AN EXPERIMENTAL EVALUATION OF SOFTWARE PERFORMANCE MODELING AND ANALYSIS TECHNIQUES

Julie A. Street and Robert G. Pettit IV
The Aerospace Corporation, 10549 Conference Center Dr, Chantilly, USA

Keywords: Unified Modeling Language (UML), Software Performance Modeling, UML SPT Profile, Petri nets.

Abstract: In many software development projects, performance requirements are not addressed until after the application is developed or deployed, resulting in costly changes to the software or the acquisition of expensive high-performance hardware. Many techniques exist for conducting performance modeling and analysis during the design phase; however, there is little information on their effectiveness. This paper presents an experiment that compared the predicted data from the UML Profile for Schedulability, Performance, and Time (SPT) paired with statistical simulation and coloured Petri nets (CPNs) for a sample implementation. We then discuss the results from applying these techniques.

1 INTRODUCTION

Software performance requirements are often not addressed until after applications are developed and sometimes until after they are deployed. Then the software has to change, which is very expensive and can impact the project's schedule, or high-performance hardware must be bought, which is also very expensive.

To remedy this, researchers have developed performance modeling and analysis techniques for addressing performance requirements early in the software lifecycle. There are many different approaches available; however, there is little empirical evidence to validate their effectiveness. This makes the choice of which technique to use very difficult.

The purpose of the experiment discussed in this paper is to develop empirical evidence to validate performance modeling and analysis techniques. This experiment compares the predicted performance data from two different performance modeling and analysis techniques with actual implementation data. If predicted data is close to the implementation data, the technique was validated in the context of the experiment. As more experiments are conducted, an empirical base on the validity of the different techniques can be developed. Once this empirical evidence exists, the choice of selecting a performance modeling and analysis technique becomes easier.

The structure of the paper is as follows: Section 2 discusses work related to this paper. Section 3 describes how the experiment was conducted. Section 4 presents the data and analysis. Finally, Section 5 provides the main conclusions.

2 RELATED WORK

Software performance modeling and analysis is a popular research topic; however, little work has been done specifically on empirical validation. Hooman *et al* validate their UML models of embedded systems and predict performance through coupled simulations of multidisciplinary tools (Hooman et al., 2004). Graf *et al* describe and demonstrate how their OMEGA-RT UML Profile can be implemented in a validation tool and applied to case studies (Graf et al.). Hakansson *et al* describe how they successfully conducted performance analysis on a case study (Hakansson et al., 2004). Bennett and Field describe how they use the UML SPT Profile in combination with LSTA, a discrete-event simulation tool, to analyze a telecommunications billing system (Bennett and Field, 2004). Woodside *et al* describe a tool architecture called PUMA, which interfaces between UML designs and analysis techniques (Woodside et al., 2005). This paper is different from these works because none of them are empirically based.

A. Street J. and G. Pettit IV R. (2007).
AN EXPERIMENTAL EVALUATION OF SOFTWARE PERFORMANCE MODELING AND ANALYSIS TECHNIQUES.
In *Proceedings of the Second International Conference on Software and Data Technologies - SE*, pages 417-420
DOI: 10.5220/0001336804170420

3 EXPERIMENT

The goal of this experiment was to compare the accuracy of two software performance modeling and analysis techniques. Our hypothesis is that the UML SPT Profile combined with a statistical simulation will produce more accurate performance data than coloured Petri nets (CPN) when compared to actual implementation data.

3.1 Case Study and Metrics

A real-time embedded distributed cruise control application described by Gomaa in (Gomaa, 2000) was the case study for this experiment. For practical reasons, only the auto control subsystem was used and the distributed aspects of the system were ignored. The auto control subsystem takes inputs from an aperiodic cruise control lever and a periodic auto sensors monitor, which polls the state of the brake and engine. The inputs are passed into the cruise control queue and processed by the cruise control in a first-in-first-out manner. Then the auto control subsystem sends messages to the car's throttle at a periodic rate.

The dependent variables (i.e. performance metrics) studied in this experiment were: maximum number of messages in the cruise control queue, number of messages in the cruise control queue over time, and throughput in terms of the number of messages processed per test case duration. To compare the performance analysis models' ability to predict performance under different circumstances, these dependent variables were collected on several tests. Each test was characterized by varying the following independent variables: cruise control lever input distribution model was varied between an aperiodic input (uniform distribution between given minimum and maximum interarrival times) and a bursty periodic input (periodic bursts with a set number of inputs at a set interarrival rate between messages and bursts); and the auto sensors periodic input rate was also varied.

3.2 Artifact Construction

After the case study and metrics were determined, the next step was to build the implementation, CPN model and UML SPT model. The implementation consisted of 27 Java classes and 1838 source lines of code (SLOC). SLOC was captured using Code Counter Pro (Geronesoft).

The CPN model for this experiment was built by another software engineer using Design/CPN (CPN group at the University of Aarhus, 2004). The CPN Editor was used to construct the small modules. Once all the individual modules were built, they were integrated into a larger architecture and the communication paths between modules were defined.

The UML SPT profile model was built by another software engineer. The SPT model was composed of a deployment diagram and multiple sequence diagrams. The deployment diagram captured the system's hardware architecture. The sequence diagrams outlined the processing steps required for each type of input. The UML SPT model was paired with statistical simulation as described by Minh in (Minh, 2001) for conducting analysis.

Validation was performed both the CPN and UML SPT models to ensure they were accurate representations of the implementation.

3.3 Data Analysis

This experiment compared accuracy, where accuracy is defined as the ability to produce maximum queue size and throughput values within a 60 percent deviation of the implementation's values. The deviation percentage is the difference between the predicted value and the implementation value divided by the implementation value. 60 percent was selected based on the median deviation percentage for all the tests. The median deviation percentage was 65 percent; therefore a 60 percent threshold ensured that an accurate test performed better than the median. For queue size over time, accuracy was determined based on an examination of the growth trend using Microsoft Excel.

3.4 Experimental Validity

Several measures were taken to ensure the data was valid. First, to ensure that there was no bias towards a particular performance model, two different software engineers constructed the models. Both engineers were provided with the same case study information and used the same predicted performance values in their analysis. Another concern for the SPT model was the selected analysis technique. The SPT model can be mapped to multiple analysis techniques. Therefore, it is possible that a different technique will produce different results.

When drawing conclusions from this experiment, it is important to note the scope of applicability. This experiment was conducted on only one application and therefore represents one data point. The results drawn from this experiment may or may not be applicable to other applications. Finally, at the time this experiment was conducted a

real-time environment was not available, and it is possible that a real-time environment may produce different results.

3.5 Data and Analysis

This experiment conducted nine different tests. The first set of tests, P1–P3, examined the models' ability to handle periodic inputs. On test P1, where the periodic input rate was 100ms, the SPT was accurate on maximum queue size, but exceeded the 60 percent deviation threshold on throughput, as shown in Table 1. The CPN accurately predicted a maximum queue size, but was inaccurate for throughput. When the periodic input rate was increased to 10ms on test P2, the SPT predicted an accurate maximum queue size, but was unable to accurately predict throughput. Both of the CPN's predictions were inaccurate. On the last test, P3, the periodic input rate was increased to 1ms, the SPT produced an accurate maximum queue size, while the CPN did not. However, both models predicted throughputs within the deviation threshold.

Table 1: Periodic Test Results.

	Max Msg in the Queue			Throughput		
Test	Imp.	SPT	CPN	Imp.	SPT	CPN
P1	1561	1951 (25%)	930 (-40%)	301	108 (-65%)	1668 (454%)
P2	2773	3665 (32%)	239 (-91%)	713	195 (-73%)	1667 (134%)
P3	9877	11151 (13%)	859 (-91%)	2031	888 (-56%)	1668 (-18%)

Next, to see if the predicted behavior of the queue matched the implementation for tests P1-P3, the queue size over time was plotted. On test P1 and P2 the implementation's queue began to backlog immediately and the SPT exhibited the same behavior. The CPN model predicted similar behavior, but it backlogged at a slower rate. Therefore the CPN was not accurate on P1 or P2. On test P3 both techniques failed to predict queue size over time.

The next set of tests, A1–A3, were used to assess the techniques' sensitivity to aperiodic inputs. On test A1, where the aperiodic input was set to occur between one and 100ms, the implementation had a small maximum queue size with a throughput around 400, as shown in Table 2. The CPN accurately predicted throughput, however it was just outside the deviation threshold for maximum queue size. The SPT model was accurate for throughput, however it was inaccurate maximum queue size.

On test A2, when the aperiodic rate was increased to occur between one and 50ms, the SPT failed to predict both metrics. The CPN produced a low maximum queue size, but gave an accurate prediction for throughput. On the last test, A3, when the aperiodic rate was increased to between one and five milliseconds, both techniques were unable to predict accurate results for throughput. The SPT was able to produce a maximum queue; however the CPN was not.

Table 2: Aperiodic Test Results.

	Max Msg in the Queue			Throughput		
Test	Imp.	SPT	CPN	Imp.	SPT	CPN
A1	6	221 (3583%)	2 (-67%)	398	183 (-54%)	314 (-20%)
A2	171	453 (165%)	2 (-99%)	403	143 (-65%)	484 (20%)
A3	3415	2642 (-23%)	873 (-74%)	333	102 (-69%)	1668 (401%)

Next, the queue size over time was plotted to see if the predicted behavior of the queue matched the implementation for tests A1–A3. For test A1, the implementation maintained a small queue size throughout the test. The CPN accurately predicted the same behavior, while the SPT did not. On test A2 and A3 both analysis techniques were inaccurate.

The final tests, B1–B3, were designed to assess the techniques' ability to predict performance based on periodic bursty traffic. A summary of the results is shown in Table 3. On test B1-B3, where the burst size was 20, 10, and 5 respectively, the SPT was accurate on all the tests for both metrics. The CPN model, however, was inaccurate on all the tests.

Table 3: Bursty Periodic Test Results.

	Max Msg in the Queue			Throughput		
Test	Imp.	SPT	CPN	Imp.	SPT	CPN
B1	2956	2988 (1%)	932 (-68%)	223	104 (53%)	1668 (-648%)
B2	1447	1780 (23%)	434 (-70%)	238	107 (55%)	1668 (-601%)
B3	683	1004 (47%)	2 (-100%)	276	118 (57%)	1102 (-299%)

Next, to see if the predicted behavior was accurate, the queue size over time was plotted. On B1 and B2, both the implementation and SPT backlogged and grew linearly. The SPT predicted accurate queue behavior. The CPN also produced a backlog; however, it grew at a slower rate than the implementation. Therefore the CPN did not predict accurate behavior on B1 and B2. Finally, on the last

test, B3, the implementation again backlogged and grew at a linear rate. The SPT had similar behavior, and therefore was considered accurate. The CPN predicted no backlog on this test.

4 CONCLUSION

In conclusion, the experimental results showed that the SPT combined with a statistical simulation was accurate the majority of the time. With respect to predicting maximum queue size, the SPT was accurate on 78 percent of the tests making it more accurate than the CPN, which was accurate on 11 percent of the tests. When the SPT approach was inaccurate, it consistently overestimated problems. On the CPN's inaccurate tests, it consistently underestimated performance problems.

The next metric examined was throughput. The SPT approach produced accurate values on 56 percent of the tests, while the CPN was accurate on 33 percent of the tests. Since the SPT approach was accurate on the majority of the tests, it was considered the more accurate technique. On the SPT's inaccurate tests, it constantly overestimated performance problems. On the CPN's inaccurate tests, it again underestimated the performance problems.

The last metric studied was the queue behavior over time. The SPT was accurate on 56 percent of the tests and the CPN was accurate on 11 percent of the tests. Therefore the SPT approach was considered the more accurate for predicting queue size over time. On its inaccurate tests, SPT consistently overestimated the problem or identified non-existent problems. Again, the CPN consistently underestimated or missed the performance problems.

In summary, the data suggests that the SPT Profile combined with statistical simulation is more accurate than CPNs, which supports the hypothesis. However, as discussed in section 3.4, the data is from a single experiment, and may not have broad applicability. It was also observed that the SPT consistently overestimated problems and identified non-existent problems. This would lead to unnecessary changes in design; however it guarantees that all existing problems would be fixed. On the other hand, the CPN consistently underestimated or missed performance problems. This would avoid unnecessary change in design, but not all problems would be fixed. Therefore it is better to err on the side of overestimation to ensure that all problems are fixed. This further suggests the use of SPT with statistical simulation over CPNs

REFERENCES

Bennett, A. J. & Field, A. J. (2004) Performance Engineering with the UML Profile for Schedulability, Performance and Time: a Case Study. IN IEEE (Ed.) *12th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2004).* Vollendam, The Netherlands.

CPN Group At The University of Aarhus, D. (2004) Design/CPN. 4.0 ed., CPN group at the University of Aarhus, Denmark.

GERONESOFT Code Counter Pro. 1.27 ed., Geronesoft.

Gomaa, H. (2000) *Designing Concurrent, Distributed, and Real-Time Applications with UML,* Boston, Addison-Wesley Object Technology Series.

Graf, S., Ober, I. & Ober, I. A real-time profile for UML Software Tools for Technology Transfer manuscript.

Hakansson, J., Mokrushin, L., Pettersson, P. & YI, W. (2004) An Analysis Tool for UML Models with SPT Annotations. *Workshop on SVERTS: Specification and Validation of UML models for Real Time and Embedded Systems.* Lisbon, Portugal.

Hooman, J., Mulyar, N. & Posta, L. (2004) Validating UML models of Embedded Systems by Coupling Tools. *Workshop on SVERTS: Specification and Validation of UML models for Real Time and Embedded Systems.* Lisbon, Portugal.

Minh, D. L. (2001) *Applied Probability Models,* Pacific Grove, CA, Brooks/Cole.

Woodside, M., Petriu, D. C., Petriu, D. B., Shen, H., Israr, T. & Merseguer, J. (2005) Performance by Unified Model Analysis (PUMA). *Fifth International Workshop on Software and Performance (WOSP 05).* Palma, Illes Balears, Spain.