# From UML to Performance Measures - Simulative Performance Predictions of IT-Systems using the JBoss Application Server with OMNET++

Andreas Hennig, Dean Revill and Michael Pönitsch
Siemens AG, Corporate Technology, CT SE 1,
Otto-Hahn-Ring 6, 81739 München, Germany,
Andreas.Hennig@siemens.com, dean_ntu@hotmail.com, Michael.Poenitsch@siemens.com,

**KEYWORDS**
Software Performance Engineering, UML Modelling, Software Development, Performance Annotation, Simulation, JBoss, J2EE, LoadTest

**ABSTRACT**

In this paper, we argue the case for thorough performance engineering already in the early development phases of complex IT-systems, particularly web-based ones on the example of the Open Source Application Server JBoss. We show the need for a fast and efficient modelling of web-architectures, shortly recall a proposed UML notation and conversion framework [Hennig02], report progress of our end-to-end integration of simulation into a commercial UML-Tool and demonstrate its benefit on the example of a JBoss-based application. The abstractions chosen for the JBoss EJB Application Server model in the OMNET++ Simulator are described and the predicted performance is compared to values observed in load tests on the finished system.

**INTRODUCTION**

Most of today's complex IT-systems consist at least in parts of web-technologies – increasingly even in their core functionalities, not merely the user front end. Apart from the reduced number of core standards and mechanism (e.g. http, SOAP, WebServices, J2EE) as well as the (hopefully) improved interoperability, it is the wide availability of server software (commercial and public domain), engineering tools and development know-how that advances their use. Also the fast pace of developing the initial (typically highly presentable) increments is in favour of the web, but often conceals the long way to complex distributed systems with good performance.
For the single and local user that typically develops and tests the system, everything works smoothly and transparently, but the "going live" almost immediately exposes them to a large, wide and highly critical audience: the www-users and the competitor's web-sites "one click away". Unfortunately, typical failures in distributed systems do not arise during the test of individual components by individual users but occur during integration, system and deployment test with many concurrent users and transactions. Unless located and corrected in time they may result in cost overruns and missed deadlines. The prediction of performance properties of software systems (e.g. capacity, speed, stability, reliability, scalability), particularly distributed ones like web-applications, is therefore vital for substantiated design decisions at an early stage. Since these systems are often business critical and/or highly image critical, insufficient performance can incur heavy costs e.g. compensation, penalties, superfluous hardware, sub-optimal decisions based out-dated information, loss of market shares...

Simulation as a method of Software Performance Engineering (SPE, [Smith90]) has a long and successful track record of early and reliable performance predictions, which assess suitability of design choices and thus helps to ascertain time, cost and function targets. However, even if known, SPE methods often fail to keep up with the fast pace of development, mostly through high modelling efforts, but also through their high communication needs between the developer (or designer) and the modelling expert.
Based on the work of our group within Siemens Corporate Technology, which offers SPE consulting to the Siemens business units we are convinced that simulation makes valuable contributions, but needs simplified and more intuitive modelling in a "native language" of the developers and powerful and flexible pre-modelled abstractions of common infrastructure components and protocols to be sufficiently fast, trustworthy and effective. The currently most widespread "native" language of SW-developer would be the UML. One such component would be an J2EE application server, a central and common infrastructure element of current web-architectures.

In the following sections, we will recall the main elements of the UML-notation used; the one-click integration approach of the simulation into the UML-Tool before describing the application of the notation to an EJB application server and the JBoss model developed for OMNET++. We then compare the predications of a simple EJB system from the simulation to values measured in load-tests on real prototype systems before concluding the paper.


 *ISBN 3-936150-25-7*

## UML PERFORMANCE ANNOTATIONS

UML has established itself as the "native" modelling language of SW-development – despite all its shortcomings and ambiguities. It is therefore not surprising, that we among many others advocates of SPE (e.g. [Mirandola00], [Klein96], [Dimitrov00], [Xu03]) favour UML for SPE modelling as the most suitable way to integrate SPE methods and software engineering. However, we regard the SPE annotations as the "guests" in the "production model" and therefore formulated strict requirements to ensure acceptance of SPE annotations [Hennig01]: one single model, the same UML tool (commercial off-the-shelf, even if incomplete in support of UML standard), same interpretation of UML elements, same diagram types, no (or very few) additional diagrams, no interference with forward/reverse engineering, no SPE artefacts in generated code, no manual conversion steps…

This means in particular the use of sequence diagrams instead of state diagrams for representation of behaviour and resource consumption, since sequence diagrams are by experience not only available far earlier at less effort and require less detail; they are also more intuitive to developers, analysts and project owners. Using sequence diagrams directly without manual transformation ensures the consistency of team-model and SPE-model. The lack in precision (of modelling all possible behaviours) is more than compensated by the fact that it is modelled for SPE evaluation - even if only "typical" behaviours are modelled rather than "all possible".

Use-case diagrams are used to aggregate behaviour and allow for usage variation through parameterisation. Actors in Sequence and Use-Case diagrams represent the load onto the system. Deployment diagrams are used as the obvious choice (like in [Williams98], [Dimitrov02], [Mirandola00], [Petriu99], unlike e.g. [Arief00]) to model HW and SW entities as well as their available resources and connections. UML-multiplicities model repeated occurrences of identical entities like servers in a server farm or a pool of clients. Performance measures and requirements can be modelled in sequence diagram, where the observed values can be recorded or requirement violations can trigger alerts.

Of the many possible diagrams in a production model, the experiment setup defines which deployment, use-case and sequence diagrams should be evaluated, together with setting parameteris values to allow for variants. This way, one single production model can contain various experiments ("what happens in infrastructure scenario a, b, c; which user behaviour x, y, z") or of abstraction levels ("strata") in a consistent and non-interfering manner. Although creating special SPE-models for each experiment would separate individual inquiries more clearly, it would render consistency over various fast-paced iterations virtually impossible.

## SIMULATOR INTEGRATION INTO UML-TOOLS

In order to achieve a transparent end-to-end process for the user, we devised and implemented the following integration concept (Fig. 1). In the UML-CASE tool (currently TogetherJ 4.2 [togethersoft]), the simulation cycle is initiated ("one-click"), for which the required diagrams are obtained and compiled into a XML document describing the entire experiment. The experiment description is complemented by optional default settings and information on available library modules like JBoss. The "converter" generates a simulation definition file from the experiment, which is then compiled into the network and behaviour parts of the simulator. The simulator is based on the freely available discrete event simulator OMNeT++ [Varga01], [Varga97] and contains the relevant core modules and specific SPE extensions (scheduler, workflow execution engine…) as well as pre-modelled modules like the representation of JBoss (see next sections). The statistics of performance observations collected during the execution of the simulator are compiled and fed back textually into dedicated tagged values in the UML-model (if requested). The entire cycle is controlled by a set of platform-independent scripts, which could also be used to run an entire series of experiments.

Instead of producing merely NED for OMNetT++ as simulation definition file, it is also possible for the converter to be expanded to produce definition files for different evaluation techniques, e.g. Queuing networks, Bottleneck Analysis [Eckard01] or Performance Prototyping [Hennig03].

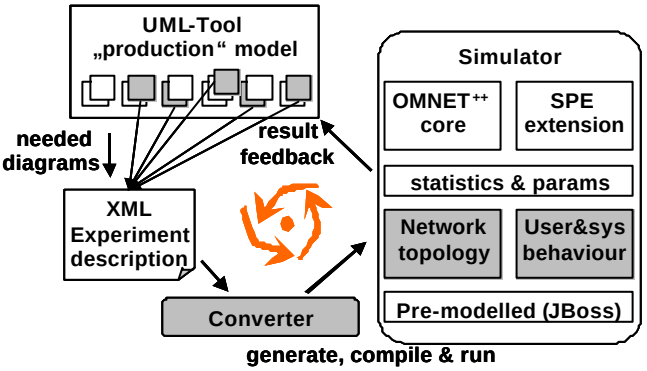


*Fig. 1: End-to-end simulation cycle from UML-tool*

In order to keep the simulation cycle short, we decided against using XMI [XMI02] directly instead of the XML experiment description, since the export and subsequent parsing and traversal of an entire production model with many unused diagrams would be too resource-intensive. In the future, we intend to use a programmatic interface (based on the Meta Object Facility MOF [MOF02]), which allows to query and update specifically identified elements of the model according to the MOF metamodel without having to go through its XMI representation.

## MODELLING THE SAMPLE JBOSS APPLICATION

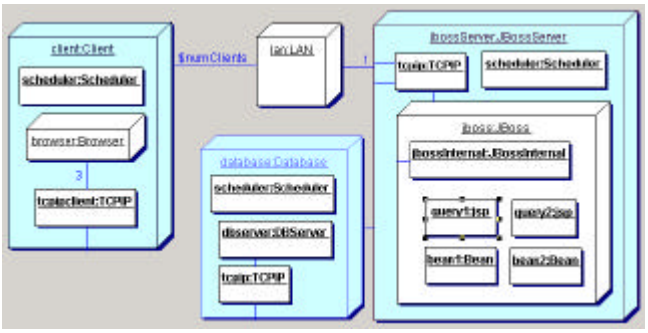


*Fig. 3: EJB network model in UML deployment diagram*

The infrastructure and network of the example application is modelled in the UML deployment diagram depicted in Fig. 3. It contains (on the right) the machine hosting the JBoss server with abstractions of a scheduler as provider of the resource "CPU", a TCP/IP communication protocol stack and the JBoss server itself. The business logic modules, i.e. the JSPs and beans, are placed inside the JBoss server. The specific type of beans is specified by UML attributes (tagged values) of the beans: beantype (session or entity), statefulness (stateful and stateless) and persistence (container- or bean-managed). To avoid cluttering of the diagram by the internals of the JBoss (e.g. the servlet engine and containers for the different types of beans with their numerous interceptors), the UML diagram contains a module "JBossInternal" representing the internals. The diagram also shows the physical connections between the hosts and network nodes, as well as multiplicities indicating e.g. how many web browsers (3) are running per client host ($numClient).

The intended behaviour of the application is modelled with the sequence diagram shown in Fig. 4, where the actor "webUser" represents the load of one or more concurrent users with prescribed load pattern (e.g. gradual ramp-up or continuous, interarrival delay, think-times). Upon a single "click" the browser submits three http requests to JSPs query1 and query2, which serve as a front end to the beans. On one occasion, bean1 consults the database before sending the response back to the browser. We also

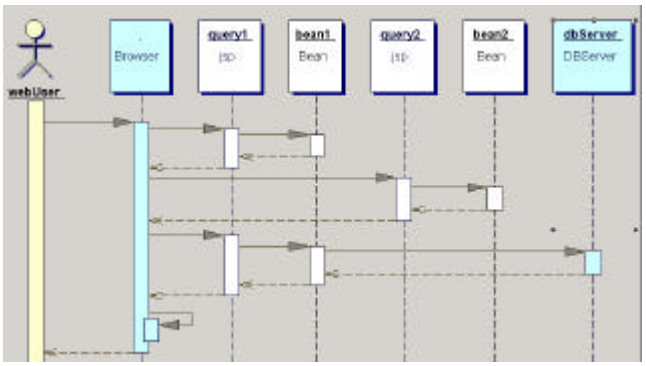


*Fig. 4: EJB behavioural model in UML sequence diagram*

specified how much CPU-time each step consumes. CPU-consumption is specified either as CPU-seconds or in iterations of the core operation of standard or application-specific benchmarks like whetstone or heapsort. This allows simple scalability investigations, by adjusting the amount of provided CPU-resources in the scheduler. Requirements are specified in the sequence diagram as well, e.g. the combined start-to-finish time the three browser requests should be below a given threshold.

After modelling infrastructure, load, behaviour and requirements in various UML diagrams, an experiment definition diagram is used to specify the subset of "investigated diagrams" and overall parameters like a scalability factors for number of clients (e.g. $numClients in. Fig. 2), the workload or overall think time.

## JBOSS SIMULATION MODULES IN OMNET++

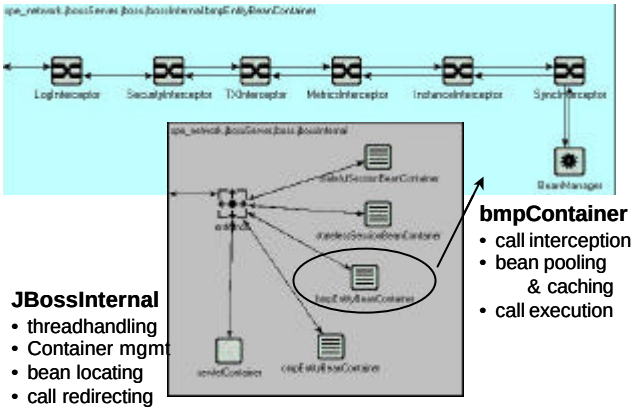


*Fig. 5: JBoss internal modules in OMNET++*

The option to use pre-modelled simulation modules allows us to develop complex infrastructure components separate from the UML-models of the target application. This helps avoiding unnecessary detail and clutter in the UML-diagrams, but also provides the means to build model libraries in the native programming environment of the simulation engine (C++ for OMNET++). It also enables us to use models from the growing OMNET++ user community to build larger scenarios more quickly. Typical examples for pre-modelled modules are communication protocols like TCP and infrastructure components like JBoss. Fig. 5 therefore shows the sub-modules of JBossInternal in terms of OMNET++. The "user" of the JBoss models would only use the more familiar UML representations in Fig 3 and 4.

The module "JBossInternal" is responsible for modelling the thread-handling of the JBoss and for the overall management of the bean containers. When a request to a bean arrives at JBossInternal, the target container is located based on the type of bean (session or entity, container- or bean-managed persistence) or servlet. The call is then redirected towards the appropriate container, in Fig. 5 this is a "bmpContainer" for entity beans with bean-managed persistence.

Inside the container, the call traverses a series of configurable interceptors (e.g. for logging, security,

transaction processing) before arriving at the beanManager. The beanManager obtains the requested bean instance either from a pool or cache of beans, where the call will be processed. At this time, the information of the sequence diagrams is consulted to obtain the prescribed behaviour like resource consumption, timestamp collection and requirement evaluation. If nested calls are required (like the database request from bean1 in Fig 4.) they are executed according to condition and iteration specification given in UML. One of the most challenging aspects of the JBoss model was the fact that the entire sequence of containers, interceptors and managers is executed within the same java thread – which could clearly be seen in traces generated from a simple manual prototype application. This meant that all simulation modules had to reuse a specific existing thread to prevent loss of simulation accuracy due to excessive context switches.

## EXPERIMENTS AND RESULTS

For experimental evaluation, we used the above simple behaviour and varied the load on the system through the number of simulated users. The load as increased every 2 minutes by one additional simulated user until approximately 10 minutes after a clear saturation of the system had been reached. In the sequence diagram, the call to bean2 was specified to be resource intensive (the equivalent of performing 3500 heapsorts on arrays of 1000 floating point values) e.g. for analysing user authorization), the calls to bean1 are less demanding (1500 heapsorts). The think time was 5 seconds on average, which represents the time a user would need before the first click in the browser.

In order to compare the simulation results with real installations of JBoss, we built a prototype of above specifications and deployed it onto two different hosts. Both hosts run under Linux, dax is a dual-processor server, ibex a single-processor workstation. For reasons of simplicity, scaling of the simulation model to the reference hosts was done through adjustment of the CPU-capacity of the server in UML diagram only.
For the time being, this ignores other influences like network bandwidth, i/o speed and latencies, which could distort findings significantly.

*Unfortunately, we experience a software incompatibility, which forces us to upgrade underlying parts of the system. Rather than presenting misleading low-quality data, we give a overview on the type of data and investigation we will present. For the final paper / camera-ready copy to be submitted for the ESM, we will obtain additional measurements and run further experiments to calibrate the model more precisely and then verify the accuracy of the prediction on a larger sequence of interactions and further examples. We apologize the inconvenience.*

Taking the prototype measures on dax as an example, the system went into saturation (ref. markers in Fig. 6) after 60% of the experiment time with an approximate capacity of 1.9 TPS, which was caused by 23 concurrent simulated users. Further users did not increase the transaction rate but only resulted in increased response times due to shared use of the CPU resources. The response time for the first user (on a basically idle system) was 0.2s without the prescribed think time, but rose to 1.1s at the saturation point of the system (ref Fig. 7). Table 1 summarily lists the achieved or predicted rate of fully completed transactions per second (TPS, number of finished workflow instances per second).

| | | Capacity | | Response time | |
|---|---|---|---|---|---|
| | | TPS | Users | "idle" | saturated |
| simulation | ibex | | | | |
| | dax | | | | |
| prototype | Ibex | 0.6 | 10 | 1.0s | 4.6s |
| | dax | 1.9 | 23 | 0.2s | 1.1s |

***Table 1: Comparison of simulation and prototype results***

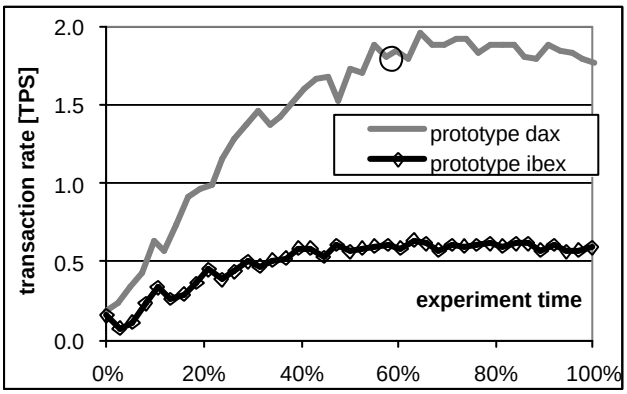


***Fig. 6: Transaction rates of simulation and prototype***

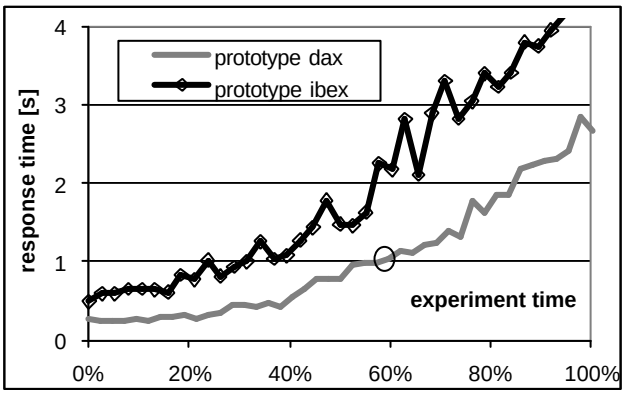


***Fig. 7: Response times of simulation and prototype***

## OUTLOOK AND CONCLUSION

The simulation model presented in this paper provides a performance engineer with a simple and intuitive way of noting infrastructure, load, behaviour and requirements of a distributed system, which uses a JBoss application server. By splitting the model into an application (UML) and infrastructure level (native OMNET++), we achieve sufficient expressive and simulative potential without requiring excessive detail.

We currently work on integrating our method with models from the OMNET++ community, most importantly the TCP/IP models described in [Kaage2001]. Another aspect is the calibration of the model by efficient determination of model parameters on various infrastructures by means of benchmarking, prototyping (manual and automatic) and load testing. Particularly the possible interaction between Simulation and Prototyping is very promising.

In [Hennig03], we present a way to use the notation, conversion methodology and UML-models described here to automatically generate and deploy performance prototypes. The efficient combination of corresponding prototypes and simulation models opens new opportunities to predict system performance faster and more accurately with affordable effort and time.

## REFERENCES


[Arief00] Arief, L.B., Speirs, N.A. "*A UML Tool for an Automatic Generation of Simulations Programs*", WOSP 2000, p 71-76, 2000

[Dimitrov02] Dimitrov E., Schmietendorf A., Dumke, R.: "*UML-based Performance Engineering Possibilities and Techniques"*, IEEE Software, Darmstadt, p. 74-83, Vol 19/1, Jan/Feb 2002

[Dimitrov00] Dimitrov E., Schmietendorf A.: "*UML-basiertes Performance Engineering*", "in Performance Engineering in der Softwareentwicklung (PE 2000), ", Darmstadt, p. 41, 2000

[Hennig01] Hennig, A.; Eckardt, H., 2001, *"Challenges for Simulation of Systems in Software Performance Engineering*", in Proc. ESM'01, Prague, p. 121-126, 2001

[Hennig02] Hennig, A. R. Wasgint; "Performance Modeling of Software Systems in UML-Tools for the Software Developer", in *Proceedings of European Simulation Multiconference ESM'2002*, Darmstadt, Germany, 2002

[Hennig03] Hennig, A., Hentschel, A. and Tyack, J., "Performance Prototyping - Generating and Simulating a distributed IT-System from UML models" submitted to *European Simulation Multiconference ESM'2003*, Nottingham, UK, 2003

[Kaage01] Kaage, U., Kahmann, V., Jondral, F., *„An OMNet++ TCP/IP MODEL*", in Proc. ESM'01, Prague, p. 409-413, 2001

[Klein96] Klein, M.H.,: *"State of Practice report: Problems in the Practice of Performance Engineering*". Technical Report, Pittsburg, Pennsylvania: Software Engineering Institute, 1996

[Mirandola00] Mirandola, R., Cortellessa, V.; 2000, *"UML Based Performance Modeling of Distributed Systems"*, UML 2000 - 3rd Int. Conf., York, UK, 178-193, 2000

[XMI02] OMG, XMI 1.2, 2002 , http://www.omg.org/technology/documents/formal/xmi.htm

[MOF02] OMG, MOF 1.5 RTF, 2001/2002, in Revision, http://www.omg.org/techprocess/meetings/schedule/MOF_1.5_RTF.html

[Petriu99] Petriu, D., Wang, X. *„From UML descriptions of High-Level Software Architectures to LQN Performance Models*", Proc AGVTIVE'99, Springer Verlag, LNCS 1779, p47-62, 1999

[Smith90] Smith, C.U., 1990. *"Performance Engineering of Software Systems"*, ISBN 0-201-53769-9, Addison-Wesley, Reading, US, 1990

[Togethersoft] Togethersoft Corporation, http://www.togethersoft.com/

[Varga01] Varga A., "*The OMNET++ Discrete Event Simulation System*", in Proc. ESM'01, Prague, p. 319-324, 2001

[Varga97] Varga, A. OMNET++ Homepage, http://www.hit.bme.hu/phd/vargaa/omnetpp.htm, 1997

[Williams98] Williams, L.G., Smith C.U., "*Performance Evaluation of Software Architectures*", WOSP 1998, p 164.177, 1998

[Xu03] Xu, Z, Lehmann, A., "Automated Generation of Queuing Network Model from UML-based Software Models with Performance Annotations", Techgnical Report #2002-06, Universität der Bundeswehr, München, 2002