Validation of Temporal Simulation Models of Complex Real-Time Systems

Farhang Nemati, Johan Kraft, and Christer Norström
Mälardalen Real-Time Research Centre, Mälardalen University, Västerås, Sweden
{farhang.nemati, johan.kraft, christer.norstrom}@mdh.se

Abstract

Model based analysis has the potential to facilitate maintenance of complex real-time systems, as it allows for impact analysis with respect to the systems' temporal behavior. Model based analysis of temporal behavior of a legacy real-time system has also the potential to support migration toward component based system. However, since most software systems today have been developed in a traditional, code oriented manner, sufficiently detailed models are typically not available. To apply model based analysis on these systems, models have to be extracted from their implementation and observed run-time behavior. This requires methods for model validation. The paper proposes a novel method for model validation and presents a framework for evaluation of model validation methods, which will be used to evaluate the proposed method. The method is targeting temporal models extracted from complex real-time systems.

1. Introduction

Large industrial software systems often have very high costs for maintenance. Such systems are generally very complex and have long lifecycles, during which they have been developed and maintained by many different developers, many no longer available. Design documentation is often incomplete and out-dated; we refer to systems with these characteristics as *legacy systems*.

Maintenance is the major cost during the lifecycle of large and complex systems. However in academia there has not been much attention to maintenance [9]. If a software system becomes hard to maintain due to increased complexity, it may be more cost-effective to redevelop the software from the beginning, instead of maintaining it further. This is not economically feasible for large legacy systems, which may contain millions of lines of code, involving hundreds of person-years in the development process.

Maintenance of systems with real-time requirements has additional challenges, as the changes may violate temporal requirements of the system. Examples of complex legacy systems with real-time requirements can be found in industrial products such as in industrial robotics, telecom systems and process automation systems. Often, the details of such legacy systems' timing behavior are not known, or it may be quite complex. Thus, the smallest increase in execution time can potentially cause a system failure.

Model based analysis of a legacy real-time system has the potential to facilitate migration toward a component based real-time system by analyzing the timing properties of the legacy code and wrapping it into components.

Model based analysis can also be used to analyze the impact of changes on a system's temporal behavior, before introducing the changes to the system; we refer to this as *impact analysis* [1]. Since analysis models of legacy systems usually don't exist, model extraction techniques are necessary to extract the analysis models from the observed behavior and the implementation of the system. Our aim is to extract analysis models from complex real-time systems automatically, as far as possible. Andersson et al in [1] present a semi-automated method that extracts models from complex real-time systems using system's source code and its recorded behavior during run-time. Huselius et al proposed a method [7] that automatically extracts temporal models from run-time recordings of complex real-time systems, containing task switch events and inter process communication (ipc) events. The recording is used as input to an offline tool that generates a model automatically. The extracted models are probabilistic [11] because the run-time observations can not reflect all details of the implementation. A discrete event simulator is used to analyze the extracted analysis models, e.g. for impact analysis.

To convince the system experts to use the simulation models, the models should reflect the system with a satisfactory level of significance; therefore an appropriate validation process should be performed before using the models.

The contribution of this paper is a novel method for validation of simulation models, describing temporal behavior of complex real-time systems, by comparing recordings from the system to corresponding recordings from simulation of the extracted model. The paper also presents a framework for evaluation of model validation methods, which is used to evaluate the proposed model validation method. This work was partially supported by the

Swedish Foundation for Strategic Research (SSF) via the strategic research centre (PROGRESS) at Mälardalen University.

2. Model Validation

Model validation is defined as "The process of determining whether a simulation model is an accurate representation of the system, for the particular objectives of the study" [10]. A model is an abstraction of the system, and details may be omitted from the model [10], for instance by probabilistic modeling. Thus, the results from a simulation of such models may not be identical to the recordings of the system, e.g., with respect to the exact execution times of the tasks. However, this kind of simulation model is supposed to be sufficiently accurate approximations of the actual system, not an exact representation of the system. The goal of a validation process is to show that the model reflects the system accurately enough to rely on for impact analysis.

To perform the model validation process, observations from the system and the predictions from the simulation model are compared under the same experimental conditions. There are various methods to do the comparison; these methods are either objective or subjective. Subjective methods are often used for validation of simulation models; examples of subjective methods are Face Validation, Graphical Comparisons, Hypothesis Validation, and Sensitivity Analysis [4]. They are however highly dependent on domain expertise and prone to human error. Objective methods use mathematical methods to compare the outputs from the real system to the outputs from the simulation model, for instance statistical tests such as the chi-square test, the Kolmogorov-Smirnov test, and the Mann-Whitney test [5]. However, statistical test have many assumptions which limits their applicability for model validation. For instance, the system property in focus may not be stationary, i.e., the distributions of output data changes over time. Moreover, statistical tests are only concerned with distributions and not the order of events.

A framework for validation of behavioral models extracted from complex real-time systems is presented in [3]. The authors present their notion of model equivalence based on observable property equivalence which is used to compare results of a model and an actual system. In the framework they identify distinct classes of response times or execution times from both the model and the system that match, and the results of the model and the system are compared. However the authors believe that for model validation comparing one single observation is not enough and multiple observations of the system should be used to increase the confidence in the extracted models.

A method in [6] is presented for automated validation of models extracted from real-time systems. Model checking is used to investigate if the model can generate the same event sequences as the recorded event sequences from the system.

The same author have also proposed a method to evaluate the quality of models extracted from real-time systems [8]; a method for comparing recorded distributions of timing properties, such as task response time. Such distributions are typically very complex and not suitable for comparison using traditional statistical tests targeting model validation.

Since our focus is validation of simulation models extracted from complex real-time systems, using only one method of these methods may not be enough to validate a simulation model. However, a collection of methods, each of which investigates a different aspect of the model, can increase the credibility of the model. The methods described in [3, 6, 8] are specifically used for validating temporal models extracted from legacy real-time systems.

3. Method

In this section we present a method for validating temporal models of real-time systems. The targeted properties are resource consumption properties such as usage of CPU time and usage of logical resources, such as message queues. Relying on one single technique for model validation will be risky in the sense of accepting an invalid model or to rejecting a valid model. Thus, we recommend to use the algorithm we present here as a complementary method with other appropriate model validation methods such as statistical tests and graphical methods.

In the algorithm, traces of resource consumption property, e.g. execution times of a task, from both the simulated model and the system are compared. Firstly a scenario that contains experimental conditions should be defined. The scenario should be the same for both the model and the system; because in the algorithm it is supposed that traces from both the model and the system start from the same state. A small deviation between start states may lead to wrong results. A solution can be comparing the traces from the model and the system immediately after a specific event, which is available in both traces and marks a suitable state where to begin the comparison, e.g. the start of a periodic task with high priority.

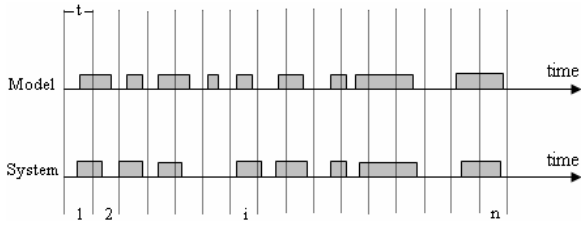

Figure 1: Traces of execution times from model and system are divided into time windows

In the method the traces of the resource consumption property, from the model and the system are sliced into *n* equal time windows (Figure 1). The precision of the comparison increases as *n* is

increased. However the maximum *n* is a number that slices the traces into the smallest time unit of the traces.

Definition 1: v = ValM (i)
, where v is the value of the property from the model within time window i.

Depending on the resource consumption property, the calculation of its value can be different e.g. *ValM(i)* of a message queue can be the maximum number of messages during the time-window, while ValM(i) for a task's CPU usage is typically percentage of the CPU time used by the task during the time-window.

Definition 2: v = ValS(i)
, where v is the value of the property from the system within time window i.

Definition 3:
Difference(i) = |ValM(i) – ValS(i)|

The function returns the absolute value of the difference between property values from the model and the system within time window i.

Definition 4: c = Credit(i, α)
, where c is either 1 or 0, depending on the following conditions:

Credit(i, α) = 1 if
Difference(i) $\leq \alpha$

Credit(i, α) = 0 if
Difference(i) $> \alpha \vee (ValS(i)=0 \wedge ValM(i) = 0)$

, where α is the tolerable boundary of difference between the execution times in the model and the system.

Since the model is an abstraction of the system and the model is probabilistic, the prediction values from the model are not expected to be exactly the same as the observations from the system, therefore the value of *α* should be provided so that a leeway is allowed for the model. The unit of *α* should be the same as the unit of value of the property e.g. if the execution times are compared, the unit of *α* should be time unit. The value that is assigned to *α* depends on the level of significance that we expect from the model; for smaller value of *α*, we get more accurate results, but the risk for rejecting a valid model increases. If the difference between corresponding property values within a time window is less than or equal to the boundary value, the property value from the model within the time window is considered as accurate. Thus *Credit(i, α) =1* means that the model in the time window *i* has enough accuracy. As the number of accurate time windows increases, the outputs of the model and the system are considered as more similar.

The property values of both the model and the system within some of the time windows may be 0; we refer to these time windows as empty time windows. Suppose the trace of execution times from a system and the corresponding model would look like Figure 2.

According to the Figure 2 it is obvious that the model does not reflect the system well and therefore we would expect the validation method to give the model a low credit, but the number of empty windows are relatively high and since the *Credit (i, α)* for theses time windows will be 1 (*Difference (i)=0*). To avoid empty windows affecting the result they should be removed from the calculation; therefore we should assign *Credit (i, α) = 0* for the empty windows.

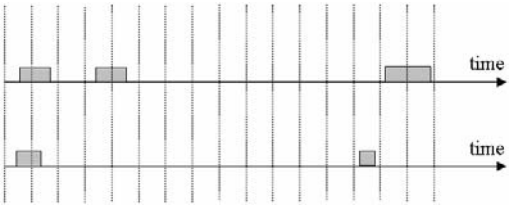

Figure 2

Definition 5:

$$P(\alpha, n) = \frac{\sum_{i=1}^{n} Credit(i, \alpha)}{n - k}$$

, where n is the total number of the time windows and k is the number of empty windows.

P has a value between 0 and 1. The closer P is to 1, the better is the model's approximation of system, with respect to the property in focus.

4. Evaluation Framework

We present a framework to evaluate model validation methods, including the method proposed in Section 3. This framework evaluates the methods for different types of systems with respect to a set of changes applied to the model specifications; we refer to the change set as *change scenarios*. The framework contains a two dimensional matrix as depicted in Figure 3. The matrix will contain the results (P values) of evaluation experiments for different types of systems and with respect to the change scenarios.

		Systems			
		System 1	Systems 2	…	System N
Change Scenarios	Scenario 1	P11	P12	…	P1N
	Scenario 2	P21	P22	…	P2N
	…	…	…	…	…
	Scenario M	PM1	PM2	…	PMN

Figure 3: The Framework Matrix to Evaluate Validation Methods

Example 1 represents the framework including the change scenarios [12] and two different types of systems:

Example 1:

Change scenarios:

- Case 0: No change at all,
- Case1: Add a "dummy task", without functionality, but with a short oscillating execution time and low priority,
- Case2: Raise the priority of the dummy task drastically,
- Case3: Increase the period time for the dummy task and extend its execution time.

System Types:

- A task model that contains both periodical and sporadic tasks, where the tasks communicate through message queues.
- A task model that only contains periodical tasks.

The validation results for each type of system with respect to the change scenario will be represented in a column of the matrix. Each property that is to be included in the model validation process is compared in a separate matrix.

5. Example

We have performed an evaluation of our model validation method by applying it to a system type which contains a task model with both periodical and sporadic tasks, where the tasks communicate through message queues. The task model is presented in the Table 1. The first output of the simulator was used as the output of the system, and then we changed the models regarding following change scenarios and compared the outputs against the original model to investigate the method.

- Case0: No change at all
- Case1: Change the behavior of existing tasks e.g. execution times
- Case2: Change the priority of existing tasks
- Case3: Change the period time of a task
- Case4: Add a new task called dummy with a short oscillating execution time and low priority
- Case5: Raise the priority of the dummy task drastically
- Case6: Increase the period time for the dummy task and extend its execution time.

Table 1: The Model

Task	Priority	Period (ms)
PLAN_TASK	5	40
CTRL_TASK	4 or 2	10 or 20
DRIVE_TASK	1	2
IO_TASK	3	5

The specifications of the model are as following:

- A task may trigger other tasks by using message queues,
- A task may be triggered by timers, events, or a combination of both,
- Depending on the state of the system, the temporal behavior of a task may change,
- Semaphores may block the tasks,
- Scheduling priority and period of the tasks may change,
- A task with a lower priority value is more significant (has a higher priority).

Execution times, communications, and other behaviors that affect timing behavior are described in the model.

5.1. Results

To implement the method we presented in the Section 3 we have developed a tool that compares the original model to the model regarding each change scenario, and outputs the respective P values. The number of time windows was 1000 and α was initiated to 5 milliseconds. For the evaluation we compared the execution time traces of CTRL_TASK task. The results are presented in the Table 2.

- Case1: P value is very low since IO_TASK has a higher priority than CTRL_TASK and changing the execution time of IO_TASK affects the execution times of CTRL_TASK
- Case2: When the priority of PLAN_TASK is decreased, P value shows that execution times trace of CTRL_TASK is not affected because the priority of PLAN_TASK is lower than the priority of CTRL_TASK, but when the priority of PLAN_TASK is set to a higher priority than the priority of CTRL_TASK, P value is very low showing that its execution times trace is affected quit much.

- Case3: P value shows that the execution times trace of CTRL_TASK is not affected by increasing the period of PLAN_TASK with a lower priority than CTRL_TASK, but the trace is affected drastically by increasing the period of DRIVE_TASK which has a higher priority than CTRL_TASK.

- Case 4: Adding a task (DUMMY_TASK) with a lower priority than CTRL_TASK doesn't influence execution times trace of CTRL_TASK.

- Case5: Increasing the priority of DUMMY_TASK to a value higher than the priority of CTRL_TASK influences the execution times trace of CTRL_TASK as P value shows.

- Case6: The results in the Case 6 show that increasing the period of DUMMY_TASK and extending its execution times don't affect the execution times trace of CTRL_TASK as long as its priority is lower than the priority of CTRL_TASK, but when its priority is more than the priority of CTRL_TASK the changes influence the trace.

Table 2: The Results

Change Scenarios	Changes	P (5, 1000)
Case0	-	1
Case1	IO_TASK: Before: C=23 Changed: C=46	0.38
Case2	PLAN_TASK: P=6 PLAN_TASK: P=2	1 0.18
Case3	PLAN_TASK: T=80 DRIVE_TASK: T=10	1 0.03
Case4	DUMMY_TASK: P=6, T=5, C=25 or 50	1
Case5	DUMMY_TASK: P=1, T=5, C=25 or 50	0.37
Case6	DUMMY_TASK: P=1, T=10, C=50 or 100 DUMMY_TASK: P=6, T=10, C=50 or 100	0.40 1

6. Conclusions and Future Work

In this paper we have presented a method to validate simulation models extracted from complex real-time systems. The method compares two recordings, one from the real system and one from a simulation model, with respect to resource consumption. The method can be used as a complementary method with appropriate statistical tests and subjective methods for model validation.

We have presented a framework to evaluate the proposed method. The framework evaluates the method for different types of systems with respect to a set of changes, representing common types of changes.

For our future work we plan to identify several different types of systems to evaluate the proposed method and using the presented framework. In the future work we also plan to develop a framework for validation of simulation models extracted from complex real-time systems. The framework will contain our proposed method and other applicable methods such as statistical tests (e.g. Kolmogorov-Smirnov test) and subjective methods (e.g. Graphical Comparisons).

References

[1] J. Andersson, J. Huselius, C. Norström, and A. Wall. Extracting Simulation Models from Complex Embedded Real-Time Systems. In *Proceedings of the 2006 International Conference on Software Engineering Advances ICSEA'06, IEEE,* Tahiti, French Polynesia, October 2006.

[2] J. Andersson, A. Wall, and C. Norström. Decreasing Maintenance Costs by Introducing Formal Analysis of Real-Time Behavior in Industrial Settings. In *Proceedings of the 1st International Symposium on Leveraging Applications of Formal Methods*, October 2004.

[3] J. Andersson, A. Wall and C. Norström. Validating Temporal Behavior Models of Complex Real-Time Systems. In *Proceedings of the 4th Conference on Software Engineering Research and Practice in Sweden (SERPS'04)*, Linköping, Sweden, September 2004.

[4] O. Balci. How to Assess the Acceptability and Credibility of Simulation Results. In *Proceedings of the 21st Conference on Winter Simulation*, *pages 62-71,* Washington, D.C., United States, 1989.

[5] A. V. Gafarian and J. E. Walsh. Statistical Approach for Validating Simulation Models by Comparison with Operational Systems. In *Proceedings of the 4*th *International Conference on Operations Research*, *pages 702-705,* New York, United States, 1969.

[6] J. Huselius, J. Andersson, H. Hansson, and S. Punnekkat. Automatic Generation and Validation of Models of Legacy Software. In *Proceedings of the 12*th *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), pages 342-349*, Sydney, Australia, August 2006.

[7] J. Huselius, J. Andersson. Model Synthesis for Real-Time Systems. In *Proceedings of the 9th*

European Conference on Software Maintenance and Reengineering, *pages 52–60*, March 2005.

[8] J. Huselius, J. Kraft, H. Hansson, and S. Punnekkat. Evaluating the Quality of Models Extracted from Embedded Real-Time Software. In *Proceedings of the 14th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, pages 577-585, IEEE*, Tucson, USA, March 2007.

[9] J. Huselius. Reverse Engineering of Legacy Real-Time Systems: An Automated Approach Based on Execution-Time Recording. *PhD Thesis, Mälardalen University Press,* June 2007.

[10] A. M. Law. How to Build Valid and Credible Simulation Models. In *Proceedings of the 38th Conference on Winter Simulation, pages 58-66,* Monterey, California, United States, 2006.

[11] A. Wall, J. Andersson, and C. Norström. Probabilistic Simulation-based Analysis of Complex Real-Time Systems. In *proceedings of the 6th IEEE International Symposium on Object-oriented Real-time distributed Computing, IEEE Computer Society,* Hakodate, Hokkaido, Japan, May 2003.

[12] A. Wall. Architectural Modeling and Analysis of Complex Real-Time Systems. *PhD Thesis, Mälardalen University Press,* September 2003.