PERFORMANCE ANALYSIS OF SYSTEM MODELS WITH UML AND GENERALIZED NETS

Evelina Koycheva[1], Klaus Janschek[1]

[1]Institute of Automation, Technische Universitaet Dresden,
Mommsenstrasse 13, 01062 Dresden, Germany
klaus.janschek@tu-dresden.de(Klaus Janschek)

Abstract

Functional correctness and verified performances of automation solutions are mandatory system requirements to guarantee a certain quality of service. Performance verification in the early design phases can reduce considerably the project costs. The current paper presents a new approach for performance evaluation of automation systems using UML for system modeling and performance specification and applying Generalized Nets (a variant of timed Petri Nets) for a simulation based performance evaluation. Standard UML modeling techniques are used to specify the basic automation system functions, their hardware/software allocation and the interaction with the technical process. Specific performance parameters and the interaction with the human operators are incorporated in the UML models through the standardized Profile for Schedulability, Performance and Time (SPT-profile). The resulting annotated UML models are automatically transformed to Generalized Nets via XML style sheets. Monte Carlo type time simulations can be performed with the Generalized Nets system models to derive representative performance measures. The paper gives an overview on the used UML and SPT-profile properties for performance modeling and specification, it introduces briefly the Generalized Nets concept and it describes thoroughly the transformation approach and the implementation in an XML-based framework. Results from a case study show the practical potentials of the proposed approach.

Keywords: system design, performance analysis, UML, Petri nets

Presenting Author's biography

Klaus Janschek is managing director of the Institute of Automation and holds the chair of Automation Engineering in the Department of Electrical Engineering and Information Technology at Technische Universität Dresden. Main research topics: automation systems design, tele-automation, mobile robotics, navigation, optical data processing.

ISBN 978-3-901608-32-2 *Copyright © 2007 EUROSIM / SLOSIM*

1 Introduction

Automation systems design requires an integrated view on abstract and heterogeneous system models which are describing different system properties (physics, hardware, software, human operation) and system behavior (continuous/discrete time, discrete event). Moreover increased system complexity makes these models even less transparent and in consequence more subject to design and specification errors. Beside functional consistency and integrity also the guarantee of performance characteristics such as response times, throughput and utilization is mandatory. It is well known, that recognizing performance deficits in the early design phases, in particular before the implementation and the procurement of the necessary hardware, is of highest importance to avoid cost and schedule overruns. Key issues for modern systems engineering are therefore

- appropriate modeling and specification methods for complex heterogeneous systems;
- automated tools and frameworks for practical engineering work;
- representative methods for performance evaluation.

The current paper presents a new approach for an automated simulation based methodology for performance evaluation using standardized UML system models realized within a XML-based automated framework.

The paper is organized as follows. Chapter 2 introduces the application of UML for automation systems modeling. Chapter 3 discusses state-of-the-art methodologies for performance evaluation and introduces the Generalized Nets as a very promising and appropriate concept for the verification task in question. The detailed performance modeling approach is described in Chapter 4 and its implementation in a XML-based automated framework is outlined in Chapter 5. A case study including performance related annotations is presented in Chapter 6.

2 UML based automation system modeling

In the early phases of product development only abstract models of the anticipated system are available. To accomplish a representative performance analysis, the entire automation system model must incorporate at least the following subsystems and environment components:

- system and application software;
- hardware platform, on which the modeled software runs;
- behavior of the human operators ("actors");
- properties of the technical process (including attached sensors, actuators, etc.).

All enumerated components including their performance characteristics form the system performance model (Figure 1).

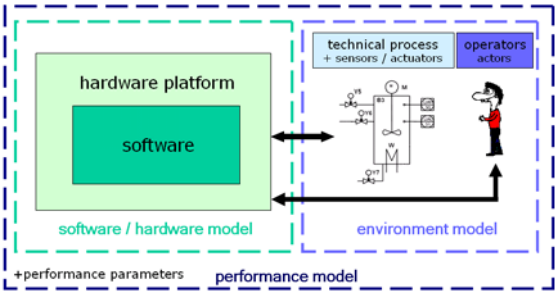

Figure 1. Automation system decomposition and mutual relations between components

2.1 Software model

Object oriented software development has been established today as the standard approach for automation solutions. In this context UML - *Unified Modeling Language* has proved to be one of the most powerful and meanwhile most popular modeling languages for object oriented specification and design of software systems [1]. The big success of UML is based on its broad pallet of modeling possibilities, the consistency of the approach, the excellent tool support as well as the intuitive clarity of its graphic models, which makes them usable also for specialists from different non-software application areas. Software properties, which are relevant for the desired goals of the performance analysis, are modeled in UML most frequently with *activity* / *sequence* / *communication diagrams* and *state charts* [1].

2.2 Hardware model

A unified model, which contains all necessary component and subsystem models and which is based on a uniform modeling techniques, offers a set of advantages. The first advantage is that the system designers have to be familiar with only one single modeling technique. Moreover unified models are more comprehensive and they help reducing the probability of design and specification errors. As today the software design is commonly specified in UML notation, it is straightforward using UML as hardware modeling language as well. Hardware nodes and their relations and properties can be covered conveniently by the UML *deployment diagram*.

2.3 Performance model

Pure UML notation does not allow a specification of comprehensive performance properties. With the adoption of the recently introduced UML extension

 Copyright © 2007 EUROSIM / SLOSIM

Profile for Schedulability, Performance and Time (SPT-profile, January 2005, [2]), it is more easily possible to include performance relevant system properties and requirements directly into the UML design model.

2.4 Environment model

The SPT-profile also provides *stereotypes* and *tagged values* for modeling of open and closed *workloads*, which can be used for illustrating the number and the behavior of the system operators and other interacting entities (see Fig. 1).
On the other hand UML contains specific model elements like *external signals*, which allow the modeling of the behavior of the underlying technical process and its different entities. Additional external parameters, which are related to the automation process, can be modeled by *OCL-expressions*, a further standardized UML extension.

The most common methods for performance analysis can be divided into two large groups: analytical methods and simulation based methods. The most frequently used *analytical methods* are based on Markov Chains [3], Execution Graphs [4, 5] and Queuing Networks (QN, [3, 6]) and their extensions Layered QN (LQN, [7, 8, 9]) or Extended QN (EQN, [10, 11, 12]). In addition there are available well-known approaches on basis of Process Algebras [13, 14] and Stochastic Petri Nets [15, 16]. The *simulation based methods* use normally Execution Graphs, Hi-Level Petri Nets or proprietary tool-based approaches [17, 18].
Although analytical methods are preferable to get most reliable performance measures with highest validity, their practical applicability is limited to more or less simplified and standardized performance properties. A more general approach, allowing the specification of arbitrary performance properties, is given by simulation. Due to the

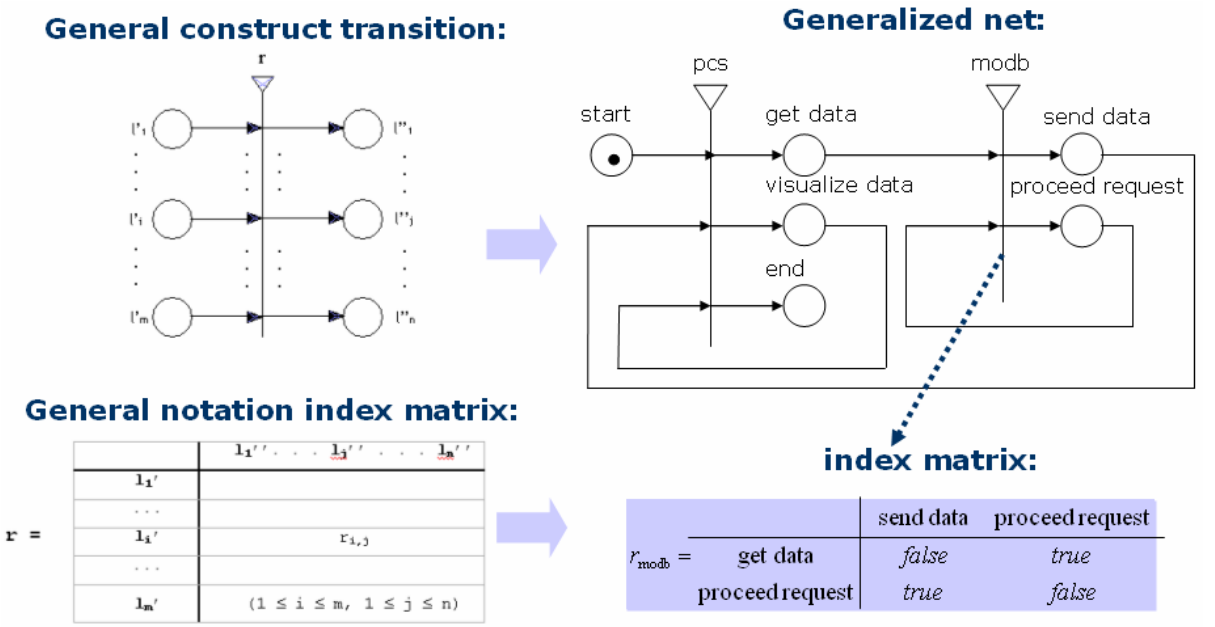

Figure 2. Generalized Nets modeling capabilities

3 Introduction to Generalized Nets

Although UML offers a large variety of modeling possibilities, the lack of a seamless simulation or performance analysis of the annotated UML models remains an open problem. Two reasons can be identified as main problem sources: UML models do not represent executable models UML is actually not a strict formal specification, which makes the analysis extremely difficult. To overcome these hurdles, a *transformation* of already existing UML design models into another model domain, which permits a strict model verification of the model, is a promising approach.

inherent discrete event nature of automation solutions, Petri Nets [19] have been successfully used for the modeling, simulation and analysis tasks since many years in the automation community. However the application of the classical Petri Nets shows considerable weaknesses like fast growing size and associated vagueness of the model. In order to overcome these disadvantages, numerous modifications of the Petri nets have been developed in the past years. Various pilot investigations [20, 21] have shown, that so called Generalized Nets (GN), a special kind of the Hi-level Petri nets, are very well suited and powerful for systems engineering and performance analysis tasks. Therefore we have adopted the Generalized Nets

 Copyright © 2007 EUROSIM / SLOSIM

[22] as methodological baseline for performance modeling and performance evaluation.

Generalized Nets are a generalization of several modifications of Timed Petri Nets [19]. Among the most important differences between Generalized Nets and other classes of Petri Nets should be mentioned the definition of a generalized transition object, which includes the transition symbol, all appertaining input and output places as well as several index matrices (see Fig. 2). One index matrix defines the capacities of the binding arcs for each transition, another index matrix represents the transition predicates. The evaluation values of the latter index matrix elements determine the direction of the token flow from the input to the output places. The tokens of a GN are in general distinguishable instances, which enter the net with particular initial characteristics. During their travel through the net, these tokens are acquiring further properties, representing historical information. Furthermore the time step for the token movements in a Generalized Net can be selected at any time scale. Generalized Nets incorporate also numerous operators (e.g. hierarchies, reduction). Some other differences are for instance dedicated capacities for edges, places and tokens as well as priorities for transitions, places and tokens.

In the illustrating example in Figure 2 the Generalized Net owns two transitions – `pcs` and `modb`. Transition `pcs` contains three input places – `start`, `visualize data` and `send data` – and three output places – `get data`, `visualize data` and `end`. Every place has always only one input and one output arc, which connects the place with the corresponding transition. The places `start` and `end` are at the same time also input and output for the whole net. In the input `start` tokens are generated and the output place `end` collects the tokens, which already finished their movement in the net (if the conditions for the appropriate movements are fulfilled as required). The index matrix shown in Fig. 2 outlines the predicates, which control the movement from the input places to the output places of the transition `modb`. For every input place one row is built in the index matrix and equivalently for every output place one column is built. The elements in the matrix determine the movement between certain places. A predicate can be any arbitrary logical expression, for simplifying reasons here we have shown simple `true` and `false` expressions. The evaluation of the predicates takes place always when the appropriate transition is activated. For the illustrating example each transition is activated at each operation time step of the net and the activation duration is one step. Thereby all tokens, which are in input places can move to the output places of the same transition as soon as a connection exists and the appropriate predicate for this movement is evaluated equal `true`. Anytime when a token arrives at a new place, its characteristics are complemented. This can be defined accordingly in any form.

As a simple example for the represented net it is assumed, that one token should be generated in the net input at time step 0. This token will move in each step to another place in the net in the following sequence: `start – get data – proceed request – send data` (see predicates in the index matrix) - `visualize data – end`. If the net runs more steps as required for these movements, the generated token will remain in the output place `end` for the rest of the time. At each move of the token the actual time step and the actual place can be added to the characteristics of the token, e.g. „...On time step 1 in place `get data`…“. So the whole history of the token movements is collected and can be analyzed subsequently.

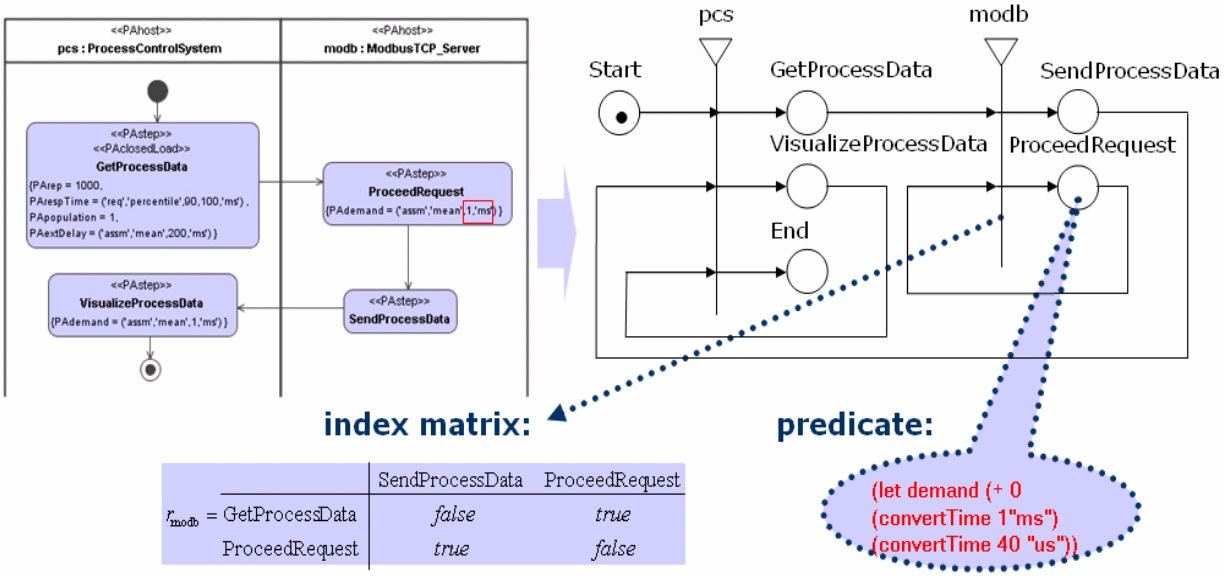

Figure 3. Annotated *activity diagram* and corresponding Generalized Net

4 Performance modeling approach

The proposed approach for system performance modeling and evaluation is starting with the establishment of an UML system model. As outlined in chapter 3, the distribution of software components on hardware nodes can be modeled with UML by deployment diagrams. The software-hardware-design has to be extended by specifying explicit performance parameters. This can be done by annotating the UML model with elements from the SPT-profile in accordance with the specified assignment rules.

The annotation of the UML model refers to both diagram types - the software and the hardware relevant diagrams. After the annotation with all desired and necessary performance parameters, the resulting complete design model can be transformed into a set of Generalized Nets. Usually the number of the Generated Nets is equal to the number of existing UML-diagrams. General transformation rules for the UML sequence diagram (in the earlier version 1.4, without SPT-profile annotation) are presented in [23].

Figure 3 shows a simple example of an annotated UML activity diagram and the equivalent Generalized Net, resulting from the transformation. The most important transformation rules applied for the shown activity diagram are as follows:

- A transition in the Generalized Net is generated for each partition (swim lane) of the activity diagram.
- One output place for the corresponding transition and its binding arcs are created for each action in the partition.
- One input place for the start node and one output place for the final node and their binding arcs are generated for the relevant transition.
- Index matrices are generated for all transitions. On each position in the matrix, which represents a possible path (direct connection between the actions in the activity diagram), "`true`" is registered. If additional conditions are present, they are registered to the respective place in the matrix instead of "`true`". Remaining elements are set to "`false`".
- The UML-tag `PApopulation` determines the amount of generated tokens.
- If the tag `PAdemand` for the stereotype "`PAstep`" is defined, the predicate of the corresponding place takes into account the indicated delay (see annotated text in Fig. 3).
- After `PAextDelay` steps a new iteration, i.e. a new execution cycle of the net, is initiated.
- Altogether `PArep` repetitions of the action are realized (in the current case 1000 repetitions of the action `GetProcessData`).

The transformation of the UML design model can be performed in one or two steps. With the 1-step-transformation the UML elements and the SPT-annotation are regarded as one whole entity. The 2-step-transformation splits the transformation process into two sequential steps. First a transformation of the standard UML elements (without any annotation) into Generalized Net elements is performed and the resulting GN-model is augmented in a second step in accordance to the SPT-annotation.

The second method offers much better opportunities for modularization of the transformation rules as well as the option of replacing the Performance Subprofile with other UML-specified profiles, for example the Schedulability Subprofile (also part of the SPT-profile) or the UML Testing profile (see [24]). These properties make the 2-step-transformation much more attractive from the systems engineering point of view and it is therefore used as the preferred approach for our research, despite of the much higher complexity of the transformation rules.

Unfortunately the SPT-annotation does not specify all necessary system parameters required for an executable model. To close this gap, the following enhanced specifications have to be added to gain a complete set of transformation rules:

- *Completion* of the generated GN-model by defining a suitable elementary time step for the simulation, as well as the definition of place capacities or token priorities;
- *Restrictions* according to the modeling freedom offered by UML, in order to get an unambiguously interpretable GN-model - the transformation rule specifies clearly the only one, namely the most probable, implementation variant;
- *GN-Simulation parameters*, e.g. external signals from the technical process or internal variables, which affect the evaluations during the simulation.

After augmentation of the system model the simulation experiments with the GN model can be accomplished. Sufficiently many statistics can be collected by Monte Carlo simulation, so that a representative analysis of the system behavior is possible. Following the evaluation of performance properties, the assessment of demanded response times or the desired range for utilization or throughput for used resources can be performed. Furthermore the behavior of the system with a certain frequency of externally produced events can be analyzed and bottlenecks in the modeled system behavior can be located.

ISBN 978-3-901608-32-2 Copyright © 2007 EUROSIM / SLOSIM

5 Automated UML-GN framework

The outlined UML-GN system performance analysis approach has been implemented in an automated XML (eXtensible Markup Language, [25]) based framework (see Fig. 4). It interconnects standard UML-Toolsets with a proprietary GN-simulator via specific XML meta-models. For the UML model the format XMI (*XML for Metadata Interchange* [26]) is used, which is prescribed by the OMG (Object Management Group). The GN-model is represented by a proprietary XML-format, called XGN (*XML for Generalized Nets*). The transformation of XMI into XGN takes place through XML style sheets (XSL, [27]).
The GN-based simulation is accomplished with the powerful *GNTICKER* simulator for Generalized Nets [28], which was developed in co-operation with the Bulgarian Academy of Sciences. The simulation experiment results can be used twofold. Once they can be forwarded to a postprocessor to evaluate the experiment statistics and visualize further performance metrics for the user (outer loop in Fig. 4). It is also possible to encode some of the experiment results into the SPT-format and feed them back directly to the primary UML-model. This feedback path allows checking directly and automatically the fulfillment of the performance requirements (inner loop in Fig. 4)

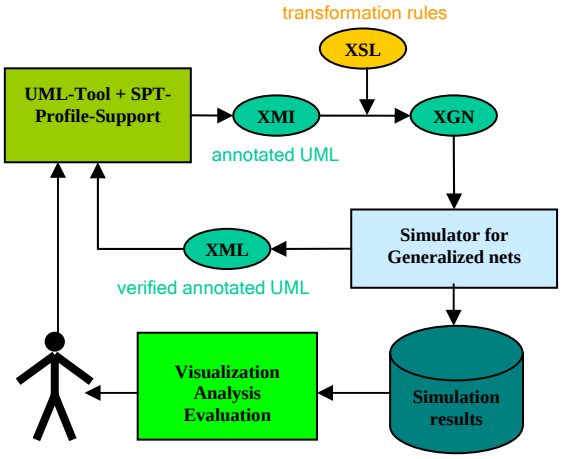

Figure 4. UML-GN framework architecture

6 Case study results

A case study shall illustrate the complete procedure of the presented approach including the transformation rules. The case study represents a distributed web-based automation application. A conventional browser without additional plug-ins communicates via internet with a web service, which implements the OPC-XML-DA specification [29]. The web service gets current process data from a ModbusTCP server, which is attached as auxiliary module of a PLC. Figure 5 illustrates the software components, their distribution on hardware nodes and some performance specific parameters in SPT-format, which are necessary for the further performance analysis. The essential communication between the software components takes place in two cycles. In the first cycle the OPC-XML-DA service polls the ModbusTCP server for current process data and stores it in its cache. In a second cycle the operator behind the internet browser receives selected data from the cache of the web service. The obtained data are visualized appropriately and changes in the process get animated if necessary.

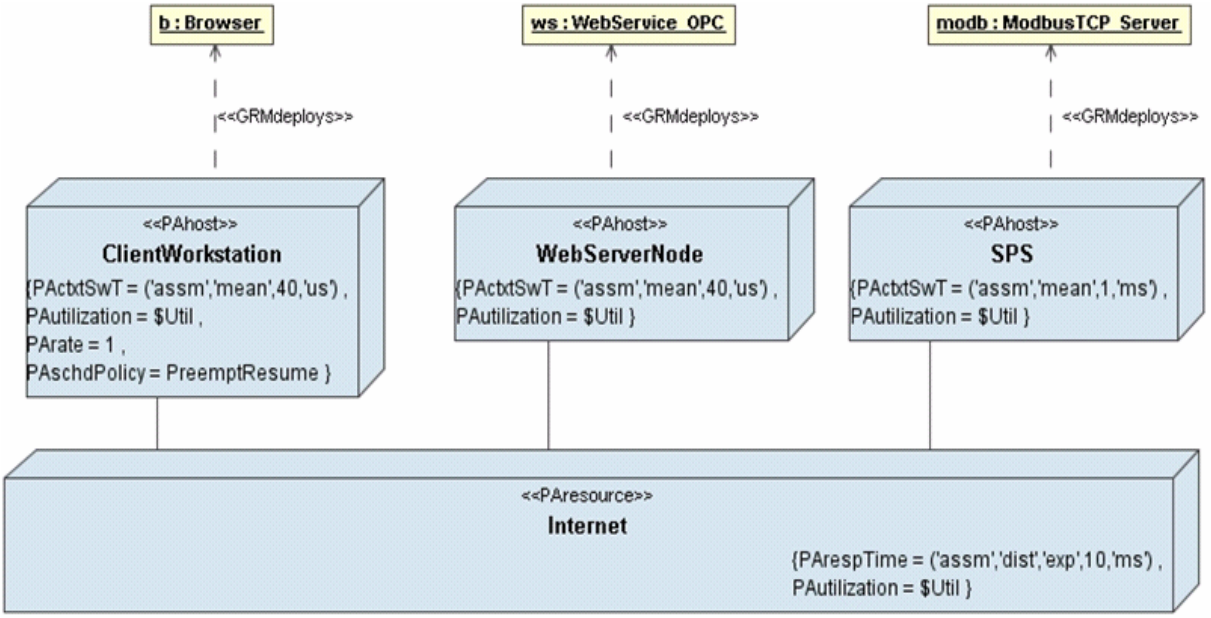

Figure 5. Case study system architecture

 Copyright © 2007 EUROSIM / SLOSIM

The representative functionality for this scenario is modeled by a UML activity diagram including swim lanes (roles). The particular steps in the activity diagram are augmented by SPT-annotation in an equivalent manner as shown in the deployment diagram (Fig. 5). An additional requirement for the excess of a time interval between two consecutive visualizations is set.

From this annotated UML model a Generalized Net has been automatically generated and simulated by the *GNTICKER* simulator for Generalized Nets. The obtained results are compared with the measured values from a prototype implementation. The fulfillment of the requirement is examined and the detailed reasons for an eventual non-fulfillment are analyzed.

Fig. 6 represents the explained scenario as UML activity diagram. The numerical parameter values are predominantly based on measurements of different prototypes. The case study demands, that two successive visualizations, i.e. two successive executions of the action `VisualizeProcessData` take place within 30 ms in at least 90% of the cases. This requirement serves in the following as reference parameter for the comparison of the measurements of the real solution and the simulation results.

The measurements with the real prototype solution led to the following summarized result:

- **67%** of the measured intervals were below 30 ms – mean value for the interval **16 ms;**
- **33%** of the measured intervals were over 30 ms – mean value for these intervals **32 ms.**

The simulation of the automatically generated Generalized Net on the other side can be summarized as follows:

- the repetition interval between two visualizations in **86%** of the cases was below 30 ms with mean value for the interval **17 ms;**
- in **14%** of the cases the measured intervals were over 30 ms – mean value for these intervals **38 ms.**

With the real solution a very small variance was remarkable. This is due to the measuring accuracy of the timer in the program. The same small variance has been found with the simulation experiments, which gives evidence of a successful and representative simulation of the discrete behavior.

The simulation results reveal that the candidate system design does not accomplish the requirement for 30 ms intervals between two visualizations - only 86% of the intervals lie below the limit of 30 ms and not as required – at least 90%. With the real solution only 67% of the intervals were below the 30 ms barrier and the required limit from 90% could not be achieved as well. Thus a general agreement of the results from measurement and simulation can be found, with some more optimistic performance estimates from the simulation model.

The detailed analysis of the simulation results revealed, that the essence of the intervals form transportation delays by the internet and the operating time constitutes a fairly negligible contribution. This led to the conclusion that even the possible purchase of a more efficient hardware could not assure to the fulfilment of the given requirement. As long as the internet reflects the given behavior (exponential distribution, mean value 10 ms) the resulting repeat intervals are only rarely below the requested limit from 30 ms and the requirement does not seem to be realistic. Therefore the importance of the requirement to the solution would be reconceived and probably a lower percentage limit (<90%) or higher time limit (>30 ms) should be set.

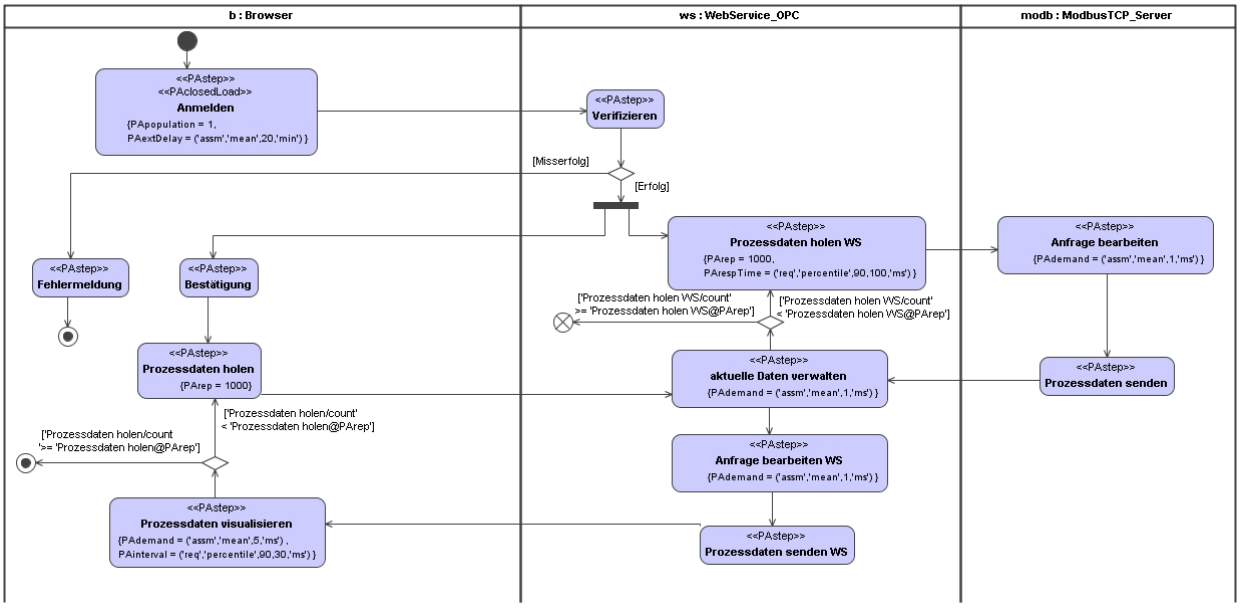

Figure 6. Behavior of the case study, modeled by an *activity diagram*

 Copyright © 2007 EUROSIM / SLOSIM

7 Conclusions

The presented approach supports an efficient and comprehensive performance analysis of system models in the early design phases. This offers potentials for substantially lowering of project costs. The underlying system performance model is based on an annotated UML model, which has to be augmented for a proper and complete performance specification. The augmented performance model can be transformed automatically into an executable Generalized Nets model. The modular transformation rules have been realized by means of XML style sheets and they can be easily adapted and modified to other annotation profiles.
An open challenge is the enormous variety of available UML elements. This variety as well as the nearly unrestricted possibility to interconnect the elements calls for a reduction to appropriately limited set of UML elements, relevant for automation solutions. In addition it should be mentioned that the SPT-profile is relatively young and not yet well-engineered and specification errors have to be removed. This will require further adaptation of the transformation rules. A third open item is due to the lack of standards for XMI, which results at present in proprietary solutions from different UML tool providers and no universal solutions are existing yet.

Acknowledgement

The authors are indebted to HERBERT-QUANDT-Foundation for financial support and to Tr. Triffonov and. K. Nikolov, Bulgarian Academy of Sciences, for the provision of and cooperation on the *GNTICKER* simulator.

8 References

[1] Oestereich, Bernd: *Analyse und Design mit der UML 2.1 - Objektorientierte Softwareentwicklung*, Oldenbourg Wissenschaftsverlag, 2006.

[2] SPT-Profile: http://www.omg.org/technology/documents/formal/schedulability.htm

[3] Daniel A. Menasce, Virgilio A.F. Almeida, and Lawrence W. Dowdy. *Performance by Design*. Prentice Hall, 2004.

[4] C.U. Smith. *Performance Engineering of Software Systems*. Addision-Wesley, 1990.

[5] Connie U. Smith. *Performance Solutions: A Practical Guide To Creating Responsive, Scalable Software*. Addison-Wesley, 2002.

[6] Simonetta Balsamo, Antinisca Di Marco, Paola Inverardi, and Marta Simeoni. *Model-based performance prediction in software development: A survey*. IEEE Transactions on Software Engineering, 30(5):295–310, May 2004.

[7] C.M. 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, 44(1):20–34, January 1995.

[8] J.A. Rolia and K.C. Sevcik. *The method of layers*. IEEE Transactions on Software Engineering, 21(8):682–688, 1995.

[9] G. Gu and D.C. Petriu. *Xslt transformations – from uml models to lqn performance models*. In ACM Proceedings of the International Workshop on Software an Performance, 2002.

[10] E. D. Lazowska, J. Zahorjan, G. Scott Graham, and K.C. Sevcik. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Prentice-Hall, 1984.

[11] K. Kant. *Introduction to Computer System Performance Evaluation*. McGraw-Hill, 1992.

[12] P. Kähkipuro. *Uml-based performance modeling framework for component-based distributed systems*. Lecture Notes in Computer Science 2047, 2001.

[13] P.G. Harrison and J. Hillston. *Exploiting quasi-reversible structures in markovian process algeba models*. Computer Journal, 38(7):510–520, 1995.

[14] H. Hermanns, U. Herzog, and J.P. Katoen. *Process algebra for performance evaluation*. Theoretical Computer Science, 274(1-2):43–87, 2002.

[15] M.K. Molloy. *Performance analysis using stochastic petri nets.* IEEE Transactions on Computers, 1982.

[16] M. Ajmone Marsan. *Stochastic petri nets: An elementary introduction*. In Advances in Petri Nets, volume 424 of Lecture Notes in Computer Science, pages 1–29. Springer Verlag, 1990.

 Copyright © 2007 EUROSIM / SLOSIM

[17] Moreno Marzolla. *Simulation-Based Performance Modeling of UML Software Architectures*. PhD thesis, Universit'a Ca Foscari di Venezia, 2004.

[18] A. Hennig, D. Revill, and M. Ponitsch. *From uml to performance measures – simulativeperformance predictions of it-systems using the jboss application server with omnet++*. In Proc. of ESM '03, the 17th European Simulation Multiconference, 2003.

[19] Cassandras, Chr.G., Lafortune, St.: *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[20] Dimitrov, E; Schmietendorf, A.; Atanassov, K.: *Netzbasierte Modelle für die Performance Analyse von multi-tier Client/Server Systemen*, In Proc. zum 2. Workshop Performance Engineering in der Softwareentwicklung (PE2001) S.21-32, Universität der Bundeswehr München, 26. April, 2001

[21] Schmietendorf, A.; Dimitrov, E.; Atanassov, K.: *The use of generalized nets within tasks of software performance engineering*, In Proc. of the Second International Workshop on Generalized Nets S. 1-12, Sofia, Bulgaria, June 2001

[22] Atanassov, Krassimir: *Generalized nets*; World Scientific Publ. Co., 1991, ISBN 981-02-0598-8.

[23] Koycheva, E.N., Trifonov, T.A., Aladjov, H.T.: *Modelling of UML sequence diagrams with generalized nets*; First International IEEE Symposium Intelligent Systems, Varna 2002, pages: 79-84, ISBN: 0-7803-7134-8

[24] UML Testing Profile: http://www.omg.org/technology/documents/formal/test_profile.htm

[25] Simon St. Laurent & Michael Fitzgerald: *XML Pocket Reference*; O'Reilly, Third Edition September 2005, ISBN 978-0-596-10050-6

[26] XMI: http://www.omg.org/technology/documents/formal/xmi.htm

[27] Frank Bongers: *XSLT 2.0 Grundlagen, Anwendung und Referenz*; Galileo Computing, 2004, ISBN 978-3-89842-361-8

[28] Centre of Biomedical Engineering, Bulgarian Academy of Sciences: http://www.clbme.bas.bg; http://generalised.net

[29] Frank Iwanitz and Jürgen Lange: *OPC - Fundamentals, Implementation, and Application* / Hüthig, 2005; ISBN-13: 978-3778529041

 Copyright © 2007 EUROSIM / SLOSIM