Therefore, each of these domain-specific quality attributes is also included for pair-wise comparison together with the other evolvability subcharacteristics. The pair-wise comparison is conducted for all pairs, hence, *n*(*n*-1)/2 comparisons are made by each stakeholder role. Afterwards, for each stakeholder role, the aforementioned equations in AHP method are used to create a priority vector signifying the relative preference of evolvability subcharacteristics. As different stakeholder roles might have diversified preferences on evolvability subcharacteristics, for each evolvability subcharacteristic, we obtain normalized preference on an evolvability subcharacteristic by dividing sum of the preference of each stakeholder role by the number of roles.

The description below concretizes the calculation procedure, providing an example calculation of preferences of subcharacteristics aggregated from two stakeholders' perspectives. A matrix of pair-wise comparison is shown below, in which S_1 represents one stakeholder role, Q_1, Q_2 and Q_k are evolvability subcharacteristics, I_{ij} represents pair-wise comparison in terms of relative importance based on Table 1 (Note I_{ij} = 1 if *i* = *j*).

$\boldsymbol{S_1}$	$\boldsymbol{Q_1}$	$\boldsymbol{Q_2}$	... $\boldsymbol{Q_k}$
Q_1	I_{11}	I_{12}	
Q_2	I_{21}	I_{22}	
…			
Q_k	I_{k1}	I_{k2}	I_{kk}

Then by applying equation a), we get the sum of the columns:

$$n_j = \sum_{i=1}^{k} (n_{ij})$$

By applying equation b), we get the following new matrix:

$$m_{ij} = n_{ij}/n_j$$

Then by applying equations c) and d), we get normalized preference weight information of subcharacteristic Q_i from stakeholder S_1 perspective as shown in equation 1) below:

$$PQ_{is1} = \frac{\sum_{j=1}^{k}(m_{ij})}{k} \quad (i \text{ is an integral and } 1 \leq i \leq k) \quad \textbf{1)}$$

Likewise, the values indicating the preference weights of subcharacteristics ($Q_{1,} Q_{2,} \ldots Q_k$) from stakeholder S_2 perspective are calculated. We designate them as PQ_{1s2}, $PQ_{2s2,} \ldots PQ_{ks2}$.

Given that the preference consistency is correct, the overall stakeholders' preferences and prioritization of subcharacteristics are calculated by aggregating the preferences from the two stakeholders S_1 and S_2 as shown below, in which PQ_1, $PQ_\mathbf{2}$ and $PQ_\mathbf{k}$ indicate respectively the overall preferences on the specific subcharacteristics aggregated from the two stakeholders:

$PQ_1 = (PQ_{1s1} + PQ_{1s2})/2$

$PQ_\mathbf{2} = (PQ_{2s1} + PQ_{2s2})/2$

…

$PQ_\mathbf{k} = (PQ_{ks1} + PQ_{ks2})/2$

Generalizing the above results, the overall preference weight on subcharacteristic Q_i aggregated from *n* number of stakeholders is shown in equation 2) below:

$$PQ_\mathbf{i} = \frac{\sum_{j=1}^{n}(PQ_{isj})}{n} \quad (i \text{ is an integral and } 1 \leq i \leq k) \quad \textbf{2)}$$

3.3.2 Refactoring alternatives' impacts on evolvability subcharacteristics. In this extension, system architects or main technical responsible persons provide their judgment on how well each refactoring alternative supports different evolvability subcharacteristics. This is firstly done by a pair-wise comparison of the refactoring alternatives (Alt_i, Alt_j) with respect to a certain evolvability subcharacteristic, using the weighting scale in Table 1. Next, for each evolvability subcharacteristic, the aforementioned equations in AHP method are used to create a priority vector signifying the relative weight of how well different refactoring alternatives support a specific evolvability subcharacteristic. Afterwards, recalling the overall weights, i.e. stakeholders' preference weight of evolvability subcharacteristics (as described in the previous subsection) and the weight of how well different refactoring alternatives support a specific evolvability subcharacteristic, we can obtain a normalized value, designating the overall weight for each refactoring alternative's support on evolvability in general.

The following description concretizes the calculation procedure, providing an example calculation of two refactoring alternatives' overall support on software evolvability. A matrix of pair-wise comparison is shown below, in which Q_1 represents one of the evolvability subcharacteristics, Alt_1 and Alt_2 are two architectural alternatives, $S_\mathbf{ij}$ represents pair-wise comparison (based on Table 1, $S_\mathbf{ij}$ = 1 if *i* = *j*) in terms of relative support of each alternative on a certain subcharacteristic such as Q_1 as shown below:

$\mathbf{Q_1}$	$\mathbf{Alt_1}$	$\mathbf{Alt_2}$
Alt_1	$S_\mathbf{11}$	$S_\mathbf{12}$
Alt_2	$S_\mathbf{21}$	$S_\mathbf{22}$

Then by applying equations a) and b), we get the following matrix:

$\mathbf{Q_1}$	$\mathbf{Alt_1}$	$\mathbf{Alt_2}$
Alt_1	$n_{11} = S_\mathbf{11}/(S_\mathbf{11} + S_\mathbf{21})$	$n_{12} = S_\mathbf{12}/(S_\mathbf{12} + S_\mathbf{22})$
Alt_2	$n_{21} = S_\mathbf{21}/(S_\mathbf{11} + S_\mathbf{21})$	$n_{22} = S_\mathbf{22}/(S_\mathbf{12} + S_\mathbf{22})$

Then by applying equations c) and d), we get normalized support rates of the two architectural alternatives with respect to Q_1 as shown below, in which $PAlt_{1q1}$ and $PAlt_{2q1}$ indicate impacts of the two alternatives on subcharacteristic Q_1, i.e. how well they respectively support Q_1.

$PAlt_{1q1} = (n_{11} + n_{12})/2$

$PAlt_{2q1} = (n_{21} + n_{22})/2$

Likewise, the values indicating how well the two alternatives support other subcharacteristics ($Q_2 \ldots Q_k$) are calculated. We designate them as $PAlt_{1q2}$, $PAlt_{2q2}$ … $PAlt_{2qk}$ and $PAlt_{2qk}$.

Then

	$\mathbf{Alt_1}$	$\mathbf{Alt_2}$
Q_1	$PAlt_{1q1}$	$PAlt_{2q1}$
Q_2	$PAlt_{1q2}$	$PAlt_{2q2}$
…		
Q_k	$PAlt_{1qk}$	$PAlt_{2qk}$

Given that judgment of architectural alternatives' support on subcharacteristics is consistent, the overall weights of the two alternatives' support on evolvability are calculated by aggregating the preferences of subcharacteristics from the previous quantitative analysis (i.e. PQ_1 and PQ_2 in the previous subsection) as shown below, in which W_{Alt1} and W_{Alt2} indicate respectively the overall weights of how well the two alternatives support evolvability:

$$W_{Alt1} = \sum_{i=1}^{k} \left(PQ_k \times PAlt_{1qk}\right)$$

$$W_{Alt2} = \sum_{i=1}^{k} \left(PQ_k \times PAlt_{2qk}\right)$$

Generalizing the above results to *m* architectural alternatives, alternative *m*'s support on evolvability is expressed in equation 3) as shown below:

$$W_{Altm} = \sum_{i=1}^{k} \left(PQ_k \times PAlt_{mqk}\right) \qquad \textbf{3)}$$

4. Example

In this section, we illustrate the method by using an example which is simplified but has realistic context [6], in which a complex industrial control system was analyzed and refactored to facilitate product line architecture migration, driven by the need to improve its evolvability. Within this setting, we examine a subsystem for inter-process communication (IPC), which includes mechanisms that allow communication between processes, such as remote procedure calls, message passing and shared data.

4.1 Stakeholders' Preferences on Evolvability Subcharacteristics

Suppose we have two involved stakeholder roles with different perspectives on evolvability subcharacteristics. The domain-specific attribute is performance, due to the fact that the software has critical real-time calculation demands. One of the stakeholders' preferences on subcharacteristics is expressed in Table 2.

Table 2. Evolvability Subcharacteristics Preference Weights from One Stakeholder's Perspective

Q1: Analyzability; Q2: Architectural Integrity; Q3: Changeability; Q4: Extensibility; Q5: Portability; Q6: Testability; Q7: Performance

	Q1	Q2	Q3	Q4	Q5	Q6	Q7
Q1	1	1	1/5	1/4	3	3	1/3
Q2	1	1	1/2	1/3	1	1	1/3
Q3	5	2	1	1	3	5	3
Q4	4	3	1	1	3	5	1
Q5	1/3	1	1/3	1/3	1	1	1/3
Q6	1/3	1	1/5	1/5	1	1	1/3
Q7	3	3	1/3	1	3	3	1

After performing calculations based on equation 1) as described in section 3.3.1, the values indicating subcharacteristic preference from stakeholder *S1* perspective are summarized below. The figures suggest that, from stakeholder *S1* perspective, the evolvability subcharacteristics are prioritized as (in declining order): changeability, extensibility, performance, analyzability, architectural integrity, portability and testability.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7
S1	0.097	0.078	0.281	0.238	0.065	0.055	0.187

Likewise, the second stakeholder's preferences on evolvability subcharacteristics are collected and calculated. The values are summarized below.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7
S2	0.114	0.105	0.178	0.316	0.088	0.058	0.180

We aggregate the stakeholders' preferences and prioritizations of evolvability subcharacteristics based on equation 2) as described in section 3.3.1, and the results are shown in Table 3.

Table 3. Aggregated Stakeholders' Preferences and Prioritizations of Evolvability Subcharacteristics

	Q1	Q2	Q3	Q4	Q5	Q6	Q7
S	0.106	0.092	0.230	0.277	0.077	0.057	0.184

The above aggregated values indicate that extensibility has the highest priority, followed by changeability, performance, analyzability, architectural integrity, portability and testability.

4.2 Architectural Alternatives' Impact on Evolvability Subcharacteristics

Three alternatives are considered for the IPC subsystem:

- ***Alt1****: Static allocation of connection slot*

All the slot names and slot IDs that are used are defined in a C header file in the system. The developers edit this file to register their slot name and slot ID, and recompile. Afterwards, both the slot name and slot ID are specified in the startup command file for thread creation.

- ***Alt2****: Dynamic allocation of connection slot*

All the slot names and IDs are defined and used without booking in any header file. The IPC connection is established dynamically, and a connection slot ID is returned when no predefined slot ID is given.

- ***Alt3****: Static allocation of connection slot for base software and dynamic allocation for application extensions*

The slot names and IDs that are used by the base software are defined in a C header file in the system. The base software developers edit this file to register their slot name and slot ID, and recompile. The slot IDs for application extension are not booked in the header file. The command attribute dynamic slot ID is used instead.

The three alternatives are rated with respect to how well each supports each evolvability subcharacteristic. From changeability perspective, an example of the pair-wise comparison of the three alternatives is shown in Table 4.

Table 4. Pair-wise Comparison of Alternatives with Respect to Their Support on Changeability

Changeability	Alt1	Alt2	Alt3
Alt1	1	1/5	1/9
Alt2	5	1	1/3
Alt3	9	3	1

After performing the AHP calculations, the values indicating the support weights of the three alternatives with respect to changeability are summarized below. The values indicate that, from changeability perspective, the choices of alternatives are prioritized as (in declining order): *Alt3*, *Alt2* and *Alt1*.

	Alt1	Alt2	Alt3
Changeability	0.064	0.267	0.669

Likewise, the values of how well the three alternatives support the other evolvability subcharacteristics are collected and calculated. The values are summarized in Table 5.

Table 5. Alternatives' Support on Evolvability Subcharacteristics

	Alt1	Alt2	Alt3
Analyzability	0.633	0.106	0.260
Integrity	0.333	0.333	0.333
Changeability	0.064	0.267	0.669
Extensibility	0.106	0.260	0.633
Portability	0.143	0.429	0.429
Testability	0.333	0.333	0.333
Performance	0.633	0.106	0.260

Consequently, considering the prioritization weights of evolvability subcharacteristics in Table 3, together with the values indicating each alternative's support on evolvability subcharacteristics shown in Table 5, the overall weight for *Alt1* is calculated based on equation 3) as:

$W_{Alt1} = 0.106 \times 0.633 + 0.092 \times 0.333 + 0.230 \times 0.064 + 0.277 \times 0.106 + 0.077 \times 0.143 + 0.057 \times 0.333 + 0.184 \times 0.633 = 0.288$

Likewise, $W_{Alt2} = 0.247$, $W_{Alt3} = 0.487$, which indicates that *Alt3* is the preferred alternative with respect to evolvability.

4.3 Analysis

As illustrated in the example, the extended analysis approach provides quantitative support in understanding subjective decision making which is influenced by multiple attributes, e.g. stakeholders' preferences and architectural refactoring alternatives' impacts on evolvability subcharacteristics. Through the relative importance measuring process, we gain an explicit view on how stakeholders prioritize numerous evolvability subcharacteristics, and on the rationale behind a choice of an architectural alternative. Thus, the quantitative extension provides decision support and helps to avoid intuitive prioritization of evolvability subcharacteristics and intuitive choice of refactoring solutions.

5. Related Work

Several researches focus on quantitative evaluation of software architecture. Adaptability Evaluation Method (AEM) [18] is an integral part of the Quality-driven Architecture Design and quality Analysis (QADA) methodology [11] with specialization in the adaptability aspect. AEM defines adaptability goals through capturing the adaptability requirements that will be subsequently considered in the architecture design. One feature of AEM is that it fills the gap between requirement engineering and architecture evaluation as it provides guidelines on how to model adaptability in architectural models, and qualitatively/quantitatively analyzes candidate architectures to ensure that adaptability requirements are met before system implementation. However, this method does not explicitly focus on evolvability.

The idea of using Analytical Hierarchy Process to quantitatively support architectural decisions has been described in several research studies; nonetheless, these studies do not focus on evolvability analysis. For instance, [16] introduces a method for architecture evaluation and selection to ensure that the selected software architecture is the most potential one for fulfilling a blend of quality attributes. This method uses AHP to support the comparison of candidate architectures in order to reach consensus among stakeholders. [17] describes another quantitative quality-driven design approach that applies AHP for architectural design process. It helps evaluate stakeholder quality preferences and design alternatives, utilize optimization techniques in order to determine the optimal combination of design alternatives. [19] introduces a method which uses scenarios from the Architecture Tradeoff Analysis Method (ATAM) [8] and analyzes them with Analytical Hierarchy Process for making decisions in evaluating different integration strategies.

Cost Benefit Analysis Method (CBAM) [12] is an architecture-centric economic modeling approach that helps to address the long-term benefits of a change and its complete product lifecycle implications. This method quantifies design decisions in terms of cost and benefits analysis to determine the level of uncertainty and decides how to prioritize changes to architecture, based on perceived difficulty and utility.

Software architecture decisions carry economic value in form of real options [2, 15]. Options offer flexibility and take into account architectural evolution over time. [9] incorporates the concept of architecture options into design in order to exploit the optimal degree of design flexibility and provide a quantitative means of optimizing system architecture. [10] hypothesizes that architectural patterns carry economic value in the form of real options, and proposes an approach that considers cost, value and alignment with business goals to support architectural evolution. This approach guides the selection of design patterns, elicitation of architecturally significant requirements, and valuation of architecture in terms of design decisions with multiple quality-attribute viewpoints. Another application of real options theory is described in [1], which provides insights into architectural flexibility and investment decisions related to the evolution of software systems. This approach examines a set of probable changes as well as their added value, e.g. accumulated savings through enduring the change without violating architectural integrity; supporting future growth; and capability of responding to competitive forces and changing market conditions. In this paper, instead of focusing on the values of each design decision, we focus on how well each design alternative supports evolvability subcharacteristics.

6. Summary and Future Work

This paper proposes and demonstrates a quantitative extension to strengthen the tradeoff analysis in the architecture evolvability analysis (AREA) method, which was developed in our earlier work and applied in a complex industrial context to assist software evolvability analysis. As architecture is influenced by system stakeholders representing different concerns and goals, the business and technical decisions that articulate the architecture tend to exhibit tradeoffs and need to be rigorously negotiated and resolved. The extended quantitative analysis provides a structured way in eliciting stakeholders' preferences for desired evolvability subcharacteristics and in obtaining quantitative understanding of the impacts of architectural refactoring solutions on evolvability. We have described the approach in detail and illustrated theoretically its use with nonetheless a realistic example. Compared to only using the AREA method and qualitatively analyzing software architecture evolvability, the quantitative extension strengthens the tradeoff analysis part in AREA method and provides a structured way for explicit reasoning around the tradeoffs among evolvability subcharacteristics, as well as explicit reasoning around choice of architectural alternatives.

In future work, we intend to continue working on the extended architecture evolvability analysis method by conducting industrial case studies to collect experiences and refine the approach.

7. References

[1] Bahsoon, R., and Emmerich, W.: 'Evaluating architectural stability with real options theory', IEEE Computer Society, ICSM 2004, pp. 443-447

[2] Baldwin, C.Y., and Clark, K.B.: 'Design rules: Volume 1: The power of modularity', MIT Press Cambridge, MA, 2000.

[3] Bass, L., Clements, P., and Kazman, R.: 'Software Architecture in Practice', Addison-Wesley Professional, 2003.

[4] Breivold, H.P., and Crnkovic, I.: 'Software Architecture Evolution – An Integrated Approach in Industry', accepted at ASWEC, 2010

[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', ICSEA 2008, pp. 205-213

[7] Breivold, H.P., Larsson, S., and Land, R.: 'Migrating Industrial Systems towards Software Product Lines: Experiences and Observations through Case Studies', Euromicro SEAA 2008, pp. 232-239

[8] Clements, P., Kazman, R., and Klein, M.: 'Evaluating software architectures: methods and case studies', Addison-Wesley, 2006.

[9] Engel, A., and Browning, T.R.: 'Designing systems for adaptability by means of architecture options', Systems Engineering, 2008, 11, (2)

[10] Ipek, O., Rick, K., and Mark, K.: 'Quality-Attribute Based Economic Valuation of Architectural Patterns', Proceedings of the First International Workshop on the Economics of Software and Computation 2007.

[11] Matinlassi, M.: 'Quality-driven software architecture model transformation', WICSA 2005.

[12] Rick, K., Jai, A., and Mark, K.: 'Quantifying the costs and benefits of architectural decisions', Proceedings of the 23rd International Conference on Software Engineering, Toronto, Ontario, Canada, 2001.

[13] Rowe, D., Leaney, J., and Lowe, D.: 'Defining systems evolvability-a taxonomy of change', ECBS 1998, pp. 541-545

[14] Saaty, T.L.: 'The analytical hierarchy process', McGraw-Hill, New York, 1980.

[15] Sullivan, K.J., Chalasani, P., Jha, S., and Sazawal, V.: 'Software design as an investment activity: a real options perspective', Real Options and Business Strategy: Applications to Decision Making, 1999, pp. 215–262

[16] Svahnberg, M., Wohlin, C., Lundberg, L., and Mattsson, M.: 'A quality-driven decision-support method for identifying software architecture candidates', International Journal of Software Engineering and Knowledge Engineering, 2003, 13, (5), pp. 547-573

[17] Tariq, A.-N., Ian, G., Muhammed Ali, B., Fethi, R., and Boualem, B.: 'A quality-driven systematic approach for architecting distributed software applications', Proceedings of the 27th international conference on Software engineering, St. Louis, MO, USA2005.

[18] Tarvainen, P.: 'Adaptability evaluation of software architectures; A case study', COMPSAC 2007, pp. 579-584

[19] Wallin, P., Froberg, J., and Axelsson, J.: 'Making Decisions in Integration of Automotive Software and Electronics: A method based on ATAM and AHP', Proceedings of the 4th International Workshop on Software Engineering for Automotive Systems, IEEE Computer Society, 2007.

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