Preliminary Study of Software Performance Models

Issamjebreen
Faculty of Information Technology
Zarqa University, Zarqa, Jordan

Mohammed Awad
Faculty of Applied Engineering
University of Palestine

***Abstract*—Context: Software performance models can be obtained by applying for specific roles, skills and techniques in software life cycle, and it depends on formulating the software problem as well as gathering the performance requirements. This paper presents a preliminary review of the software performance models. This constitutes a reference for the IT companies and personnel that help them select the suitable model for their projects. Also, the study helps researchers find out further research areas in this field. A preliminary review according to a predefined strategy is used to conduct previous approaches of software performance models integrated with software development cycle in early software cycle. A review has been done for exploring and comparing the software performance models that are published previously. This study results in a comprehensive review for the existing software performance models. This review composes a clear reference for highlighting the weak and strength points of these models.**

Keywords*—Performance Models; Measurement Model; Performance Prediction; Performance evaluation

I. Introduction

Developing software requires ensuring that software performance requirements are considered and achieved. Software performance is a process to predict and evaluate if the system meets business goals. Performance Predictive models (Model-Based) require detail descriptions at run-time behavior of a system, in order to estimate the execution time and other performance issues i.e. cache misses. It is used by architects to avoid the performance problems at system implementation time, and to estimate designs, and to explore a new optimization by compiler writers. In addition, developers can adjust their programs. Conversely, Evaluation models (measurement models) attempts to measure the system performance activities when the system has been implemented. In order to defined performance problems and bottlenecks.

This paper presents a preliminary review of the software performance models. The main goal is to presents explorations of research of performance models as well as clarifying the variance of elements used for each model. In order to help researchers to find out further research areas and to constitute a reference for the IT companies and personnel that helps them select the suitable model for their projects.

The rest of this paper is organized as follows: Sec2: Research Description and Presentation; Sec3: Literature Review, Sec4: Preliminary Results and Discussion; Sec5: Conclusion and Future Work.

II. The Review Process

This section declares the review process, research framework and the objectives of the preliminary review. This paper aims to present a preliminary review between "Performance models", in order to clarify the elements used to generate these models. The framework of this review considers studies of software performance based on simulation, Analytical methods, and component based approach in order to explore the elements used to generate the performance models.

A. Research Planning

The research Strings was established by academic as following: ("Software Performance Engineering, Modeling Techniques, UML, Performance Models."), the Research question (RQ) is: RQ. What are the elements used to generate the performance models? Our research resources namely: IEEE and ACM Digital Library.

B. Conducting the Research

The criteria for determining whether a study should be included as a related study (named "Primary Study") or not, was by first analyzing research titles, abstracts, keywords and introductions from the studies retrieved through search.

C. Selection of the Primary Studies

The inclusion criteria for the selection of primary studies are listed below:

- Studies that proposed Performance models".
- Studies that describe their methods in details.

The exclusion criteria for the selection of primary studies are listed below:

- Studies that don't answer the research question.
- Studies that don't present Models OR Meta-Models of software performance.

III. Primary Studies

Smith & Williams [1], who have defined SPE (Software Performance Engineering) information requirements, have proposed integrated software development cycle with performance models; They defined information requirements for Early Life-Cycle performance analysis, the performance analysis according to authors were: performance objective, performance scenario that includes software plans + workload, execution environment, resource requirement and processing overhead.

They starting building the Meta model SPE from the performance scenario, using the workload to describe the ratio of different types of requests; SW plans defined the execution path for each workload, also used class diagrams to define the objects and the relationships between them, and thus modeling the performance scenarios using a form Execution Graph EG. That enables transferring information between CASE Tools and performance model. Also Smith et al, [2] updated the SPE meta-model that is proposed by Smith & Williams by adding subclass to processing node and adding project, facility node; then applying XML formats to Software Performance Model Interchange Format (S-PMIF meta-model) and export UML Diagrams when they are ready into the S-PMIF.

Additionally, Henia, Rafik, et al. [3] proposed an approach named SymTA/S, which is considered as system-level performance as well as timing analysis, and based on formal scheduling analysis techniques, in order to support diverse Architectures & task dependencies & collect optimization algorithms with analysis of rapid design. Support performance issues such as bus, processor utilization, and worst case scheduling scenarios. Moreover, D'Ambrogio, Andrea [4] presented a framework that aims to transfer source UML of software models into performance prediction models layered queuing networks LQNs, which required understanding the syntax & semantics for the source and P models. That enhances software designer's productivity as well as software quality.

Smith, Connie U., & Lloyd G. Williams [5] said that most performance anti pattern problems result through the architecture/design stages, unfortunately these problems don't appear during the implementation stage. The solutions need software changes opposed to system tuning changes. Smith, Connie U., & Lloyd G. Williams [5] presented three new performance anti patterns and gave examples to illustrate them. These anti patterns help developers and performance engineers avoid common performance problems.

Woodside, Murray, et al. [6] analyzed the exchange information provided from a performance model and the process of creating a performance model. They proposed PUMA transformations that define Performance evolutions from annotated UML Profile for Modeling and Analysis of Real-Time Embedded System MARTE. This approach enables to obtain performance measures such as throughout and response time throughout software life-cycle. Moreover, Sim, Jaewoong, et al. [7] proposed a framework in order to analyze the performance, which supports shed light of bottlenecks of GPGPU applications. In addition, this framework helps GPGPU Profile tools and supports programmers in measurements as well as metrics during run time.

Bammi, Jwahar R., et al. [8] proposed two approaches for handling issues of performance evaluation as well as SW cost for embedded system design. The first approach is called Source-Based approach which employs the integration of a virtual instruction set in order to evaluate the performance. The second approach called object-based approach which translates the assembler created by the target compiler (named assembler-level).

Lindemann, Christoph, et al. [9] proposed a framework for performance estimation which enables designers to predict performance during variance stages at design phase. They presented an algorithm that supports the state space creation resulted from State & Activity diagrams. In order to enable a quantitative analysis for Stochastic Process and Generalized Semi-Markov Process GSMP is used. Additionally, Denaro, Giovanni, Andrea Polini, and Wolfgang Emmerich [10] proposed an approach for performance testing in particular for distributed systems during early life cycle phases. They created test cases to examine these systems starting from architectural designs. They observed that middleware functionality e.g. transactions & remote communication primitives control these systems.

Bertolino, Antonia, and Raffaela [11] presented an approach named CB-SPE for component-based SW performance, which adopted CB (Component-Based) framework to model the standard RT-UML Profile restructure depending on CB Role. CB-SPE approach applied on both component layer (parametric performance estimation) and application layer (predictive performance for assembled components).

Tribastone, Mirco, and Stephen Gilmore [12] proposed a procedure to systematically map activity diagrams into stochastic process algebra model referred as PEPA Models. PEPA performance model clarifies a Markov [9] in semantics to enable the computation of performance issues i.e. workload, response time and the throughput. They are concerned about tools produced in Eclipse platform; to enable transfer from MARTE annotated UML activity diagrams into PEPA Models

Gu, Gordon P., and Dorina C. Petriu [13] present a method that enables transfer between the results annotated from UML with performance models, which is generated at a higher level of abstraction. They use a lower level XML trees manipulations i.e. XML algebra. They use also LQN to apply their method, which can be designed to other performance model formalisms. Moreover, Zheng, Gengbin, et al, [14] proposed a performance predictive model for big weigh computers (i.e. blue gene machine), that include a parallel simulator, bigsim, bignetsim. The simulator can deal with advanced features of modeling, also supports performance predictions for huge machines. In addition, Kähkipuro, Pekka [15] proposed a framework, in order to introduce performance modeling; at first they have explained an overview of the proposed framework and clarified the major components for this framework. After that they have clarified relationships between these components.

Finally, Zolfaghari, Rahmat [16] presented a method for transforming UML of SW architecture to QNM (Queuing Networks Model). In order to support performance as well as quality of the models that employ UML in designing software. They have used the deployment diagram in SW components with hardware resources. The activity diagrams extract the system behaviors and the use case diagrams extract workloads.

IV. Results and Discussion

Table 1 shows the Advantages & Limitations of Performance Models. Regarding Data Extraction, this research has predefined Database that contains Authors, Titles, Published Years, descriptions and summaries of this

comprehensive review. For the Evidence Synthesis, most approaches employ the results annotated from UML, SPT profiles to integrate software models with performance models through high abstract level information. The Use Case diagrams are used to describe workload density, and behaviors, while the Activity and Sequence diagrams are used to extract computations of a system performs service requests to the devices resources (the dynamic behavior). Deployment Diagram provides hardware recourses such as passive and active resource modeling. Processing resources extract from Active resources (devices), while operating system processes extract from Passive resources.

TABLE I. ADVANTAGES & LIMITATIONS OF PERFORMANCE MODELS

Ref.	Advantages	Limitations
[1]	Provides an interchange format that enables CASE & Performance Tools to exchange information	Considers Class & ER Diagrams only.
[2]	Provides interchange formats that support flexibility in when & who performance specifications are provided	Needs SPT Profile to export the resource requirements Considers Class Diagrams only
[3]	Supports diverse Architectures & task dependencies Supports performance issues such as bus, processor utilization, and worst case scheduling scenarios.	Provides only a system-level performance
[4]	Transfers source UML of software models into performance prediction models layered queuing networks LQNs Enhances software designer's productivity as well as software quality	which required understanding the syntax & semantics for the source and Pmodels
[5]	Provides three new performance anti patterns. These anti patterns help developers and performance engineers avoid common performance problems	The solution need software changes opposed to system tuning changes
[6]	Provides PUMA transformations that define performance evolutions from annotated UML Profile for Modeling and Analysis of Real-Time Embedded System MARTE. Enables to obtain performance measures such as throughout and response time throughout software life-cycle	Focuses only on Real-Time Embedded System MARTE
[7]	Supports shed light of bottlenecks of GPGPU applications. Supports programmers in measurements as well as metrics during run time	it assumes that a memory instruction is always followed by consecutive dependent instructions; hence, MLP is always one. it assumes that there is enough instruction-level parallelism. Thus, it is difficult to predict the effect of prefetching or other optimizations that increase instruction/memory-level parallelism.
[8]	Employs the integration of a virtual instruction set in order to evaluate the performance. Translates the assembler created by the target compiler (named assembler-level).	it is quite difficult to account for potential compiler optimizations that do not fall into any of the Virtual Instruction categories
[9]	Enables designers to predict performance during variance stages at design phase. Supports the state space creation resulted from State & Activity diagrams.	Limited to Time-enhanced UML Diagrams
[10]	Enables test cases to examine systems starting from architectural designs	Cannot identify performance problems that are due to the specific implementation of late-available components. For example, if the final application is going to have a bottleneck in a business component that is under development, the approach has no chance to discover the bottleneck that would not be exhibited by a stub of the component. Performance analysis models remain the primary reference to pursue evaluation of performance.
[11]	Applied on both component layer (parametric performance estimation) and application layer (predictive performance for assembled components)	it leads to sound results only for a specific platform
[12]	Enable the computation of performance issues such as workload, response time and the throughput	Restricted to final node activities
[13]	Enables transfer between the results annotated from UML with performance models, which is generated at a higher level of abstraction. Uses a lower level XML trees manipulations such as XML algebra. Uses also LQN to apply their method, which can be designed to other performance model formalisms	Cannot build the complete behavior for every component
[14]	Provides a performance predictive model for big weigh computers (i.e. blue gene machine), that includes a parallel simulator, bigsim, bignetsim.	Large meshes must be generated, which is difficult with today's tools. The meshes must be partitioned for parallel execution
[15]	Introduces performance modeling Clarifies the major components and relationships between these components	the approach limits the available scheduling disciplines, service time distributions, and arrival rate distributions
[16]	Supports performance as well as quality of the models that employ UML in designing software. Uses the deployment diagram in SW components with hardware resources.	Must have both software and hardware components to follow it. It is not perfect for pure software solutions

V. Conclusion and Future Work

The results of this review show that most approaches widely used UML Diagrams and SPT Profiles to support generation of Performance Models. There are several performance models introduced to provide analytical assessment. These models aim to help designers and architects to predict performance measurements at different steps.

This review has revealed that data pre-processing has received considerable attention in the Software Engineering research community. The same cannot be said regarding data collection procedures and the identification of data quality issues, which can compose future research topics.

Acknowledgment

This research is funded by the Deanship of Research and Graduate Studies in Zarqa University /Jordan

References

[1] Williams, L., Smith, C., “Information Requirements for Software Performance Engineering”, the National Science Foundation, 1995

[2] Smith, C., Lladó. C., Cortellessa, V., Marco, A., Williams, L., “ From UML models to software performance results: An SPE process based on XML interchange formats”, WOSP, 2005

[3] Henia, Rafik, et al. "System level performance analysis–the SymTA/S approach." IEE Proceedings-Computers and Digital Techniques 152.2 (2005): 148-166.

[4] D'Ambrogio, Andrea. "A model transformation framework for the automated building of performance models from UML models." Proceedings of the 5th international workshop on Software and performance. ACM, 2005.

[5] Smith, Connie U., and Lloyd G. Williams. "More new software performance antipatterns: Even more ways to shoot yourself in the foot." Computer Measurement Group Conference. 2003.

[6] Woodside, Murray, et al. "Transformation challenges: from software models to performance models." Software & Systems Modeling 13.4 (2014): 1529-1552.

[7] Sim, Jaewoong, et al. "A performance analysis framework for identifying potential benefits in GPGPU applications." ACM SIGPLAN Notices. Vol. 47. No. 8. ACM, 2012.

[8] Bammi, Jwahar R., et al. "Software performance estimation strategies in a system-level design tool." Proceedings of the eighth international workshop on Hardware/software codesign. ACM, 2000.

[9] Lindemann, Christoph, et al. "Performance analysis of time-enhanced UML diagrams based on stochastic processes." Proceedings of the 3rd international workshop on Software and performance. ACM, 2002.

[10] Denaro, Giovanni, Andrea Polini, and Wolfgang Emmerich. "Early performance testing of distributed software applications." ACM SIGSOFT Software Engineering Notes. Vol. 29. No. 1. ACM, 2004.

[11] Bertolino, Antonia, and Raffaela Mirandola. "Towards component-based software performance engineering." Proceedings of the 6th ICSE Workshop on Component-Based Software Engineering. 2003.

[12] Tribastone, Mirco, and Stephen Gilmore. "Automatic extraction of PEPA performance models from UML activity diagrams annotated with the MARTE profile." Proceedings of the 7th international workshop on Software and performance. ACM, 2008.

[13] Gu, Gordon P., and Dorina C. Petriu. "From UML to LQN by XML algebra-based model transformations." Proceedings of the 5th international workshop on Software and performance. ACM, 2005.

[14] Zheng, Gengbin, et al. "Simulation-based performance prediction for large parallel machines." International Journal of Parallel Programming 33.2-3 (2005): 183-207.

[15] Kähkipuro, Pekka. "UML based performance modeling framework for object-oriented distributed systems." «UML»’99—The Unified Modeling Language. Springer Berlin Heidelberg, 1999. 356-371.

[16] Zolfaghari, Rahmat. "Software Performance Evaluation with Converting UML Description of Software Architecture to QNM." Int. J. Emerg. Sci 3.3 (2013): 268-27