STRUCTURED PERFORMANCE MODELING AND ANALYSIS FOR OBJECT BASED DISTRIBUTED SOFTWARE SYSTEMS

P. Katsaros
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, 54006, Greece
katsaros@csd.auth.gr

C. Lazos
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, 54006, Greece
clazos@csd.auth.gr

Abstract

In this paper, we address the problems related to the performance modeling of object based software systems, distributed across multiple platforms. Classical flat queuing models seem to be inappropriate, because of the inevitable complexity caused by the platform heterogeneity and the dual client/server role that objects often play in their interactions. Models should be structured vertically, in hierarchical levels, as well as horizontally, in interconnected model components. Each component may be using several analytically intractable queuing network extensions, for the precise representation of the synchronization phenomena that arise in the various object interaction cases. The overall model structure is utilized not only for incrementally specifying a complex workload and resource contention description, but also for approximately analyzing it, by successive flow equivalence aggregations. In this context, we discuss the properties that the model should possess, for achieving satisfactory levels of accuracy. The modeling of common design cases, like the synchronous object invocation, the distributed callback and the multithreading process structure, is illustrated, in the context of CORBA based object interactions. The suggested approach promotes the accomplishment of the appropriate type of analysis at the most suitable level of abstraction, in respect to the specific credibility and cost requirements of each study.

KEYWORDS: distributed objects, modeling and simulation, performance evaluation, queuing networks

1 INTRODUCTION

This paper describes a structured modeling approach for the performance analysis of object-based distributed software systems. Such an analysis aims in:

- detecting and smoothing architectural bottlenecks caused by communication costs due to object interaction,
- detecting performance critical points, where a more thorough modeling and design study should be focused,
- achieving adequate scalability, in respect to the specified application's requirements.

Each performance model is composed of an anticipated load, the corresponding computational work and an appropriate contention model description.

The load description is the result of the so-called workload characterization process, which aims in describing the system's global workload in terms of its main functional classes and their intensities. We discriminate two different types of loads: the user behavior based loads and the artificial or synthetic loads.

A computational work description reflects the system's resource demands for a single workload object. As a workload object, it is considered to be whatever represents an entity of work, in the context of the conducted modeling study; it can be a service request, a transaction or anything else. The level of detail, imposed by the selection of the elementary workload object, determines the level of detail of the performance measures to be estimated. This means that if for example the selected workload objects are service requests, then the obtained performance results will be expressed in terms of service requests.

As a contention model, a queuing or an extended queuing network, can be formed, for representing the interconnection of the available system resources in an adequate manner. The term "extended" refers to the queuing networks, that make use of a number of auxiliary nodes, which violate product form assumptions, for providing increased modeling flexibility (passive resource allocation nodes for the representation of simultaneous resource possession phenomena, fission/fusion nodes for the representation of parallel flows of execution etc).

Performance analysis of distributed systems based on one-level unstructured (flat) models is severally complex, due to the intricate structure of these systems.

In this work, we suggest the adoption of a structured modeling approach for the incremental development of performance models, structured vertically in hierarchical levels, as well as horizontally, in interconnected components. Thus, any performance model is expressed as a hierarchy of interacting contention models and a corresponding hierarchy of workload objects (job types). This approach provides a basis for:

- performing approximate, but sufficiently accurate model analyses, using appropriate simulation or hybrid simulation techniques, as for example the ones described in [21] and [2],
- convenient experimentation with different design alternatives in some parts of the model, without modifying other parts of it.

The purpose of this paper is not the provision of a complete systematic approach, but the introduction of a basic set of foundation concepts and methodological issues regarding the development of structured performance models of object based distributed software systems. The existence of a number of advanced modeling tools (like for example the ones described in [1], [16] and [8]), for the implementation and the analysis of performance models of this form, is a first measure of the feasibility of the described approach. Important considerations that are imposed by the theory of near complete decomposability ([3] and [4]) and have to be taken into account are also discussed. Finally, the modeling of common design cases, like the synchronous object invocation, the distributed callback and the multithreading process structure, is illustrated, in the context of CORBA based object interactions.

2 STRUCTURED PERFORMANCE MODELS

Structured performance modeling is achieved by associating a high-level model's component to a submodel, so that the model-submodel pair is characterized by exactly the same set of interactions. Thus, a high-level model may yield a hierarchical structure, either:

- by separating part(s) of it into a set of corresponding submodels,
- by refining model components into more detailed submodels or
- by defining distinct interacting models, which represent different levels of processing in the same system.

Whatever specification technique is to be used, there is no difference in the resulting hierarchy of models. Generally, structured modeling constitutes an attractive approach, in cases, where either:

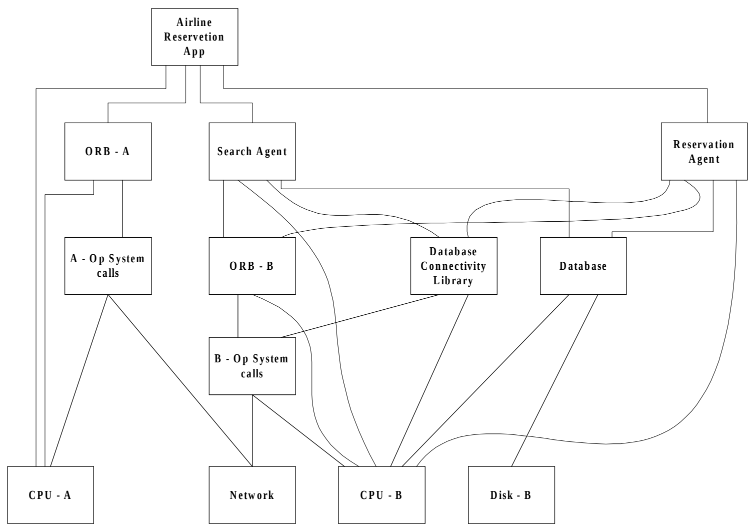
- A model's size and complexity raises understanding, management and analysis difficulties, as the system grows. When structuring a performance model vertically, in hierarchical levels and horizontally, in interconnected components, the internal state of each lower-level model is only partially transparent to its immediate higher-level counterpart. In this way, we hide information among the levels of the hierarchy and keep the model easier to understand and analyze.
- Our aim is the analysis of the same model in a number of alternative versions, each one incorporating changes to specific parts (components), without affecting the rest of it.
- The model includes a number of identical submodels, which can be then analyzed only once and directly aggregated in the rest of cases.
- Different solution techniques or modeling paradigms are to be used for the performance submodels.

Structured modeling provides a basis for:

- the adoption of a top-down model specification procedure and
- for the approximate bottom-up analysis of the resulting model, by successive composite submodel aggregations, in the form of flow-equivalent service centers.

Each submodel in the structure receives jobs from the higher-level model(s), processes them according to well defined processing patterns, called services, and when a service request to a lower-level component is placed, then a job is sent to the corresponding lower-level submodel. Thus, each composite submodel provides a set of services and makes use of services provided by lower-level composite submodels and a number of basic components, which can be either queuing or delay centers. Depending on the modeling environment to be used, each service is described either by an appropriate algorithmic language or by a graphical or textual (extended) queuing network notation. The HIT tool [1], for example, utilizes the first approach, while the QNAP2 [16] and the RESQME [8] modeling environments, the second one.

Since each model component is possible to provide more than one service, it is actually considered as serving a number of different job classes. Thus, the overall performance model is eventually expressed as a set of hierarchically interacting multichain queuing models, with service centers that can impose service demands either to lower-level composite models or to predefined basic components, which represent the system's hardware resources and the network.



Airline Reservation App	search_for_flight	Search Agent	search_flight
	search_for_seats_on_flight		search_seats
	search_for_active_reservation		search_reserv
	make_reservation	Reservation Agent	mk_reserv
	cancel_reservation		canc_reserv
ORB – A, ORB – B	send_method_invoc	A – Op System calls,	
	receive_method_invoc	B – Op System calls	memcpy, writev,
	send_response		etc
	receive_response	Database	process_query
CPU – A, CPU – B	exec_instr	Disk – B	disk_access
Network	transfer_msg		

Figure 1 A structured performance model of a CORBA based airline reservation system and some of the services provided by each one of the model's components

Our framework suggests using platform independent service demand descriptions for the service calls to the system's hardware and network resources. Thus, machine instructions are an appropriate way for quantifying CPU use, as well as visit counts are for I/O activity and sent messages for network resource demand descriptions. As a consequence, the hardware service demands are fully determined by the average time spent on each visit at the resource and this allows easy model parameterization in different hardware configuration cases. Moreover, services provided by composite (sub)models should be possible to be described parametrically, based on appropriate fitted data and environment dependent functions, in order to account for the varying conditions under which they are executed. As regarding, the system's load scenarios, they are usually specified as a set of workload intensities for each one of the services provided by the model's top-level component.

In accordance to the modeling concepts discussed so far, Figure 1 introduces the performance model of a sample airline reservation application.

Structured modeling also provides, as it has been already noted, a basis for applying the most suitable submodel solution techniques and approximately aggregating them, in the form of flow-equivalent service centers, for analyzing the model in higher levels of abstraction. Even if one or more submodels must be solved using simulation, such an analysis is likely to yield important computational savings, since, in each submodel aggregation, a large number of events is replaced by a single event for the high-level model analysis. The credibility of the employed evaluation procedure is strongly dependent on the modeling environment to be used. However, there are also some important considerations, imposed by the theory of near complete decomposability, that have to be taken into account. More precisely, a model is said ([3] and [4]) to be near completely decomposable, if it is made up of submodels that always reach an internal equilibrium between external interactions. Analysis of such models by decomposition and aggregation guarantees a bounded error ε, which depends on the highest level of coupling between the submodels. Thus, the choice of the partition of the submodels is crucial, in order to obtain decompositions with small ε. Equally important, the theory does not prescribe the solution techniques to be used for the submodel and the aggregated model analyses.

Each submodel is solved for all possible job populations (and job class combinations) that can arise. The technique of short-circuiting is applied, such as any job that leaves the submodel is immediately reentered into that. The average residence times between entering and leaving the submodel are collected for all job classes and they are used to approximate the performance of the flow-equivalent, load dependent aggregate center in the high-level model analysis.

However, it is obvious, that analysis of submodel hierarchies is only possible, if a separate computational work derivation procedure is to be employed. Such a procedure aims in the model's load dependent generation of the computational work devolved on the sub-hierarchy's top-level component. An easy to apply alternative is the Structure and Performance (SP) method ([24]). According to this, for any model component *k*, the complexity matrix C_{kr} specifies the average number of times that each of the services provided by the lower level component *r* is accessed, when each of component *k*'s provided services is executed. Thus, let us consider,

$$C_{kr} = \begin{array}{c} \\ pr\_service1 \\ . \\ . \\ . \\ pr\_serviceN \end{array} \begin{array}{c} \begin{array}{ccccc} us\_service1 & . & . & . & us\_serviceM \end{array} \\ \begin{bmatrix} c_{11} & . & . & . & c_{1M} \\ . & . & & & . \\ . & & . & & . \\ . & & & . & . \\ c_{N1} & . & . & . & c_{NM} \end{bmatrix} \end{array}$$

where c_{ij}, $1 \le i \le N$, $1 \le j \le M$, is the average number of times that service *j* - provided by the lower level submodel *r* - is accessed, every time that service *i* is called. The computational work W_{kl} devolved on any specific component *l*, by service requests originated from a possibly non-adjacent higher-level component *k*, is the sum of the products of the complexity matrices of all possible intervening component paths. Thus, for the sample application presented in Figure 1, the computational work devolved on the B-OpSysCalls component is given as:

$$\begin{aligned} C_{AirResApp-BOpSysCalls} = \; & C_{AirResApp-SearAgent} \cdot C_{SearAgent-ORBB} \cdot C_{ORBB-BOpSysCalls} + \\ & C_{AirResApp-SearAgent} \cdot C_{SearAgent-DBConLib} \cdot C_{DBConLib-BOpSysCalls} + \\ & C_{AirResApp-ResAgent} \cdot C_{ResAgent-ORBB} \cdot C_{ORBB-BOpSysCalls} + \\ & C_{AirResApp-ResAgent} \cdot C_{ResAgent-DBConLib} \cdot C_{DBConLib-BOpSysCalls} \end{aligned}$$

Near complete decomposability, implies that application of successive bottom-up aggregations, can only take place ([5]) at the levels of the model's hierarchy, where either,

- weak interactions between the adjacent model components are observed, or

- the time scale of the occurrence of the events in the higher-level component and the events in the lower-level one is very different.

Generally, in computer system models, it is true that consumption of the so-called short-time resources occurs at a rate, which is typically much greater than the changes observed in the set of active jobs in the model. However, the basic aforementioned rules have always to be taken into account, if a bottom-up aggregation analysis procedure is to be used. Accumulated small aggregation errors may have disastrous effects for the overall accuracy. Therefore, it is important for the modeling environment to be used, to offer the possibility to obtain confidence interval and error bound estimates.

Some other considerations that have also to be taken into account are:

- To not apply decomposition to submodel hierarchies with multiple entries, especially when the selection of the entry point is dependent on events outside of the submodels.

- To avoid shaping submodels, which contain events, requiring synchronization with events external to their submodel hierarchy.

- To avoid shaping submodels with long execution times, since analyzing them by simulation is not likely to yield important computational savings.

3 PERFORMANCE MODELING OF OBJECT BASED DISTRIBUTED SOFTWARE SYSTEMS

Traditional one-level unstructured models do not constitute an adequate means for the credible representation of object based distributed systems, mainly because of:

- The lack of features that promote the combined modeling of hardware and software contention as for example the contention observed for accessing software servers with a limited number of pseudo-concurrent threads or for accessing locks on files, database tables etc.

- The lack of features for the representation of the dual client/server role that objects often play in their interaction.

- The lack of features for the representation of commonly used design patterns such as the callback interaction and the recursive object invocation cases.

However, we believe that the modeling framework described in the previous section is certainly addressing the first one of the aforementioned limitations and it is also amenable to appropriately chosen extensions for addressing the other two.

We suggest the use of a number of well known ([14]) extended queuing modeling constructs, for the representation of blocking, simultaneous resource possession and parallel execution flows phenomena. Their use is illustrated in the context of CORBA-based object interaction cases and is supported by the modeling tools mentioned so far.

More precisely, a set of four auxiliary nodes has been found necessary, to allow process blocking and simultaneous resource possession within model components. An allocate node (Figure 2a) models acquisition of (a number of instances of) a resource. The available instances of the resource are modeled as a pool of tokens. If too few instances of the resource are available to a transaction, it is forced to wait in the queue. A release node (Figure 2b) returns the acquired resource(s). A create node (Figure 2c) produces a number of instances of the resource, while a destroy node (Figure 2d) consumes them.

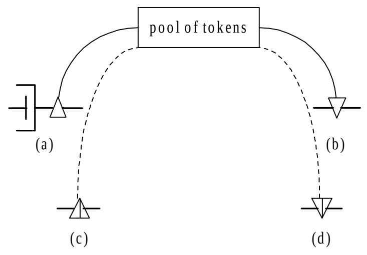
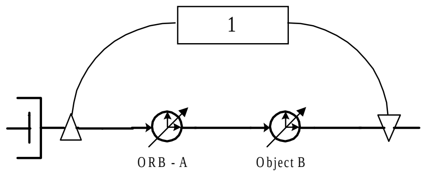


Figure 2 The set of simultaneous resource possession modeling constructs

Simultaneous resource possession is a commonplace in CORBA-based distributed systems, since their basic communication primitive is the synchronous object invocation, which causes blocking of the caller until it gets the reply. Furthermore, nested invocations are also used quite frequently, causing several objects to be successively blocked, unless multi-threading is used in the object implementations. Figure 3 illustrates the representation of a synchronous object invocation case.



ORB – A:	a flow equivalent delay representation of a send_method_invoc service request to a lower level ORB model component
Object B:	a flow equivalent delay representation of a service request to a lower level server object

Figure 3 Synchronous object invocation using simultaneous resource possession constructs

Asynchronous, one-way object invocation is also possible in CORBA-based software architectures, but it is most often used as part of the callback interaction pattern. In this case, the client object passes, as a parameter of the call, a reference to a callback handler and continues processing. When the invoked object implementation is ready to return the result of the method call, then a new one-way invocation of the callback handler is issued. The parallel execution flows of the callback's interaction pattern may be easily represented using the well-known job fission and fusion auxiliary nodes (Figure 4).

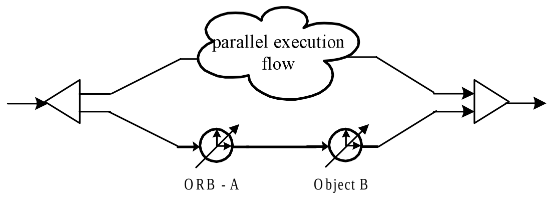


Figure 4 The callback interaction pattern

Simultaneous resource possession constructs (Figure 2) are also an appropriate means for the representation of most common threading policies. The most important of them have been experimentally studied in [20]. Thread pool policies, with a fixed number of threads, can be easily represented by selecting the number of tokens, in the modeling construct of Figure 2, to be equal to the number of available threads. Dynamic thread creation and destruction policies can be also easily represented, using the create and destroy auxiliary nodes of the same figure.

The level of detail of a structured performance model should be adjusted such that:

- its components correspond to the software components that are most likely to be changed or exchanged,
- the model is robust with respect to possible modifications,
- the decomposition accuracy considerations mentioned in the previous section are taken into account,
- it is always possible to provide hardware resource demand measurements or estimates in terms of component service requests and to satisfy the contention model's parameter needs.

However, it is often desirable and hence it should be also possible to parametrically describe service demands in terms of data or environment dependent factors. It has been found

([6] and [7]) for example, that in CORBA based distributed software systems,

- invocation marshaling/unmarshaling costs are mainly determined by the amount and the types of the transmitted data and

- invocation demultiplexing costs, which also constitute a considerable amount of the total invocation costs, are strongly dependent on the number of methods declared in the corresponding IDL interface and the number of active object implementations in the same platform.

Structured performance models should be exploited for conducting the appropriate type of analysis at the most suitable level of abstraction, in respect to the specific credibility and cost requirements of each study. Some of these types of analysis, which are more thoroughly discussed in [12], are:

- The bottleneck analysis, which aims in detecting the most utilized components that cause performance degradation in high load situations and are worth to be analyzed – redesigned in a lower level of abstraction.

- The sensitivity analysis, which aims in delivering means for the fast calculation of performance changes in respect to various (combinations of) model parameter changes. A metamodeling based sensitivity analysis approach is described in [11] and it can be easily adapted for use either for bounds analysis, parametric analysis, scalability analysis and system optimization purposes.

Performance modeling and analysis of object based distributed software systems has been recently become the subject of a number of important research activities. Thus, [23] introduces a methodology for transforming distributed object based designs into performance models that fit into the well-known software performance engineering (SPE) approach. Two separate notations are proposed: execution graphs that represent the software structure and information processing graphs, which are a form of extended queuing networks that model the overall system performance. The later are calibrated in respect to the former.

In [10], the author introduces a hierarchical model structure based on the so-called augmented queuing networks that support the use of simultaneous resource possession characteristics. The resulted model is decomposed into a number of interdependent product form multi-class queuing networks that are iteratively analyzed by an algorithm based on successive surrogate delay approximations ([9]).

The layered queuing network (LQN) formalism is another hierarchical representation that allows a software component to act simultaneously as a client to a number of software components or hardware resources and as a server to other software components. The proposed evaluation algorithms ([18] and [25]) are based on the iterative computation of a series of intermediate results, up to the time instant, where the estimated results converge.

A similar layered queuing network model is proposed in [17] for representing client-server systems where communication is carried out with synchronous and asynchronous messages. The model makes use of blocking queuing centers for representing synchronous communication phenomena.

An appropriate flow-equivalence algorithm for analyzing multi-layered models is introduced and evaluated in [13].

Similar to our structured modeling approach, that features a high degree of modeling flexibility at the cost of being restricted to only pure or hybrid analysis procedures, are, the recent work of [19] and the ongoing one, described in [22].

4 CONCLUSIONS AND FURTHER WORK

In this article, we presented a set of foundation concepts and theory and methodology issues regarding the structured performance modeling of object-based distributed systems. Such an approach provides a basis for a top-down stepwise model development, as well as, for its approximate analysis by decomposition at the most appropriate points of the model's hierarchy.

The described conceptual framework is to be integrated with a well-established and extensible modeling language. This language has to be characterized by easiness in representing hierarchical model structures and a relative proximity to the systems' functional descriptions. For these reasons, we will consider, in future work, the applicability of the recent extensions of the Unified Modeling Language (UML) that are specified in [15].

5 REFERENCES

[1] H. Beilner, J. Mater, N. Weiβenberg, "Towards a performance modeling environment: News on HIT", Proc. of the 4th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation, Palma de Mallorca, Plenum Publishing Corporation, pp. 57-75, 1988.

[2] A. Blum, L. Donatiello, P. Heidelberger, S. Lavenberg and E. A. MacNair, "Experiments with decomposition of extended queueing network models", In D. Potier (ed.): Modeling Techniques and Tools for Performance Analysis, Elsevier Science (North-Holland), pp. 623-640, 1985.

[3] P. Courtois, “Decomposability, instabilities, and saturation in multiprogramming systems”, Communications of the ACM, vol. 18 (7), pp. 371-377, 1975.

[4] P. J. Courtois, Decomposability: Queuing and Computer System Applications, ACM Series, Academic Press, New York, 1977.

[5] P. Courtois, “On time and space decomposition of complex structures”, Communications of the ACM, vol. 28 (6), pp. 590-603, 1985.

[6] A. S. Gokhale and D. C. Schmidt, “Evaluating CORBA latency and scalability over high speed ATM networks”, Proceedings of the International Conference of Distributed Computing Systems, Baltimore, Maryland, 1997.

[7] A. S. Gokhale and D. C. Schmidt, “Evaluating the performance of demultiplexing strategies for real-time CORBA”, Proceedings of the GLOBECOM’97, Phoenix, AZ, 1997.

[8] K. J. Gordon, R. F. Gordon, J. F. Kurose and E. A. MacNair, “An extensible visual environment for construction and analysis of hierarchically – structured models of resource contention systems”, Management Science, vol. 37 (6), pp. 714-732, 1991.

[9] P. A. Jacobson and E. D. Lazowska, “Analyzing queuing networks with simultaneous resource possession”, Communications of the ACM, vol. 25 (2), pp. 142-151, 1982.

[10] P. Kahkipuro, Performance modeling framework for CORBA based distributed systems, PhD thesis, Department of Computer Science, University of Helsinki, Finland, 2000.

[11] P. Katsaros, E. Angelis and C. Lazos, “Applied multiresponse metamodeling for queuing networks experiments: Problems and perspectives”, Proceedings of the 4th International EUROSIM Congress, Eurosim, Delfts, The Netherlands, 2001.

[12] P. Katsaros, Performance analysis of distributed software systems, PhD thesis (in Greek), Department of Informatics, Aristotle University of Thessaloniki, Greece, 2002.

[13] T. Kurasugi and I. Kino, ”Approximation methods for two-layer queuing models”, Performance Evaluation, vol. 36-37, pp. 55-70, 1999.

[14] S. S. Lavenberg (Ed.), Computer Performance Modeling Handbook, Academic Press, 1983.

[15] OMG document number: ad/2001-06-14, Response to the OMG RFP for Schedulability, Performance and Time, B. Selic and A. Moore (ed.), ARTiSAN Software Tools, Inc., I-Logix Inc., Rational Software Corp., Telelogic AB, TimeSys Corporation, Tri-Pacific Software, 2001.

[16] D. Potier and M. Veran, “QNAP2: a portable environment for queuing systems modelling”, In Proc. of the 1st International Conference on Modeling Techniques and Tools for Performance Analysis, Amsterdam, pp. 25-63, 1984.

[17] S. Ramesh and H. G. Perros, “A multilayer client-server queuing network model with synchronous and asynchronous messages”, IEEE Transactions on Software Engineering, vol. 26 (11), pp. 1086-1100, 2000.

[18] J. A. Rolia and K. C. Sevcik, ”The method of layers”, IEEE Transactions on Software Engineering, vol. 21 (8), pp. 689-700, 1995.

[19] N. N. Savino-Vazquez, J. L. Anciano-Martin, S. Dumas, J. A. Corbacho, R. Puigjaner, D. Boudigue and G. Gardarin, “Predicting the behavior of three-tiered applications: dealing with distributed-object technology and databases”, Performance Evaluation, vol. 39, pp. 207-233, 2000.

[20] D. C. Schmidt, “Evaluating architectures for multi-threaded CORBA object request brokers”, Communications of the ACM, vol. 41 (10), pp. 54-60, 1998.

[21] H. D. Schwetman, “Hybrid simulation models of computer systems”, Communications of the ACM, vol. 21 (9), pp. 718-723, 1978.

[22] D. Smarkusky, R. Ammar, I. Antonios and H. Sholl, “Hierarchical performance modeling for distributed system architectures”, Proceedings of the 5th IEEE Symposium on Computers & Communications, 2000.

[23] C. U. Smith and L. G. Williams, Performance solutions: A practical guide to creating responsive, scalable software, Addison Wesley, 2001.

[24] V. Vetland, P. Hughes and A. Solvberg, “Improved parameter capture for simulation based on composite work models of software”, In Proceedings of the 1993 Summer Computer Simulation Conference, Boston, Massachusetts, pp. 110-115, 1993.

[25] M. C. Woodside, J. E. Neilson, D. C. Petriu and S. Majumdar, “The stochastic rendezvous network model for performance of synchronous client-server-like distributed software”, IEEE Transactions on Computers, vol. 44 (1), pp. 20-34, 1995.