Self-Adaptive System Case-Study of Architecture-Based Software Reliability

João M. Franco
University of Coimbra, Portugal
jmfranco@dei.uc.pt

Abstract

In the last two decades different methods to assess reliability from an architecture-based system were proposed. Surveys and systematic reviews done in the last decade, summarize those methods and provide information about their applicability and suitability. However, those surveys and reviews also identify several shortcomings that today still exist in the architecture-based reliability research. The lack of studies applied to real world scenarios and the nonexistence of a common case-study used to compare different methods, are important and unsettle reliability shortcomings. As a result, they lead to a poor method validation and to several questions that still need to be answered.
We try to address some of these shortcomings by applying our work to a real case-study and compare our predicted reliability values with the ones extracted from the real system. In addition, we apply our method to a self-adaptive system to understand the evolution of an architecture which its major quality requirement is reliability.

1. Introduction and Related Work

Reliability has been widely considered, tested and several methods have been presented to address its prediction on the software architecture level. Several studies address the reliability assessment from a software architecture description [1–4]. Among the firsts to propose architecture reliability modeling using Markov chains was Cheung [5] and several surveys were presented since then [6–9]. According to these surveys, there are important shortcomings on the current state-of-the-art research which should be considered in future research:

- Lack of automated processes to assess and analyze the architecture;
- Lack of support to account with architectural styles (*e.g.*, fault-tolerant, parallelization or call-and-return) in reliability prediction;
- Assumption that reliability and usage profile values are known *a priori*;
- Poor method validation.

In previous studies [10, 11] we addressed the two firsts shortcomings by applying an automated reliability prediction and a sensitivity analysis from an architecture specification. Our approaches took into consideration several architectural features, such as the system usage profile, components' reliabilities and different architectural styles. The main goals were to predict reliability in an early phase, support architectural evolution and discard any manual activity regarding reliability prediction and analysis. As a result, our approaches guide architects on initial design decisions and help to avoid, prevent and detect undesired or infeasible architectural redesigns which could result in a loss in overall system reliability.

However, our work, as well as most of other studies on this topic, have some limitations which are reflected in the last two shortcomings. In particular, proposed methods are most often applied to made-up examples with a poor binding to real world scenarios. Reliability and usage profile values do not have real substantiation, in part because software architecture is still in its infancy and there are no public available architectures to test the proposed methods on real systems. The deficit of real world case-studies leads to a poor method validation in which authors are constrained to compare their results with the already accepted made-up examples or they have to pull up their sleeves and get a way of obtaining this information. Such examples of reliability studies applied to real world applications are Gokhale *et al.* [12] and Goseva-Popstojanova *et al.* [13]. The former applied their method to the SHARPE application and the latter applied to the C programming language. In both studies, reliability values were extracted using regression test suites from two different versions of the application and their respective software architectures were created by their authors for the purpose of the study.

This work intends to address the limitations identified above by applying different reliability prediction methods

to a case study. For this, we will use a real scenario with values extracted from the system which would let us compare the predicted reliability values with the real ones. In addition, the case-study will be under a self-adaptive system which will provide us answers to the unsettling research questions presented in Section 2.
Section 3 explains how our experimentation will be conducted and Section 4 concludes the paper.

2 Research Questions

The use of a self-adaptive system together with a real case-study in the architecture-based reliability prediction topic raises the following research questions:

- *Are the reliability prediction methods accurate enough?* Several reliability prediction methods have been presented [7], but after ten years the question still remains the same: Are they suitable to express the reliability from a real system? In this work we plan to perform a comparison between the predicted values with the real ones obtained from the deployed system.
- *Can the initial "guesstimated" values from expert knowledge be used to predict reliability in an early phase without any development has been done?* Several studies [1–4, 10, 14–17] address reliability prediction in an early phase of the software development lifecycle. However, there is the recurring problem of lack of information in such early phases, without any produced code or a deployed system is hard to obtain an estimative of the values. Some of them [14–17] address this problem with a guess-estimation of these values from the expert knowledge of architects and developers. But, are these values meaningful? We plan to contact two developers and the maintenance team of our case-study in order to obtain guess-estimated values, according to the defined failure behavior, for both reliability and usage profile and compare them to the ones obtained from the real system.
- *How the system will adapt itself over the time?* We expect that the system will face an initial period in which it will adapt itself in order to be the most reliable as possible. This adaptation process will be done by rerouting request to servers that are more reliable than others and by adding more servers. However, a highly-reliable server under a high load of requests may take more time to process those requests. Hence, if the response time of a request exceeds the reasonable time to be processed, it may be addressed as unsuccessful (although the response is correct) and automatically the reliability of that server will be decreased. As an illustration is the example of requesting a page to a web-server. If it takes an excessive time to load a web-page, the user may consider it unreasonable and he will probably be less confident on that server, considering it as unreliable.

3. Proposed Approach

In this work we will use as a case-study a web-based news provider called Znn.com. This case-study will be subject to adaptation by applying Rainbow [18], an architecture-based self-adaptive system. The following subsections present in a higher level of detail both the Znn.com and Rainbow.

3.1 Znn.com Scenario

Znn.com diagram is depicted in Figure 1 and it consists on a typical infrastructure for a news websites like CNN.com. Znn.com has a tiered architecture with a set of web-servers that serve content (graphical and textual) from backend databases to clients through a front-end presentation logic. In addition, Znn.com uses a load balancer to reroute the requests from the client to a pool of replicated servers. The number of available servers will depend on the amount of workload to be processed by the system.

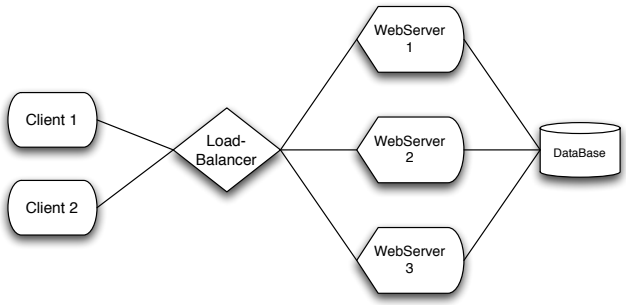

Figure 1: Znn Diagram

To perform reliability prediction and analysis in the system we need to extract a number of parameters from Znn.com. The required parameters are following detailed.

- *Reliability of every component in the system, except Clients*. The failure behavior adopted in this work will be the number of requests that are successfully served inside a time window of two seconds. The reliability value will be computed from the number of successful resolved requests over the total number of requests performed by the client.
- *Time that each request takes to be resolved.* We plan to introduce a timestamp in each request when they arrive to the system for reliability calculation purposes. After they have been processed by the web-servers, they

pass through the load-balancer which uses the current timestamp and the one attached to the packet to calculate the time that it took to be resolved.

- *System usage profile (also known as operational profile)*. In order to calculate an accurate reliability value we need to know how the system works and how many times a particular component is invoked. For this, we need to keep track of the system usage profile and, since this is a simple system, we only need to know the number of requests that are served between the different web-servers.

In order to extract those parameters from the system we will apply the method presented by Casanova *et al.* [19]. In particular, we obtain the system transactions which have the required information about what components were involved in the transaction and if it has failed or not.

3.2 Rainbow System

The Salehie *et al.* [20] presented a survey comparing sixteen different self-adaptive software projects that are currently available. Rainbow was identified as the one that addressed the highest number of self* properties: configuring, healing and optimizing. From the sixteen projects, only two use the software architecture to adapt and Rainbow is identified as the best between both. Hence, we will use Rainbow [18, 21], a software-architecture-based self-adaptation tool, to self-adapt our system. Our case-study is specified in an architecture which is described in the Acme language. Rainbow will parse the architecture, evaluate the system for reliability improvement and request adaptation by picking the most suitable strategy. The adaptation process will occur in distinct ways: by rerouting requests from the load-balancer to higher-reliable servers, by adding or removing web-servers depending on the requests demand made to the system and by applying software rejuvenation to web-servers if they present a high level of degradation.

4. Conclusions

In this paper we identified some research questions that are yet to be solved by the research community and which we devote our efforts to solve them. In addition, we present a new approach that combines a real case-study, which allows to extract real reliability and usage profile values, with a self-adaptive system that seeks for the best architectural strategy to obtain the highest system reliability.

We acknowledge that our approach still has some limitations regarding the statistical significance results obtained using a single case-study. One example is the verification of the usefulness of the "guesstimated" values in system reliability prediction in early phases. Since these values are obtained from human expert knowledge, they are independent from a case-study to another and we cannot make definitive conclusions. However, our study provides information if, in this particular system, the "guesstimated" values are close to the real ones.

With this study we intend to make a step forward to find a solution to the identified research questions and also, warning both practitioner and research communities to the shortcomings that still remain today.

References

[1] V. Cortellessa, H. Singh, and B. Cukic, "Early reliability assessment of uml based software models," in *Proceedings of the 3rd international workshop on Software and performance*, ser. WOSP '02. New York, NY, USA: ACM, 2002, pp. 302–309.

[2] S. M. Yacoub, B. Cukic, and H. H. Ammar, "Scenario-based reliability analysis of component-based software," in *Proceedings of the 10th International Symposium on Software Reliability Engineering*, ser. ISSRE '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 22–.

[3] F. Brosch, B. Buhnova, H. Koziolek, and R. Reussner, "Reliability prediction for fault-tolerant software architectures," in *Proceedings of the joint ACM SIGSOFT conferenceQoSA and ACM SIGSOFT symposiumISARCS on Quality of software architecturesQoSA and architecting critical systemsISARCS*. ACM, 2011, pp. 75–84.

[4] R. Reussner, "Reliability prediction for component-based software architectures," *Journal of Systems and Software*, vol. 66, no. 3, pp. 241–252, Jun. 2003.

[5] R. Cheung, "A user-oriented software reliability model," *IEEE Transactions on Software Engineering*, vol. 6, no. 2, pp. 118–125, 1980.

[6] A. Immonen and E. Niemelä, "Survey of reliability and availability prediction methods from the viewpoint of software architecture," *Software and Systems Modeling*, vol. 7, no. 1, pp. 49–65, Jan. 2008.

[7] K. Goševa-Popstojanova and K. Trivedi, "Architecture-based approach to reliability assessment of software systems," *Performance Evaluation*, vol. 45, no. 2, pp. 179–204, 2001.

[8] S. S. Gokhale, "Architecture-Based Software Reliability Analysis : Overview and Limitations," *IEEE*

Transactions On Dependable And Secure Computing, vol. 4, no. 1, pp. 32–40, 2007.

[9] D. Pengoria, S. Kumar, and M. S. Se, "A Study on Software Reliability Engineering Present Paradigms and its Future Considerations," *Computing*, 2009.

[10] J. Franco, R. Barbosa, and M. Zenha-Rela, "Automated reliability prediction from formal architectural descriptions," in *Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), 2012 Joint Working IEEE/IFIP Conference on*, aug. 2012, pp. 302 –309.

[11] J. M. Franco, R. Barbosa, and M. Z. Rela, "Reliability analysis of software architecture evolution." in *Latin-American Symposium on Dependable Computing (LADC) (to appear)*, 2013.

[12] S. Gokhale, W. Wong, K. Trivedi, and J. Horgan, "An analytical approach to architecture-based software reliability prediction," in *Computer Performance and Dependability Symposium, 1998. IPDS '98. Proceedings. IEEE International*, sep 1998, pp. 13 –22.

[13] K. Goseva-Popstojanova, M. Hamill, and R. Perugupalli, "Large Empirical Case Study of Architecture-Based Software Reliability," in *16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05)*. IEEE, 2005, pp. 43–52.

[14] L. Cheung, R. Roshandel, N. Medvidovic, and L. Golubchik, "Early prediction of software component reliability," in *Proceedings of the 30th international conference on Software engineering*. New York, New York, USA: ACM, 2008, pp. 111–120.

[15] S. Gokhale, "Analytical models for architecture-based software reliability prediction: A unification framework," *Reliability, IEEE Transactions on*, vol. 55, no. 4, pp. 578–590, 2006.

[16] S. Gokhale and K. Trivedi, "Reliability prediction and sensitivity analysis based on software architecture," in *13th International Symposium on Software Reliability Engineering, 2002. Proceedings.* IEEE Comput. Soc, 2002, pp. 64–75.

[17] H. Hellebro, "Architecture-based reliability modelling of software applications," Ph.D. dissertation, Royal Institute of Technology of Stockholm, Sweden, 2009.

[18] S. Cheng, *Rainbow: cost-effective software architecture-based self-adaptation*. Pittsburgh, PA, USA: Carnegie Mellon University, 2008, vol. 0389, no. May.

[19] P. Casanova, B. Schmerl, D. Garlan, and R. Abreu, "Architecture-based Run-time Fault Diagnosis," *Engineering*, no. September, pp. 13–16, 2011.

[20] M. Salehie and L. Tahvildari, "Self-Adaptive Software: Landscape and Research Challenges," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 4, no. 2, pp. 1–42, May 2009.

[21] S.-W. Cheng and D. Garlan, "Stitch: A language for architecture-based self-adaptation," *J. Syst. Softw.*, vol. 85, no. 12, pp. 2860–2875, Dec. 2012.