Deriving Performance Models from Software Architecture Specifications

Simonetta Balsamo and Marta Simeoni
Dipartimanto di Informatica, Università Ca' Foscari di Venezia, Italy[•]
{balsamo, simeoni}@dsi.unive.it

Abstract. Quantitative analysis of software systems has been recognized to be important and useful for requirements and design, and, specifically, performance analysis should be integrated in the software development life cycle from the early stages. To this aim, several approaches have been recently proposed to integrate or combine performance analysis and software architecture specification. In this paper we present a brief review and a comparison of some approaches that derive performance models from SA specification. We focus on the generality of the proposed methodologies, the required constraints and assumptions, the type of performance model, the separation of performance model and specification language, the implementation, and how easily the obtained performance results can provide a feedback to the software designer.

1. Introduction

There is a growing interest in quantitative analysis of software systems, which is important and useful for requirements and design, and it has been recognized that performance analysis should be integrated in the software development life cycle from the early stages [Smi90, WOSP2000]. Software Architectures (SA) describe the structure of software systems at a high level of abstraction and have been devised as the appropriate design level to perform quantitative analysis. To this aim, several approaches have been recently proposed to integrate or combine performance analysis and software architecture specification. Various types of performance models and different specification languages have been considered. Some approaches refer to the entire software life cycle, whereas others refer to a certain software stage, usually design specification.

In this paper we consider some approaches that derive performance models from SA specification. We present a brief review and a comparison of such different approaches, focusing on the generality of the proposed methodology, the required constraints and assumptions, the type of performance model, the implementation level, and the separation of performance model and specification language. The feedback to the software designer of the results of the software performance analysis is another important issue to be considered in the approaches' comparison, i.e., how easily the designer can interpret the quantitative results obtained by the performance evaluation at the SA design level. Several approaches refer to a software specification based on the Unified Modeling Language (UML), which is becoming a standard notation for the specification of software systems [BRJ99]. It provides several kind of diagrams, which allow the description of different aspects and properties of systems, like static and behavioral aspects, interaction among system components and physical implementation details.

The paper is organized as follows. Section 2 reviews various approaches of derivation of performance models from SA specification and Section 3 presents some comparisons and observations. Conclusions are given in Section 4.

2. From SA specification to performance models: some approaches

We are interested in analyzing and comparing different approaches and methodologies concerning the derivation of performance models from software architecture specification. In this section we present a brief description of some approaches . We name them with the initials of their authors.

We consider the generality of the methodologies and we point out whether they introduce architectural constraints or special assumptions on the software system. Each approach is based on a certain type of performance model and specification language. The latter include specification formalisms such as process algebras, Petri nets and Chemichal abstract machine, and UML based specification. Performance models include queueing networks (QN) and their extensions called Extended Queueing Networks (EQN) and Layered Queueing Networks (LQN), Stochastic Timed Petri nets (STPN), Stochastic Process Algebras (SPA) and simulation models.

Some of the proposed methods are based on the Software Performance Engineering (SPE) methodology introduced by Smith in her pioneer work [Smi90]. It has been the first comprehensive approach to the integration of performance analysis into the software development process, from the earliest stages to the end. The SPE methodology is based on two models: the *software execution model* and the *system execution model*. The former is based on execution graphs (EG) and represents the software execution behavior, the latter is based on queueing network models and represents the computer system platform, including hardware and software components. The analysis of the software model gives information concerning the resource requirements of the software system. The obtained results, together with

[•] This work has been partially supported by MURST Project Research Funds Saladin.

information about the hardware devices, are the input parameters of the system execution model, which represents the model of the whole software/hardware system.

2.1.General Methodologies

We review some approaches that propose a general methodology for deriving performance models from software architecture specifications. These methods refer to different specification languages and performance models and they consider the combination of different tools and environments for system performance evaluation.

WS: Williams and Smith in [WS98] apply the SPE methodology to evaluate the performance characteristics of a software architecture. The emphasis is in the construction and analysis of the software execution model, which is considered the target model of the specified SA and is obtained from the Sequence diagrams. The Class and Deployment diagrams contribute to complete the description of the SA, but are not involved in the transformation process. The SPE process requires additional information that includes software resource requirements for processing steps and computer configuration data.

MG: In [MG98], Menascè and Gomaa present a methodology to derive QN performance models from SA specification. It has been developed and used by the authors in a design of client/server applications. The methodology is based on CLISSPE (CLIent/Server Software Performance Evaluation), a language for the software performance engineering of client/server applications (see [M97]). Although the methodology does not explicitly use UML, the functional requirement of the system are specified in terms of use cases, and the system model is specified by the analogous of a Class diagram. The use cases, together with the client/server SA specification and the mapping associating software components to hardware devices, are used to develop a CLISSPE program specification. The CLISSPE system provides a compiler that generates a corresponding QN model. By considering specific scenarios one can define the QN parameters and apply appropriate solution methods, such as the Layered Queuing Models (LQN) [RS95, WNPN95], to obtain performance results.

BIM: In [BIM98] Balsamo et alt. provide a method for the automatic derivation of a queuing network model from a SA specification, described using the CHAM formalism (CHemical Abstract Machine). Informally, the CHAM specification of a SA [IW95] is given by a set of *molecules* which represent the static components of the architecture, a set of *reaction rules* which describe the dynamic evolution of the system through reaction steps, and an initial *solution* which describes the initial static configuration of the system. The paper presents an algorithm to derive a QN model from the CHAM specification of a SA architecture. It is based on the analysis of the Labeled Transition System (LTS) that represents the dynamic behavior of the CHAM architecture, and that can be automatically derived from the CHAM specification. The algorithm does not completely define the QN model whose parameters, such as the service time distributions and the customer's arrival processes, have to be specified by the designer. The solution of the QN model is derived by analytical methods or possibly by symbolic evaluation. Parameter instantiation identify potential implementation scenarios and the performance results allow to provide insights on how to carry on the development process in order to satisfy given performance criteria.

CM: In [CM00] Cortellessa and Mirandola propose a methodology making a joint use of information from different UML diagrams to generate a performance model of the specified system. The paper refers to SPE methodology and specifies the software architecture by using Deployment, Sequence, and Use Case diagrams. This approach is a more formal extension of the WS approach [WS98] and consists of the following steps:

1. Enrich the Use Case diagram (UCD) by assigning a probability to every edge that links a type of user to a use case, so that such probability applies to the execution of the corresponding set of Sequence diagrams. This leads to the definition of the workload of the performance model.
2. For each Use Case in the UCD, process the corresponding set of Sequence diagrams to obtain the *meta*-EG (execution graph). The algorithm incrementally builds the EG by processing in turn all the Sequence diagrams and building, for each one, the part of the EG it contributes to. It considers only Sequence diagrams without *no-reply* and *asynchronous* interactions.
3. Use the Deployment Diagram both to obtain the EQN model of the hardware platform and to appropriately tailor the *meta*-EG so obtaining an EG-instance that defines the workload of the EQN. In this step, the basic idea is to enrich the Deployment Diagram, that shows the topology of the platform and the type of sites, with the information needed to build the EQN, such as the internal structure and parameters of devices. Moreover the EG-instance defines the type of communication and the size of data exchanged. The paper does not present the details of the EQN construction from the Deployment diagram.
4. Assign numerical parameters, defined by the designer, to the EG-instance.
5. Combine EG-instance and EQNM to solve the obtained performance model by using the SPE approach.

Note that a key point of the methodology is adding the information concerning performance evaluation to the considered UML diagrams, and to the obtained EQN model. The methodology is not yet implemented in a tool.

AABI: Andolfi et alt. propose [AABI00] an approach to automatically generate queuing network models from software architecture specifications described by means of Message Sequence Charts (MSC), that correspond to Sequence diagrams in the UML terminology. The idea is to analyze MSCs in terms of the trace languages (sequences of events) they generate, in order to single out the real degree of parallelism among components and their dynamic dependencies. This information is then used to build a QN model corresponding to the software architecture description. The authors present an algorithm to perform this step. This approach is built on the previous work [BIM98] to overcome the drawback of the high computational complexity due to possible state space explosion of the finite state model of the CHAM description.

ABI: The approach proposed by Aquilani et alt. in [ABI00] concerns the derivation of QN models from Labeled Transition Systems (LTS) describing the dynamic behavior of SAs. Starting from a LTS description of a SA makes it possible to abstract from any particular SA specification language. The approach assumes that LTSs are the only knowledge on the system that they can use. This means, in particular, that it does not use any information concerning the system implementation or deployment. This approach is an extension of [ABI00], considering a finite state representation independent of a specific architectural description language and modeling more complex interaction patterns and synchronization constraints that can be represented by extended QN. Such EQN models the software concurrent execution and component interaction at the SA design level.

BCD: Bernardo et alt. in [BCD00] propose an architectural description language based on stochastically timed Process Algebras. This approach provides an integration of a formal specification language and performance models. The aim is to describe and analyze both functional and performance properties of SAs in a formal framework. The approach proposes the adoption of an architectural description language called ÆMPA, gives its syntax with a graphical and textual notation and its semantics in terms of EMPA specifications, that is a stochastically timed process algebra (see [B00]). The authors illustrate various functional and non-functional properties, including performance evaluation which is based on the generation of the underlying Markov chain that is numerically solved. To this aim the authors propose the use of TwoTowers [B00], a software tool for systems modeling and analysis of functional and performance properties, that support system EMPA description.

2.2 Methodologies based on architectural patterns

Architectural patterns identify frequently used architectural solutions and are used to describe SAs. Each pattern is described by its structure (what are the components) and its behavior (how they interact). Some approaches consider software specification of architectural patterns and derive their corresponding performance models. They use UML specification. The approaches identify a direct correspondence between each pattern and its performance model, which can be immediately derived.

GM: In [GM00] Gomaa and Menascè investigates the design and performance modeling of component interconnection patterns for client/server systems. Such patterns define and encapsulate the way client and server components of software architecture communicate with each other via connectors. The idea is to start with UML design models of component interconnection patterns, using Class diagrams (to model their static aspects) and Collaboration diagrams (to depict the dynamic interactions between components and connectors objects - instances of the classes depicted on the Class diagrams). Such models are then provided with additional performance annotations, and translated into an XML notation, in order to capture both the architecture and performance parameters in one notation. The performance models of the considered patterns are extended QN and their definition, based on previous work of the authors, depends on the type of communication. The EQN model solution is obtained by Markov chain analysis or approximate analytical methods.

PW: In [PW99, P00] Petriu and Wang consider a significant set of architectural patterns (pipe and filters, client/server, broker, layers, critical section and master-slave) specified by using UML-Collaborations that are combined Class and Sequence diagrams showing explicitly the collaborating objects. The approach shows the corresponding performance models based on LQN models. Moreover, they propose a systematic approach to build performance models of complex SAs based on combinations of the considered patterns. The approach follows the SPE methodology and generates the software and system execution models by applying graph transformation techniques. SAs are specified using UML-Collaborations, Deployment and Use Case diagrams. The Sequence diagram part of the UML-Collaboration is used to obtain the software execution model (which is represented as a UML Activity diagram); the Class part is used to obtain the system execution model (which is represented as a LQN model). Use Case diagrams provide information on the workloads, and Deployment diagrams allow for the allocation of software components to hardware sites.

2.3. Simulation methods

We shall now consider two approaches based on simulation models that consider simulation packages to define the model, whose structure and input parameters are derived from the information obtained from UML diagrams.

AS: The approach proposed by Arief and Speirs in [AS00] presents a simulation framework named Simulation Modelling Language (SimML) to generate a simulation program from the system design specified with the UML. The proposed UML tool allows the user to draw Class and Sequence diagrams and to specify the information needed for the automatic generation of the process oriented simulation model. The simulation program is generated in the Java programming language. The approach proposes an XML translation of the specified UML models, in order to store the information about the design and the simulation data in a structured way.

DLHBP: The approach proposed by de Miguel et alt. in [DLHBP00] focus on real time systems, and proposes extensions of UML diagrams to express temporal requirements and resource usage. The extension is based on the use of stereotypes, tagged values and stereotyped constraints. SAs are specified using the extended UML diagrams without restrictions on the type of diagrams to be used. Then these UML diagram are used as input for the automatic generation of the corresponding simulation models in OPNET. They also define a middleware model for scheduling analysis. The simulation model is defined by instantiating with the application information the generic models that represent the various UML metaclasses. The approach generates submodels for each application element and combines them into a unique simulation model of the UML application. The approach provides also a feedback mechanism: after the model has been analyzed and simulated, some results are included in the tagged values. This constitutes a relevant feature, which ease the SA designer in obtaining feedback from the performance evaluation results.

2.4 Case Studies

Some approaches present the generation of performance models from a software specification through an example or a case study. They consider UML specification and different types of performance models.

KP: In [KP99] King and Pooley show, through an example, how to generate Stochastic Timed Petri Net models from the UML specification of systems. They consider Use Case diagrams and combined diagrams consisting of a Collaboration diagram with State diagrams (i.e. statecharts) of all the collaborating objects embedded within them. The idea is to translate each State diagram that represents an object of the Collaboration diagram into a Petri net: states and transitions in the State diagram are represented by places and transitions in the Petri net, respectively. The obtained Petri nets can be combined to obtain a unique STPN model that represents the whole system, specifically a Generalized stochastic Petri net (GSPN) [ABC86]. The merging of nets is only explained via the running example, i.e. the paper does not include a general merging procedure. The GSPN can be analyzed by specific tools such as the SPNP package.

PK: In [PK99] Pooley and King describe some preliminary ideas on how to derive a queuing network model from the UML specification of a system. Use Case diagrams are used to specify the workloads and the various classes of requests of the system being modeled. Implementation diagrams are used to define contention and to quantify the available system resources. The idea is to define a correspondence between combined Deployment and Component diagrams and queuing network models, by mapping components and links to service centers.

P: In [Poo99] Pooley describe how to derive stochastic process algebra models from UML specifications. More precisely, the starting point is, like in [KP99], the specification of a system via a combined diagram consisting of a Collaboration diagram with State diagrams (i.e. statecharts) of all the collaborating objects embedded within them. The idea is to produce a Stochastic Process Algebra description of each object of the Collaboration diagram and to combine them into a unique model. The paper shows how this can be done on a real although simple example. The paper presents also an attempt to generate a continuous-time Markov chain directly from the combined UML diagram of the running example. The key observation here is that, at any time, each object of the collaboration diagram must be in one, and only one, of its internal states. The combination of the objects current states is called "marking". The idea is to derive all possible markings by following through the interactions: this allows building the corresponding state transition diagram and, then, the underlying Markov chain.

SW: Smith and Williams in [SW97] present an example to illustrate the derivation of a performance model from an object-oriented design model, and propose the use of the SPE•ED tool that supports the SPE methodology to evaluate object-oriented systems. Starting from a set of scenarios described by Message Sequence Charts, they derive the execution graphs that define the software execution model and then by the analyst specification of computer resource requirements they define the system execution model. The QN model is analyzed by approximate analytical methods or by simulation integrated in the SPE•ED tool. This work is related to the WS approach described in Section 2.1.

H: In [Hoe00] Hoeben discusses some rules that can be used to express or add information useful to derive performance evaluation from the various UML diagrams. The work proposes some UML extensions based on the use of stereotypes and tagged values and some rules to propagate user requests specified by UML models to define the performance model. These rules allow performance evaluation of UML models at various levels of abstraction and a prototype tool to automatically create performance estimates based on QN models uses them. The author

gives just some hints on how to obtain a QN model from the UML model of a system, and does not provide a complete tool description.

Table I summarizes some relevant features of the considered approaches. For the ABI approach we have marked the LTS because it is not a proper specification language.

APPROACH	SOFTWARE ARCHITECTURE SPECIFICATION LANGUAGE	ARCHITECTURAL CONSTRAINTS	PERFORMANCE MODEL
WS	MSC - UML: Deployment, Sequence and Class diagrams	----	EQN
MG	client/server specification, use case	client/server systems	QN / LQN
BIM	CHAM	---	QN
CM	UML: Deployment, Sequence and Use Case diagrams	---	EQN
AABI	MSC - UML: Sequence diagram	---	QN
ABI	LTS*	---	EQN
BCD	ÆMPA	---	SPA
GM	UML: Collaboration and Class diagrams	client/server systems: component interconnection patterns for synchronous and asynchronous communication with a multi-threaded server	EQN
PW	UML: Deployment, UML-Collaboration, Use Case and Activity diagrams	patterns for client/server, pipe and filters, broker, layers, critical section, master-slave	LQN
AS	UML: Sequence and Class diagram	---	Simulation
DLHBP	UML: all diagrams	real time systems	Simulation
KP	UML: Use Case, combined State and Collaboration diagrams	---	STPN (GSPN)
PK	UML: Use Case, combined Deployment and Component diagrams	---	QN
P	UML: combined State and Collaboration diagrams	---	SPA
SW	MSC	---	EQN
H	UML: all diagrams	---	QN

3. Observations

The various approaches described in the previous section range from general methodologies to case studies. Various types of performance models and different specification languages have been considered.

Most of the approaches refer to the entire software life cycle, whereas some refer to the design specification stage. The latter ones usually consider a formal behavioral software specification modelled by Stochastic Petri Nets or Stochastic Process Algebras, such as BCD. Such models integrate functional and non-functional aspects and provide a unique reference model for SA specification and performance. However, from the performance evaluation viewpoint, the analysis usually refers to the numerical solution of the underlying Markov chain which can easily lead to numerical problems due to the state space explosion. Moreover, from the software designer viewpoint, it is not clear a direct and easy interpretation of the numerical performance results in terms of SA design.

Several approaches are based on queueing network models (QN, EQN, LQN) and refer to a higher abstraction level that could make the feedback at the SA design easier than with more detailed stochastic models. In particular when SA components can be related to queueing network components (i.e., service centers or subnetworks), performance results of the performance model can be directly interpreted at the SA design level.

A common aspect of the WS and AABI approaches is that they are only interested in analyzing the SA, without specifying (or using information regarding) the underlying hardware platform. However, while in WS (and in CM) the EQN model is derived by the SPE methodology, this approach defines the queuing network model by the analysis of a Labeled Transition System associated to the Sequence diagram.

Some approaches introduce constraints on the SA, like MG that applies to client/server systems or GM and PW that consider architectural patterns. However, the latter provides a method to combine the considered patterns.

Most of the recent approaches consider the UML specification and make use of various diagrams to derive information to define and parameterize the performance model. It is often considered necessary to introduce additional

information to the UML diagram through simple annotation or extensions based on stereotypes and tagged values in order to complete the software specification for deriving a performance evaluation model.

Various tools have been proposed or used to implement some steps of the proposed approaches. However, none of them have been yet implemented into a complete environment for specification, performance analysis and feedback to the software designer. An open problem and challenge is to completely automatize the process of deriving performance models from software specification and to integrate the supporting tools in a unique environment.

4. Conclusion

Integration of software specification and performance evaluation model is considered an important and useful task from the early stages of the software life cycle. We have reviewed and discussed various approaches to derive performance models from software architecture specification, pointing out their features in terms of generality, type of performance models and specification languages, implementation and feedback to the software designer.

References

[ABC86] M. Ajmone, G. Balbo, G. Conte "Performance Models of Multiprocessor Performance" The MIT Press (1986).

[AABI00] F. Andolfi, F. Aquilani, S. Balsamo, P. Inverardi "Deriving Performance Models of Software Architectures from Message Sequence Charts" in [WOSP2000].

[ABI00] F. Aquilani, S. Balsamo, P. Inverardi "Performance Analysis at the software architecture design level" Technial Report TR-SAL-32, Technical Report Saladin Project, 2000, submitted for pubblication.

[AS00] L.B. Arief, N.A. Speirs "A UML Tool for an Automatic Generation of Simulation Programs" in [WOSP2000] pp. 71-76.

[BIM98] S. Balsamo, P. Inverardi, C. Mangano "An Approach to Performance Evaluation of Software Architectures" Workshop on Software and Performance, WOSP'98, Santa Fe, New Mexico, Oct 12-16, 1998.

[B00] M. Bernardo "Theory and Application of Extended Markovian Process Algebra" Ph. D. Thesis, University of Bologna (Italy), 1999.

[BCD00] M. Bernardo, P. Ciancarini, L. Donatiello "AEMPA: A Process Algebraic Description Language for the Performance Analysis of Software Architectures" in [WOSP2000].

[BRJ99] G. Booch, J. Rumbaugh, I. Jacobson "The Unified Modeling Language User Guide" Addison-Wesley (1999).

[CM00] V. Cortellessa, R. Mirandola "Deriving a Queueing Network based Performance Model from UML Diagrams" in [WOSP2000] pp. 58-70.

[DLHBP00] M. De Miguel, T. Lambolais, M. Hannouz, S. Betgè-Brezetz, S. Piekarec "UML Extensions for the Specification and Evaluation of Latency Constraints in Architectural Models" in [WOSP2000] pp. 83-88.

[GM00] H. Gomaa, D.A. Menascè "Design and Performance Modeling of Component Interconnection Patterns for Distributed Software Architectures" in [WOSP2000] pp. 117-126.

[Hoe00] F. Hoeben "Using UML Models for Performance Calculation" in [WOSP2000] pp. 77-82.

[IW95] P. Inverardi, A.L. Wolf "Formal Specification and Analysis of Software Architectures Using the Chemical Abstract Machine Model" IEEE Transaction on Software Engineering, Vol. 21, n. 4, pp. 373-386 (1995).

[Kan92] K. Kant "Introduction to Computer System Performance Evaluation" McGraw-Hill (1992).

[KP99] P. King, R. Pooley "Derivation of Petri Net Performance Models from UML Specifications of Communication Software" Proc. of XV UK Performance Engineering Workshop (1999).

[M97] D.A. Menascè "A Framework for Software Performance Engineering of Client/Server Systems" Proc. of the 1997 Computer Measurement Group Conference, Orlando, Florida (1997).

[MG98] D.A. Menascè, H. Gomaa "On a Language Based Method for Software Performance Engineering of Client/Server Systems" Proc. of WOSP'98, Santa Fe, New Mexico, USA, pp. 63-69 (1998).

[P00] D. Petriu "Deriving Performance Models from UML Models by Graph Transformations" Tutorial in [WOSP2000].

[PW99] D. Petriu, X. Wang "From UML descriptions of High-Level Software Architectures to LQN Performance Models" in Proc. of AGTIVE'99, Springer Verlag LNCS 1779, pp. 47-62 (1999).

[PK99] R. Pooley, P. King "The Unified Modeling Language and Performance Engineering" in Proc. of IEE Software (1999).

[Poo99] R. Pooley "Using UML to Derive Stochastic Process Algebra Models" Proc. of XV UK Performance Engineering Workshop (1999).

[RS95] J.A. Rolia and K.C. Sevcik "The Method of Layers" IEEE Trans. on Software Eng., Vol. 21/8, pp. 682-688 (1995).

[SG96] M. Shaw, D. Garlan "Software Architecture: Perspectives on an Emerging Discipline" Prentice Hall (1996).

[Smi90] C.U. Smith, "Performance Engineering of Software Systems" Addison Wesley (1990).

[SW97] C.U. Smith, L.G. Williams "Performance Engineering Evaluation of Object Oriented Systems with SPE•ED" in 'Computer Performance Evaluation: Modelling Techniques and Tools' LNCS 1245 (R. Marie et alt. Eds.), Springer-Verlag, 1997.

[WS98] L.G. Williams, C.U. Smith "Performance Evaluation of Software Architectures" in Proc. of WOSP'98, Santa Fe, New Mexico, USA, pp. 164-177 (1998).

[WNPM95] C. Woodside, J. Neilson, S. Petriu, S. Mjumdar "The Stochastic rendezvous network model for performance of synchronous client-server-like distributed software" IEEE Transaction on Computer, Vol. 44, pp. 20-34 (1995).

[WOSP2000] ACM Proceedings of Workshop on Software and Performance, WOSP2000, Ottawa, Canada (2000).