Numerical Calculation of Model-Driven Performance Analysis and Simulation of Component-Based Architectures

Dupakuntla Veena*1, K V Prasada Reddy**2

*1Department of Computer Science & Engineering, Sir Vishveshwaraiah Institute of Science and Technology, Madanapalli-517325

**2Associate Professor, Department of Computer Science & Engineering, Sir Vishveshwaraiah Institute of Science and Technology, Madanapalli-517325

veenagowri.12@gmail.com
Prasadreddy.kv7@gmail.com

Abstract— In the course of last ten years, researchers have proposed a number of model transformations enabling performance predictions. These transformations map performance-annotated software architecture models into stochastic models solved by analytical means or by simulation. In existing they did not give detailed quantitative evaluation of the accuracy and efficiency of different transformations is missing, making it hard to select an adequate transformation for a given context. But in proposed system provides an in-depth comparison and quantitative evaluation of representative model transformations to, queuing Petri nets and layered queuing networks. The semantic gaps between typical source model abstractions and the different analysis techniques are spread out. The accuracy and efficiency of each transformation are calculated by considering four case studies representing systems of different size and complexity.

I. INTRODUCTION

To ensure that a software system meets its performance requirements, the ability to predict its performance under different configurations and workloads is highly valuable throughout the system life cycle. During the design phase, performance prediction helps software architects to evaluate different design alternatives. At deployment time, it facilitates system sizing and capacity planning. During operation, predicting the effect of changes in the workload or in the system configuration helps avoiding performance problems such as long response times or

A. Over-utilized resources. Recent performance prediction approaches for component-based architectures often rely on performance-annotated software architecture models (e.g., UML2, AADL), which are transformed into stochastic performance models, e.g., queuing networks (QNs), stochastic Petri nets (SPN), stochastic process algebra (SPA), and then solved to determine performance metrics of interest (e.g., response time or throughput). Existing performance prediction approaches include transformations from UML into for example layered queuing Networks (LQNs) stochastic well-formed nets, or stochastic process algebra. We compare existing performance prediction approaches that are based on model transformations from a performance annotated software architecture model to different analysis techniques. We provide an in-depth quantitative evaluation with regard to the trade-offs between prediction accuracy and solution efficiency. Starting with a process based simulation (PCM2SimuCom we compare various model solving approaches PCM2QPN and PCM2LQNS that are expected to gradually improve prediction speed at the expense of prediction accuracy. (PCM2SimuCom we compare various model solving approaches PCM2QPN and PCM2LQNS that are expected to gradually improve prediction speed at the expense of prediction accuracy. As contributions of this paper (i) we reveal the semantic gaps between typical source model abstractions and the different analysis techniques, (ii) we evaluate the effects of the identified semantic gaps on the performance prediction accuracy, (iii) we evaluate the efficiency of each transformation and respective analysis tool in the context of four representative case studies, and (iv) we consolidate the trade-offs between the performance transformations and analysis tools to provide practical guidance on deciding when to use which tool.

II. EXISTING SYSTEM

In existing they did not give detailed quantitative evaluation of the accuracy and efficiency of different transformations is missing, making it hard to select an adequate transformation for a given context. It includes transformations from UML into for example layered queuing networks (LQNs) stochastic well-formed nets or stochastic process algebra.
Disadvantage
Provide less efficiency and accuracy

III. PROPOSED SYSTEM

In proposed system provides an in-depth comparison and quantitative evaluation of representative model transformations to, queuing Petri nets and layered queuing networks. The semantic gaps between typical source model abstractions and the different analysis techniques are spread out. The accuracy and efficiency of each transformation are calculated by considering four case studies representing systems of different size and complexity.

IV. ALGORITHM

Mean Value Analysis

Mean value analysis (MVA) is an efficient algorithm that allow us to analyze product form queueing networks and obtain mean values for queue lengths and response times, as well as throughputs. The efficiency comes with a price – MVA does not compute the joint probability distribution for queue lengths. However, in many if not most performance evaluation situations, the mean values are the performance metrics of interest.

Single Class Systems

We begin with systems in which there is a single job class (job classes are usually referred to as “chains” in queueing network theory). These systems may either be open or closed.
Architecture-level performance modeling languages are powerful tools to describe the performance-influencing factors of a software system in a concise way. Since the architecture is reflected, component resource demands, performance-relevant component control flow and its dependence on service input parameters can be directly modeled. We summarize the main concepts of a performance-annotated software architecture model by means of the Palladio Component Model [12]. It is a mature modeling language for model-driven quality analysis of component-based software architectures [3] and has been used in a number of industry-relevant case studies. Currently supported quality predictions include performance, reliability, and maintenance costs, however, in this paper the focus is on performance modeling. The performance behavior of a component-based software system is a result of the assembled components’ performance behavior. In order to capture the behavior and resource consumption of a component, the following four factors are taken into account. First, the component’s implementation affects its performance. Additionally, a component may depend on external services whose performance has to be considered as well. Furthermore, both the way a component is used, i.e., its usage profile, and its execution environment are taken into consideration.

\[
\text{for } m=1 \text{ to } M \text{ do } q_m(0) = 0 \\
\text{for } n=1 \text{ to } \sum N_n \text{ do } \\
\quad \text{for each feasible population } \pi = (n_1, ..., n_C) \text{ s.t. } \\
\quad \pi = \sum n_i . n_i \geq 0 \\
\quad \text{do begin } \\
\quad \quad \text{for } c=1 \text{ to } C \text{ do } \\
\quad \quad \quad \text{for } m=1 \text{ to } M \text{ do } \\
\quad \quad \quad \quad R_{c,m}(0) = \frac{D_{c,m}}{D_{c,m}(1 + A_{c,m}(0))}, \text{ not IS } \\
\quad \quad \quad \text{for } c=1 \text{ to } C \text{ do } X_c = \frac{n_c}{\sum R_{c,m}(0)} \\
\quad \text{end begin } 
\]

Fig 2. Analysis of Prediction Accuracy

V. PERFORMANCE-ANNOTATED SOFTWARE ARCHITECTURE MODEL

Fig 3. Rate differences between to values

Fig 4. Cycle analysis of data rate values
4.2.1 Loops

shows a behavior specification of a service named send Orders To Manufacturing. An order consisting of multiple line items is sent to the manufacturing domain. After a short pre-processing step (an internal action), the service loops through the list of line items and sends them separately to manufacturing. The example is an adapted newOrder service of the SPECjEnterprise2010 benchmark1 (cf. Section 4.3.1).

The loop iteration number is described by a PMF. With a probability of 0.3, the loop iterates 2 times. Loop counts 3 and 4 have a probability of 0.1 respectively. 0.2. 5 loop counts have a probability of 0.4. While SimuCom is able to simulate exactly the given PMF, SimQPN introduces inaccuracies because of the geometric distribution of the individual loop iteration numbers in the transformed QPN (cf. Section 3.3). In Fig. 12, we see the PMF obtained using SimuCom in the left diagram. The right diagram shows the PMF of loop iteration numbers when using SimQPN. It is more spread out compared to the original PMF. Note that the resulting average of loop iteration numbers is 3.7, it is the same for both SimuCom and SimQPN. Fig. 13 and Table 2 compare the response time predictions for service send Orders To Manufacturing derived with SimuCom respectively SimQPN. As expected, the predicted average response times are comparable. However, the response time distribution is more spread out with SimQPN than with SimuCom although in the example the exponentially distributed resource demands tend to smooth the differences of the predictions. As shown in Table 2, the 90th and 95th percentile response times are significantly higher than the reference values derived with SimuCom.
This paper presents a comparison of different transformations of performance-annotated software architecture models into stochastic performance models which apply both analytical and simulation-based solution strategies. It presents a detailed evaluation of the effects of the identified semantic gaps on the prediction accuracy as well as several case studies to provide end-to-end evaluations of the different model transformations. The results presented in this paper help performance analysts to select the right transformation depending on the analysis context. Our future work is to take automatic decision for transformation using decision tree. The automated decision has to consider the level of details of the model of the system under study and available time until the performance prediction results are required.

VI. CONCLUSION

This paper presents a comparison of different transformations of performance-annotated software architecture models into stochastic performance models which apply both analytical and simulation-based solution strategies. It presents a detailed evaluation of the effects of the identified semantic gaps on the prediction accuracy as well as several case studies to provide end-to-end evaluations of the different model transformations. The results presented in this paper help performance analysts to select the right transformation depending on the analysis context. Our future work is to take automatic decision for transformation using decision tree. The automated decision has to consider the level of details of the model of the system under study and available time until the performance prediction results are required.

REFERENCES


