A METHOD PROPOSAL FOR ARCHITECTURAL RELIABILITY EVALUATION

Anna Grimán, María Pérez, Luis E. Mendoza and Edumilis Méndez
Processes and Systems Department – LISI
Universidad Simón Bolívar, Caracas, Venezuela

Keywords: Evaluation method, Software reliability, Software architecture, Scenarios, Software quality.

Abstract: Software quality characteristics, such as reliability, maintainability, usability, portability, among others, are directly determined by software architecture and, in consequence, it constitutes a very important artifact to be evaluated as soon as a general design is obtained. This article proposes a method to estimate software reliability by evaluating software architecture. Our method combines the strengths of three evaluation methods: ATAM (Kazman et al, 2000), DUSA (Bosch, 2000) and AEM (Losavio et al., 2004) obtained by identifying the main features needed in reliability architectural evaluation and studying several architectural mechanisms which promote this quality characteristic. Based on these features and the advantages of the studied methods and mechanism, we established phases, activities, roles, inputs/outputs, and artifacts; and we constructed a feasible method which can be applied in any organization interested in improving its software construction process and product.

1 INTRODUCTION

According to (ISO 9126, 2000), Reliability is defined as *the system's capability to perform its functions within certain operative conditions within a specific time period*. The sub-characteristics of Reliability (maturity, fault tolerance, and recoverability) are globally associated to the software's architecture or to each component in particular. In addition there are certain metrics that help us quantify those sub-characteristics.
Currently, the demands of the critical and Real-Time systems are growing (Laprie, 1995). These systems in particular are used in ever more complex tasks, wherein errors can lead to catastrophic consequences. As a result, these systems must be more reliable, since they must be able to perform despite those errors. An interruption in the system's service may lead to critical and dire situations. On the other hand, many organizations which render less critical services also require systems with high capabilities to satisfy their clients (e.g. online banking). Consequently, Reliability becomes a key characteristic for different organizations (Laprie, 1995).

Additionally Reliability (as well as other quality characteristics), is directly promoted by the software's architecture. In this sense, there are different architectural quality evaluation methods. However, none of these methods addresses Reliability in depth. The purpose of this article is to propose an Architectural Reliability Evaluation Method (AREM). To that end we have studied in detail several existing evaluation methods: Architecture Trade-off Analysis Method -ATAM, (Kazman et al, 2000), Software Architecture Analysis Method –SAAM (Kazman et al., 1994), Cost Benefit Analysis Method - CBAM (Nord et al., 2004), Architecture Level Modifiability Analysis - ALMA (Bengtsson et al., 2004), Architectural Evaluation Method - AEM (Losavio et al., 2004), Software Architecture Comparison Method - SACAM (Stoermer et al., 2004), and Design and use of Software Architecture - DUSA (Bosch**,** 2000) as well as architectural mechanisms. After an in-depth feature analysis, we have used as basis ATAM (Kazman et al., 2000), DUSA (Bosch, 2000) and AEM (Losavio, 2004).
This way our proposal is grounded in a rigorous revision of the concepts related to software reliability evaluation, which allowed us to establish a set of phases, activities, roles, inputs/outputs, and

Grimán A., Pérez M., E. Mendoza L. and Méndez E. (2007).
A METHOD PROPOSAL FOR ARCHITECTURAL RELIABILITY EVALUATION.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - DISI*, pages 564-568
DOI: 10.5220/0002393205640568
Copyright © SciTePress

artifacts. We were also able to construct a feasible method. Section 2 shows ontology for software architecture reliability which served as our frame of reference. Section 3 presents the proposed model and finally in section 4 we present our conclusions and future research.

2 A CONCEPTUAL MODEL FOR RELIABILITY ARCHITECTURAL EVALUATION

Even though the bibliography refers to Reliability in a recurrent manner, there is no agreement as to the concepts related to that particular quality characteristic.

Consequently, before proposing a Reliability Evaluation Method, it was necessary to create a model to represent the involved concepts and their relationships. To specify the issues related to evaluating Reliability in software architectures ontology was proposed regarding those key concepts (Grimán et al., 2005).

Figure 1 shows the conceptual model of Reliability of software architecture as a set of related concepts (Grimán et al., 2005). It can be observed that there is a large number of conceptual relationships that, when considered, shall help to perform a much more systemic assessment of the architecture, which will translate in a much more objective and effective selection of the software architecture, ideal for the development of Information Systems (IS). Some of the concepts are shown in the Figure 1.

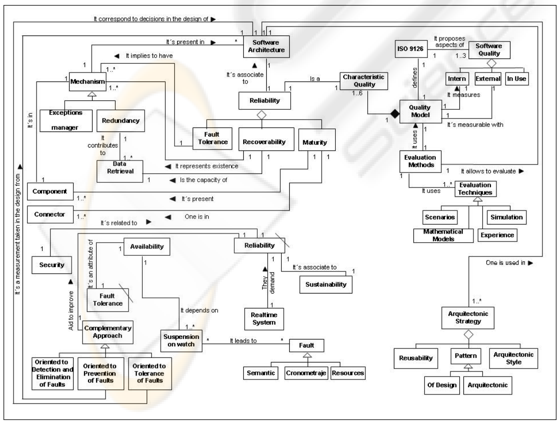


Figure 1: Conceptual Model for Software Reliability Evaluation (Grimán et. al, 2005).

Summarizing Figure 1, Reliability contains the following sub-characteristics: maturity, fault tolerance and recoverability. There are also three complementary approaches used to improve System Reliability (Sommerville, 2002): fault prevention, fault detection and elimination, and fault tolerance.

Only two of these three approaches correspond to measures that can be applied in the early stages of the development process; that is, during the architecture design, to increase a system reliability: *fault prevention and tolerance*, because decisions about the techniques to be used to prevent and tolerate faults in a system can be made during the architecture design, whereas decisions regarding techniques to be used to *detect and eliminate faults* in a timely fashion correspond to the stages of system testing.

The method we designed for the Architectural Reliability Evaluation (which was called AREM) is based on the quality model provided by the ISO/IEC 9126 international standard (as shown in Figure 1). The following section describes the proposed method in detail.

3 AREM PROPOSAL

AREM is oriented to determine if the architecture meets the initial Reliability requirements of the system. As described previously AREM is based upon the best practices within the existing methods used for reliability evaluation. In this section we describe the AREM's components.

3.1 Quality Specification Model

In order to specify the quality requirements, this method uses the quality model proposed by ISO/IEC 9126 (ISO/IEC, 2000). The functional and the quality requirements are refined into sub-characteristics which can be measured at an architectural level according to the standard, and that are addressed through a Utility Tree (UT) that will serve as a tool to specify Architectural Quality. UT is a scenario-oriented tool to evaluate quality characteristics. This way AREM is also a scenario-oriented evaluation method

3.2 Roles Involved

The roles involved in AREM are: *Requirement Engineer* who analyses the Reliability requirements of the system; and *Architect* who performs the architecture evaluation.

3.3 Phases

AREM consists of five (5) phases which involve a series of steps:

1) ***Initial Phase***: consists of the elicitation of the Reliability requirements, as well as the definition of the quality model which specifies the requirements and relates them to the quality attributes and the architecture presentation:.

- *Step 1.* Analyze the Reliability requirements of the system and then prioritize them.
- *Step 2.* Specify the quality model to be used for defining Reliability. Some of the metrics may be specified additionally according to specific components and/or connectors. We recommend to use the quality model proposed by the ISO/IEC 9126 standard.
- *Step 3.* Present the initial architecture.

2) ***Research and Analysis Phase:*** deals with the estimation of the key Reliability requirements and with the architectural focus.

- *Step 4.* Identifying the architectural approaches. These are identified but not yet analyzed. Also identified in this step are the design patterns and the architectural styles, patterns and mechanism.
- *Step 5.* Adapt the UT for Reliability, as well as, those quality sub-characteristics of the system which are related to Reliability (availability and security). These quality requirements are elicited, prioritized and specified in scenarios.

UT is a technique to transfer the goals of the quality characteristics of the system to Quality Scenarios that can be proven. It also helps to elicit a definition of the quality requirements of the system in a practical and operational way that can be understood by the stakeholders (Jones and Lattanze, 2001).

Scenarios, besides clarifying requirements, help to prioritize which parts of the architecture should be elicited in the first place (Kazman, 1999).

The Reliability analysis starts by considering several fault scenarios (see Table 1). In response, the architecture should manage the different types of faults: timing, semantic and system faults.

Table 1: Scenarios proposed.

SCENARIO
Scenario 1: A system suffers a software failure in normal operation and is reset.
Scenario 2: A power failure occurs in a system in normal operation and it is replaced.
Scenario 3: One of the servers fails in normal operation. Another server assumes operation.
Scenario 4: A failure occurs and the system notifies the user; the system can continue functioning in degraded mode.
Scenario 5: The demands for electronic FTP come to a site where the FTP server is low, the system is suspended for a period of time from the first failed demand and all resources are available while demands are suspended.
Scenario 6: A failure occurs and the system can interrupt its service for a determined period of time. This interruption is not measured versus the system availability unless it exceeds a well-defined interval.
Scenario 7: Demands for the electronic FTP come to a site where the FTP server is low; the system is suspended for a period of time from the first failed demand. The user with the failed site continues sending new orders every 10 minutes, the system queues the demands.
Scenario 8: A failure occurs during demand transaction in normal operation. The system recovers the demands before the failure. No demand should be lost as a result of the overload or failure of the system.
Scenario 9: Due to previous deliberate intrusions into the system, public data are transformed into private data and access is regulated in normal operation.
Scenario 10: In Normal operation, A failure occurs in a component of a critical system and it continues providing its services uninterruptedly.
Scenario 11: A mistake in the replication process results in a loss of synchronization of a transaction in the database with the backup of the database. The transaction is synchronized with the backup.
Scenario 12: A large number of customers need access to the server side object. The server has to deliver data within a determined response time.
Scenario 13: A large number of demands on an individual data entity come to the system from a user interface under normal conditions. The system has to transfer data within a determined period of time.
Scenario 14: An unexpected external message is received by a process during normal operation. The process informs the operator that the message has been received and continues operating without interrupting its services.

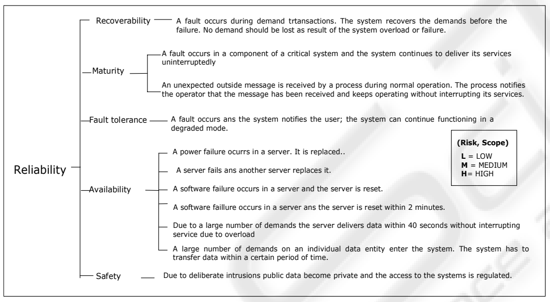
The description of the fourteen (14) Reliability scenarios proposed in this research consists of the six elements defined by Kazman (1999): stimulus, source, response, environment, stimulated artifact, and response measurement.An example of the scenario's components is shown next:
Scenario 7: Demands for the electronic FTP come to a site where the FTP server is low; the system is suspended for a period of time from the first failed demand. The user with the failed site continues

sending new orders every 10 minutes, the system queues the demands.

- *Stimulus:* The demands for the electronic FTP come to a site where the FTP server is low. The user with the failed site continues sending orders every 10 minutes.
- *Source of stimulus:* Internal.
- *Response:* Queue the demands.
- *Environment:* Normal operation.
- *Stimulated artifact:* System.
- *Response measurement:* Service suspension time.

Figure 2 shows an example of a UT proposed in AREM, which is obtained by adapting its components.

- *Step 6*. Analyze the architectural approaches. Based on the scenarios of higher priority identified in Step 5 we analyze the architectural approaches which direct these scenarios. The architectural patterns and styles, the design patterns and the previously identified mechanisms are analyzed. Potential risks, sensitive points and dependencies and interactions between the remaining quality characteristics and Reliability are identified (trade-offs.)



A fault occurs during demand trtansactions. The system recovers the demands before the failure. No demand should be lost as result of the system overload or failure.

A fault occurs in a component of a critical system and the system continues to deliver its services uninterruptedly

An unexpected outside message is received by a process during normal operation. The process notifies the operator that the message has been received and keeps operating without interrupting its services.

A fault occurs ans the system notifies the user; the system can continue functioning in a degraded mode.

A power failure ocurrs in a server. It is replaced..

A server fails ans another server replaces it.

A software failure occurs in a server and the server is reset.

A software faillure occurs in a server ans the server is reset within 2 minutes.

Due to a large number of demands the server delivers data within 40 seconds without interrupting service due to overload

A large number of demands on an individual data entity enter the system. The system has to transfer data within a certain period of time.

Due to deliberate intrusions public data become private and the access to the systems is regulated.

Figure 2: Example of a UT proposed in AREM (Grimán et al., 2005).

3) ***Trial Phase:*** the results of previous phases are compared to the specified Reliability requirements.

- *Step 7.* Analyze the results of Step 6 based on the ranges assigned to each scenario, in order to determine which attributes of the UT are inhibited, and perform a new iteration of the previous step to re-analyze the previously identified aspects. This is done in order to identify and document other architectural risks, sensitivity points, styles and trade-offs.

4) ***Transformation of the Architecture Phase:*** consists of improving the architecture so that the initial architecture is transformed into the ideal architecture following the specified Reliability requirements

- *Step 8.* Transform the selected architecture. This can be achieved by using the different architectural and design patterns and styles. The objective is to further improve Reliability by refining the architecture

5) ***Report Phase:*** This phase summarizes the results obtained during the previous phases

- *Step 9.* Present the results. Based on the information collected in the previous phases, the evaluator summarizes the results: final set of scenarios and their prioritization, Reliability UT and related quality characteristics, risks, sensitivity points and trade-offs.

The phases *Trial and Report* and *Research and Analysis* are based on ATAM's phases (Clements et al., 2002). The phase *Transformation of the Architecture* corresponds to the last phase of the DUSA (Bosch, 2000).

Table 2 shows the inputs and outputs for AREM's phases and steps, as well as the corresponding role.

Table 2: AREM's inputs/outputs.

Phases	Activities	Roles	Deliverables	Entering	Exits
Initial	1. Analyze the Reliability requirements of the system and prioritize them.	Requirements Engineer Architect	Prioritized Reliability Requirements	Requirements of Reliability of the System	Prioritized Reliability Requirements
	2. Specify the Quality Model to use in order to define Reliability	Architect	Specification the Quality Model	-	Specification the Quality Model
	3.Present the Initial Architecture	Architect	Initial Architecture documented	-	Initial Architecture Documented
Research and Analysis	4. Identify the architectural focuses, design patterns and architectural styles and patterns which promote or prevent Reliability	Architect	Architectural focuses, design patterns and architectural styles and patterns identified	Initial Architecture documented	Architectural focuses, design patterns and architectural styles and patterns identified
	5. Adapt the utility tree to Reliability and to the Quality characteristics related to Reliability.	Architect	Utility tree and set of scenarios	Specification of the Quality Model	Utility tree and set of scenarios.
	6. The architectural focuses, design patterns and architectural styles mechanisms, risks, points of sensitivity and Inter.-dependencies	Architect	Documenting of risks , points of sensitivity and Inter-dependencies. The architectural focuses are documented. Analysis of patterns, styles mechanisms identified	The architectural focuses. patterns , styles mechanisms	Documenting of risks, points of sensitivity and Inter-dependencies. The architectural focuses are documented. Analysis of patterns, styles mechanisms identified
Trials	7. Analyze the results obtained in Step 6 and perform a new iteration on the analysis of the architectural aspects identified (focuses, patterns, styles etc.).	Architect	Analysis of results	Documenting of risks , points of sensitivity and Inter-dependencies. The architectural focuses are documented. Analysis of patterns, styles mechanisms identified.	Analysis of results.
Transformation of the Architecture	8.Transform the selected Architecture	Architect	Documenting transformations achieved	Architecture initial documented	Documenting transformations achieved
Reports	9. Present Results	Architect	Final Results.	Requirements of Reliability of the System Specification of the Quality Model Document of risks and no risks. Points of sensitivity and Inter-dependencies The architectural focuses documented. Analysis of patterns, styles mechanisms identified.	Final Results

3.4 Discussion

As shown previously, AREM is a method which not only facilities the evaluation of the architecture but also promotes the design of a solution through the transformation of the architecture when there are inhibited attributes or characteristics. AREM then, can be used to evaluate architectures with different levels of maturity or detail, depending on the evaluator's objective.

Another important element of AREM is the use of questioning and measuring techniques (scenarios and metrics, respectively). This combines the exploratory potential of the scenarios with the quantitative way to represent the results.

On the other hand, the proposed roles do not suggest a specific number of people in a team. The role of *Requirement Engineer* can be represented by all the people in a team who are familiar with the requirements of the system. The role of *Architect* can be represented by those technicians involved in database, infrastructure, networks etc.

Finally, even though AREM focuses on Reliability evaluation, the architectural trade-offs obtained in step 6 are focused on other quality characteristics.

4 CONCLUSIONS

Architectural evaluation must be performed during early stages of the software's development. Nevertheless, during each stage of the development process, the metrics can identify potential problem areas that can undermine the fulfillment of the requirements (specifically those related to Reliability.) Finding those areas during the development stage reduces costs and prevents potential changes after the development cycle.

AREM entails rigorous usage, but also reports a number of advantages since it is based on the strengths of those methods which proved efficient when evaluating reliability.

In the future we expect to work on a combination of AREM and other methods, in order to provide a study of architectural trade-offs (such as Maintainability, Security and others).

REFERENCES

Bosch, J., 2000. *Design and Use of Software Architecture*. Addison Wesley. Harlow, England, 2000.

Bengtsson P., Lassing N., Bosch J. and Vliet H., 2004. *Architecture-level modi.ability analysis (ALMA)*. The Journal of Systems and Software 69 (2004) 129-147.

Clements, P., Kazman, R., and Klein, M., 2002. *Evaluating Software Architectures: Methods and Case Studies*. Addison Wesley.

Griman, A., Valdosera L., Mendoza, L., Pérez, M. & Méndez, E., 2005. *Issues for evaluating reliability in software architectures.* Proceedings of the Eleventh Americas Conference on Information Systems, Omaha, Nebraska, Estados Unidos de América, 2926-2931.

ISO/IEC 9126, 2000. *International Organization for Standardization ISO/IEC 9126 Software Engineering Product Quality*.

Jones, L., and Lattanze, A., 2001. *Using the Architecture Tradeoff Analysis Method to Evaluate Wargame Simulation System: A Case Study*. http://www.sei.cmu.edu/pub/documents/01.reports/pdf/01tn022.pdf

Kazman, R., Bass, L.,Abowd, G., and Webb, M.,1994, *SAAM: A Method for Analyzing the Properties of Software Architectures*. Proceedings of the Sixteenth International Conference on Software Engineering.

Kazman, R.,Klein, M., and Clements, P., 2000. *ATAM: Method for Architecture Evaluation*. TECHNICAL REPORT CMU/SEI-2000-TR-004. http://www.sei.cmu.edu/pub/documents/00.reports/pdf/00tr004.pdf

Kazman, R., 1999. *Using Scenarios in Architecture Evaluations*. http://www.sei.cmu.edu/news-at-sei/columns/the_architect/1999/June/Architect.jun99.pdf.

Laprie J.-C., 1995. *Dependable Computing: Concepts, Limits, Challenges.* Special Issue of the 25th International Symposium On Fault-Tolerant Computing. IEEE Computer Society Press. Pasadena, CA. pp. 42-54.

Losavio, F., Chirinos, L., Matteo, A., Le´vy, N., and Ramdane-Cherif, A., 839, 2004. *ISO quality standards for measuring architectures*. Journal of 840, Systems and Software 72 (2), 209–223.

Nord, R., Barbacci, M., Clements, P., Kazman, R.,Klein, M., O'Brien, L.,and Tomayko, J.,2004. *Integrating the Architecture Tradeoff Analysis Method (ATAM) with the Cost Benefit Analysis Method (CBAM)* (CMU/SEI-2003-TN-038). http://www.sei.cmu.edu/publications/documents/03.reports/03tn038/03tn038.html#chap04

Sommerville I., 2002. Ingeniería del Software, Editorial Addison. Wesley, Sexta Edición, México.

Stoermer, Ch., Bachmann, F., and Verhoef, Ch., 2003. SACAM: The Software Architecture Comparison Analysis Method. CMU/SEI-2003-TR-006 ESC-TR-2003-006. http://www.sei.cmu.edu/publications/documents/03.reports/03tr006.html