

A Framework for Experimental Error Propagation Analysis of Software Architecture Specifications¹

H. Ammar , D. Nassar,
W. AbdelMoez, M. Shereshevsky

Department of Computer Science and Electrical Engineering
West Virginia University, P O Box 6109
Morgantown WV 26506-6109
{ammar, dmnassar, rabie, smark}@csee.wvu.edu

Ali Mili

College of Computing Science
New Jersey Institute of Technology
Newark, NJ 07102
mili@cis.njit.edu

Abstract -- Early assessment of software quality attributes plays a central role in developing better quality software. Error propagation between software system components is a quantitative factor that reflects on the reliability of a software product. We introduce a framework for experimental error propagation analysis. This framework addresses the problem of estimating error propagation at the architecture design phase. Our approach is based on fault injection and post-simulation trace analysis. We compute error propagation estimates through comparison of faulty-run traces with a reference, fault-free, trace. We use this framework to experimentally study error propagation in a medium-sized real-time system. We believe that this framework can be further extended to allow for experimental analysis of change propagation and requirements propagation

1 Introduction

The software development process is shifting towards putting more emphasis on quality of architectural design. This shift is driven by the growing need to develop large and complex software systems in short time using COTS. Reliability of a software product is a key quality attribute. Highly reliable systems can be achieved by deploying trusted components. However, any claim of high reliability needs to be assessed using reliability analysis techniques. Error propagation is one of the factors that reflect on the reliability of the system as a whole [2].

Hiller *et. al* [3] proposed a model for analyzing propagation of data errors between inputs and outputs of a software systems. They define error permeability of a software module as the conditional probability of an error occurring at the module output given that an error occurred at its input. They use graph techniques to aggregate module permeabilities to system-level error propagation. To estimate permeability, they conduct experiments using fault injection techniques.

There are some attempts to analytically estimates quantitative factors using software metrics. Mili *et. al.*[2] suggest an information-theoretic approach to estimate quantitative factors. Researchers that develop these analytical studies often need to empirically validate their studies.

In this work, we are addressing the problem of creating an experimental framework to serve the validation effort of analytical error propagation analysis. We employ a fault-injection methodology[1] in designing a controlled experiment to experimentally estimate error propagation.

2 Experimental Error-Propagation Analysis

Our approach for experimental error propagation analysis is based on injecting faults, recording simulation traces, and then comparing the obtained “faulty-run” logs against simulation trace of a fault-free “golden-run”[3].

The fault-model provides a criterion based on which we alter the design specifications of components and/or connectors, such that faults of a certain class of (e.g. message swapping) are injected into the system.

We perform the error-propagation analysis in two phases: an acquisition phase and an analysis phase.

In the acquisition phase we extract architecture information about the components and connectors that make up the software system (e.g. the underlying finite state machines in the components, and the inter-component messages). We then use the criterion provided by the fault model to design the fault-injection experiments. In each experiment, we change some of the specifications of the corresponding architecture elements. Finally, we simulate the given system in presence of the injected fault and record the

¹ This work is supported in part by grants to WVU and NJIT from the National Science Foundation (2000-2003) under grant Number 0082574 and by the NASA Office of Safety and Mission Assurance (OSMA) Software Assurance Research Program (SARP) managed through the NASA Independent Verification and Validation (IV&V) Facility, Fairmont, West Virginia.

simulation traces for the different experiments. The faulty-run logs also contain markers that indicate when during the simulation faults are injected.

In the analysis phase, we conduct a post-processing comparison between the faulty-run logs and the reference fault-free log. Any encountered discrepancy is counted as an error propagation incidence.

We compute the experimental error propagation factor between components A and B as the percentage of the faults injected into component A that developed into errors in component A and propagated to component B.

3 A Controlled Experiment

We employed this framework in a controlled experiment for studying error propagation in a medium-sized real-time system, whose specifications are given in UML-RT [4]. We considered errors caused by message corruption over the connectors between components.

Figure 1 shows how we compare a faulty-run log (dashed line) to a reference log (solid line). The indices on the vertical axis correspond to the states of the receiving component, while the horizontal axis corresponds to the time elapsed during the simulation.

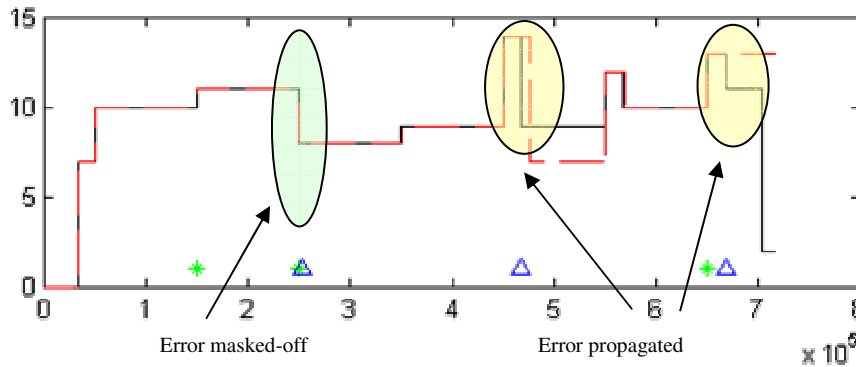


Figure 1: An illustration of log comparison.

The triangular marks correspond to fault injection markers and the star marks correspond to log-alignment markers to compensate for any timing skewes between faulty logs and the reference log. In this example three faults were injected into one of the system components, two of them caused error to propagate to the component whose state-transition log is plotted in .

4 Future Work

We intend to extend this framework and use it for change propagation analysis, and requirements propagation analysis. We are also looking at time-scaling techniques to speed-up simulations.

5 References

- [1] H. Ammar, et. al, "A Fault Model for Fault Injection Analysis of Dynamic UML Specifications", International Symposium on software Reliability Engineering, IEEE Computer Society, November 2001.
- [2] A. Mili, et. al, "Information theoretic metrics for software architectures". In Proceedings, COMPSAC 2001: Computer Software and Applications, Chicago, IL.
- [3] M. Hiller et. al, "An Approach for Analysing the Propagation of Data Errors in Software", In Proceedings-International-Conference-on-Dependable-Systems-and-Networks. 2001: 161-70, IEEE Comput. Soc, Los Alamitos, CA, USA
- [4] Rational Rose Real-Time: <http://www.rational.com/products/rosert/index.jtmpl>