

The Role of Simulation Performance in Software-in-the-Loop Simulations

Tommy Baumann, Bernd Pfitzinger, Thomas Jestädt
and Dragan Macos

Abstract The simulation performance is one aspect that becomes important in the practical application of simulations. A number of situations necessitates high-performance simulations: The simulation of large systems, over long time-periods or exploring a large solution space. Starting with these scenarios we discuss the parallelization of a discrete event simulation (DES) model using a synthetic benchmark model. Using the example of the German automatic toll system we explore the performance constraints originating from coupling an abstract simulation model with a real-world system.

1 Introduction

Simulations—regardless of the chosen technology—are an integral part of the system development process. The ever increasing complexity of software-intensive systems puts a particular strain on the technical support for automating the system development [1]—requirements could be undocumented, written in natural language, missing the user or operational perspective [2, 3] and of course incorrectly implemented.

T. Baumann (✉)

Andato GmbH & Co. KG, Ehrenbergstraße 11, 98693 Ilmenau, Germany
e-mail: tommy.baumann@andato.com

B. Pfitzinger · T. Jestädt

Toll Collect GmbH, Linkstraße 4, 10785 Berlin, Germany
e-mail: bernd.pfitzinger@toll-collect.de

T. Jestädt

e-mail: thomas.jestaedt@toll-collect.de

D. Macos

Beuth Hochschule Für Technik Berlin, Luxemburger Str. 10, 13353 Berlin, Germany
e-mail: dmacos@beuth-hochschule.de

© Springer International Publishing AG 2017

T. Pelech-Pilichowski et al. (eds.), *Advances in Business ICT: New Ideas from Ongoing Research*, Studies in Computational Intelligence 658,
DOI 10.1007/978-3-319-47208-9_2

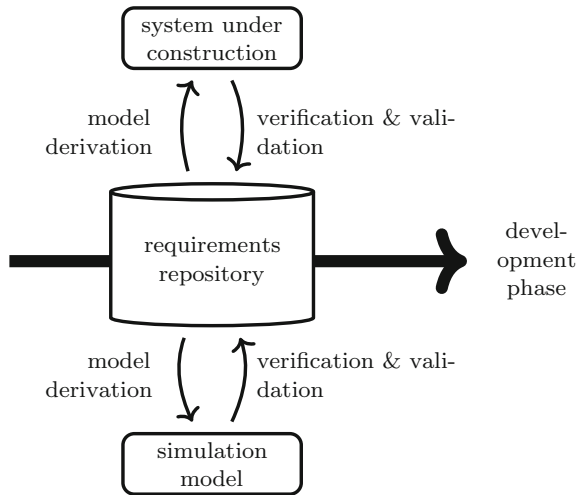
Accordingly many efforts exist to improve the development process—test driven development [4] starts with the implementation of the (unit) test cases that need to be passed by the system, agile process models [5] reduce the development risk by shortening the development period.

However, both approaches fall short of putting the development effort into the operational context: The success of a system is only determined once it is used under normal operating conditions. I.e. the usage, operational capacities and practices as well as the interplay with neighboring systems determines whether a system fulfills its requirements. Merging development and operations (“DevOps”, [6]) is one proposed solution—albeit increasing the operational risks to a level unacceptable in liability-critical systems or when a system needs to be non-discriminatory. In contrast, ‘micro-services’ split a complex system into simple, seemingly independent services and claim to eliminate the complexity of large systems [7].

The use of simulation models is a different approach to overcome these problems: A simulation model is put at the beginning of the system development—capturing the operational context, the current knowledge regarding the requirements and encompasses the whole system (including user behavior, operational procedures and partner systems). As in “test-driven”, the simulation model defines the test case and acceptance criteria—at the level of the complete, integrated system. In that sense, requirements become *executable* through models before the system development starts. During the system development effort, requirements are first included in the simulation model before the implementation starts (see Fig. 1).

Even when the simulation model remains at a certain level of abstraction (i.e. not connected to the real-world system) the simulation performance becomes a bottleneck: To realistically reproduce the operational context simulation parameters need to

Fig. 1 In the simulation driven development approach (SDD) the requirements become *executable* as a simulation model



be fitted to observations (e.g. see [8] for the German automatic toll system). The simulation performance is even more important when the simulation model is intended to interface with real-world (sub-) systems running in ‘real time’—simulation models need to be parallelized (see [9] where we introduced a simple benchmark model).

The ability to simulate at different levels of abstraction is of particular interest when simulations accompany or drive the system development process. To that extent the next section introduces the coupling between an abstract simulation model and a real-world system—either as a hardware or software-intensive system. We describe a particular challenge: The time synchronization across the whole system when the level of abstraction changes during a simulation run, e.g. a real-world system replaces a simulated sub-system starting at a given point in time. Retaining a valid concept of time spanning the simulated and the real-world system is covered in more detail in Sect. 3. In particular the section summarizes the experiences gathered from a synthetic benchmark model mimicking the interplay of a simulation model exchanging data with a web-server using thousands of parallel TCP/IP connections simultaneously. Section 4 takes the example of the German automatic toll system as a potential use case. We estimate the performance constraints for the simulation by observing the dynamics of this real-world system.

2 Hardware and Software in the Loop Simulation

Characteristic to a simulation driven design approach is the integrated system level development scope accompanying all design stages. Over time the simulation model of the system under development is refined continuously as the development progresses from the requirement phase up to the implementation phase. The requirements repository (see Fig. 1) collects the current knowledge regarding the system under construction which is in turn first implemented as an executable simulation model—putting any requirement immediately into the context of the integrated system under normal operating conditions. Operational scenarios are either set up in the requirements phase or exist from the operational experience gathered in an existing system.

With the progress of the development process already existing or newly implemented sub-systems may be tested against the holistic system model (including all sub-systems) for validation, optimization and to locate system-level integration issues. To that extent an abstract simulation model may be used to produce dynamic performance indicators for sub-systems—to be tested independently. More stringent is the use of the simulation model to drive real-world sub-systems: This can be done by Hardware-in-the-Loop (HiL) and Software-in-the-Loop (SiL) simulations. Here the simulations allow to perform functional tests (compliance with protocols) as well as load tests by coupling real-world sub-systems with the virtual sub-systems already existing in the simulation model.

To couple real-world sub-system components with virtual sub-system components two major issues need to be addressed: The abstraction gap needs to be overcome and the time synchronization assured. The abstraction gap results from the typically lower level of detail used in simulation models compared to real-world systems. Models are always built to answer certain questions and therefore they implement only properties necessary to answer these questions. Most often, a sizable amount of structural and behavioral properties can be ignored in that context. However, to couple a real-world system component using HiL/SiL exactly these properties are necessary, e.g. to concisely follow the specification of a given communication protocol.

The second challenge is the synchronization between ‘real time’ and simulation time. Usually the simulation time should progress significantly faster than reality. This is a desired effect, since the simulation results need to be available as fast as possible—decision support is infeasible when the simulation results do not arrive in time. Nevertheless, it is necessary to synchronize the simulation speed down to real time as soon as data is exchanged with a real-world component for the first time. Up to that point in time the simulation may run at a faster speed to reach a certain system state. Then both time frames need to proceed in step. In addition, it may

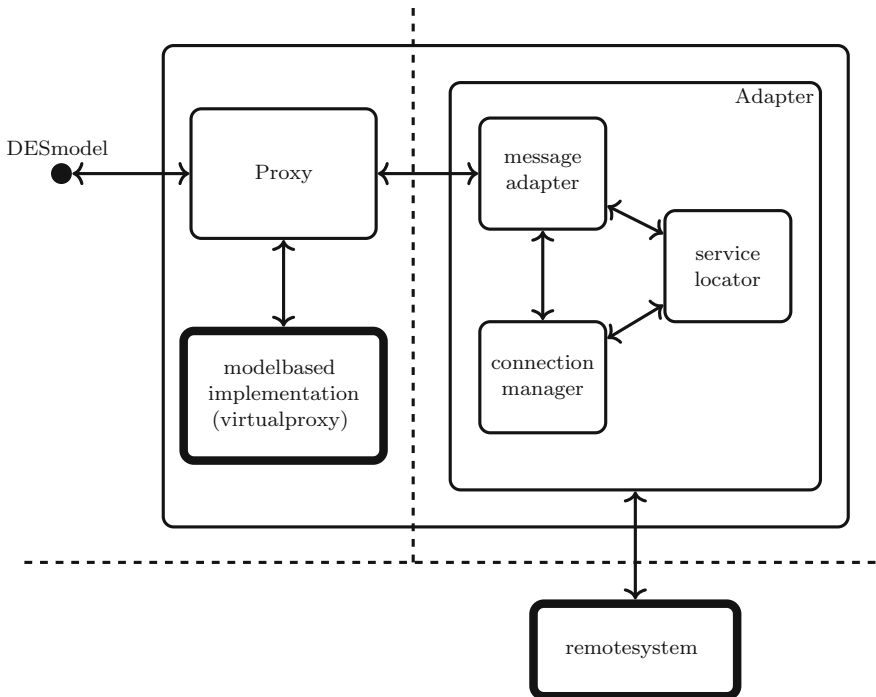


Fig. 2 A proxy pattern [10, 11] is used to implement a sub-system within the model either as a DES model (*left of the dashed line, bold*) or to adapt the discrete event messages to the technical interface used in a real-world remote system (*below the dashed line, bold*)

become necessary to take over the states from the virtual representation of the real system component to the real component (for example open sessions of a server). One technical prerequisite is the possibility to adjust the simulation speed during an ongoing simulation—a (not yet common) feature of the simulation environment.

The typical architecture of a HiL/SiL simulation solution is shown in Fig. 2. The model-based representation of a certain sub-system is permanently or temporarily replaced by a real-world system. A proxy component within the simulation model conveys the requests between the model-based system and the real-world sub-system. In addition, an adapter is necessary to translate the discrete event messages to the technical interface used in the real system. The adapter receives discrete event messages from the simulation model and creates actual network connections (e.g. TCP/IP based). In most cases additional information related to the connection is necessary, since the messages of the simulation model are depicted at a higher abstraction level. In our example, all network connections are managed by the same adapter in parallel and the next section explores the scalability of this approach. In effect this means that a serial-mode simulation environment may become too slow to process requests in a timely fashion, when a large amount of parallel connections are active—even when the simulation speed is normally faster than reality.

3 Performance Issues

To investigate the potential performance issues when coupling a serial simulation model to a parallel real world adapter we created a synthetic benchmark model in MSArchitect [12]. The model mimics the interplay of a simulation model exchanging data with a web-server using thousands of parallel TCP/IP connections simultaneously. The synthetic benchmark model implements a test driver component to open a configurable amount of parallel connections to a real server. The main task of the adapter is to redirect the data flow from the model to the real server and vice versa, where 1 kB of data is transferred in each communication session with the ability to throttle the bandwidth of each connection e.g. to the bandwidth typical in mobile data networks.

For the analysis we created several setups of synchronous and asynchronous multi-threaded TCP-servers (real-world mock-up server) and TCP-clients (as part of the proxy within the simulation framework MSArchitect) using Winsock [13] and C++ ASIO [14]. The focus was to test if over 10,000 connections can be opened and processed in parallel.

The best results could be achieved by the C++ ASIO based loop-back scenario. Here the client and the server are running on the same computer, more than 10,000 parallel connections (and therefore threads) could be achieved. The whole simulation run was executed within a few seconds using a typical multi-core server. In a second setup the TCP-server sub-system was moved to its own dedicated computer accessible via a LAN connection. This reduced the number of parallel connections to approximately 5,500—adding more connections would lead to connections being

closed spontaneously. The reasons are unknown so far, whether it is problem of the network, the operating system or deficiency in the implementation of client and server. In general it could be observed that the time necessary to establish a connection was significantly higher than the time necessary to transfer the data (payload).

4 Example of German Toll System

The performance constraints mentioned above depend on the use case. In our example we take the German automatic toll system as use case—a typical example of a toll system based on global navigational satellite systems [15]. The system uses OBUs deployed in the heavy-goods vehicles (HGVs) to determine the tolls. At present almost one million OBUs are deployed, communicating with the centrals system to transmit toll data or to download updates to the geo and tariff data and the OBU software. In the past we have developed a realistic DES model for the most important processes of the toll system (see [9] and references therein).

For many purposes the simulation model is sufficient even if it retains a certain level of abstraction. In our case the system size is at a scale of 1:1, communication bandwidth and latency are included in the simulation model. However, communication processes are not modeled in detail—neither the processing steps nor the exchange of TCP/IP packets is included in the model. Even with these limitations the simulation model can be applied to predict the dynamic system behavior—either of the existing system under normal operating conditions or beyond the design specification. New processes and systems can be included in the model and put into the 1:1 operational environment to predict e.g. the system sizing or the resource consumption.

For software-in-the-loop simulations the existing model would need to be expanded first to include all process steps and then interface with the relevant communication protocols—most notably communicating via TCP/IP. The interface to TCP/IP based connections is the natural boundary where a real-world system—e.g. the central server responsible for communicating with the OBU fleet—could be interfaced with the simulated behavior of the OBU fleet. Section 2 explained the technical pre-requisite: The ability to translate between an event-driven simulation model and thousands of parallel TCP/IP connections to an existing communication server.

To illustrate the necessary changes to the simulation model we look at the most common communication process used in the toll system (see Fig. 3): The exchange of a status message when no further activities are required. So far the simulation model contains realistic triggers for the process but summarizes the process by including only the beginning and the end of the process (steps A and E in Fig. 3) with the typical time taken for and data transmitted during a connection (a histogram of the overall connection duration is given in Fig. 4).

SiL simulations would need to implement the complete process and to interface with the TCP/IP based communication. Figure 3 indicates the minimum complexity required—even a simple exchange of a status message involves many steps: In the

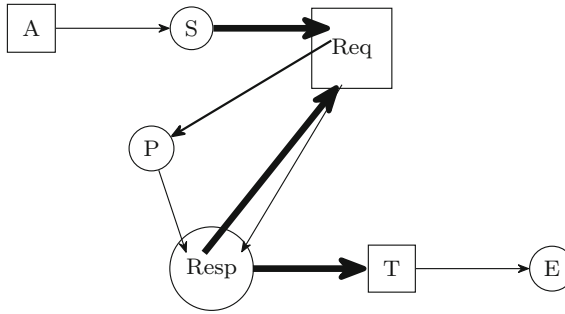


Fig. 3 Process flow for the most common communication process in the toll system example—the status message exchange. Rectangles (*circles*) denote state changes due to client (server) activities, the bold arrows indicate activities with longer duration. The process starts with a TCP/IP connection (A) and the spawning of a server-side communication thread (S), proceeding through the activities request (Req), processing (P) and response (Resp) before the connection is terminated (T) and the thread ends (E)

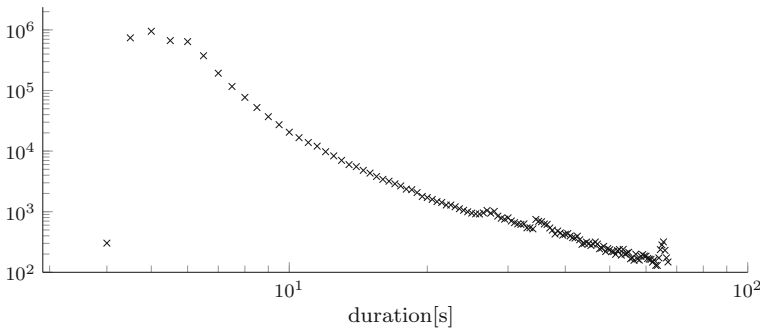


Fig. 4 Histogram of the duration of the status message exchange process (time elapsed between steps A and E in Fig. 3)

example the OBU initiates the communication by establishing a TCP/IP-connection to the central server (step A) which in turn forks a new thread dedicated to the newly established communication channel (step S).

The actual process consists of a simple loop: A request is received (Req) by the server, processed (P) and a response sent to the OBU (Resp). Even with a negligible message size the latency inherent in mobile data networks leads to delays of several seconds (indicated by the bold arrows), whereas server-side processing is typically very fast (on the order of milli-seconds up to a few hundred ms for steps including storage or encryption). To reach this level of detail in the simulation model, access to the technical processes and protocol specifications is needed. Yet—depending on the use case—it might still not be necessary to implement all details within the SiL simulation model, e.g. when the central billing systems are not covered by the model, many details concerning the correct billing data can be omitted from the simulation model.

In practice, access to real-world statistical data is also needed: In fact, in our example the process steps in Fig. 3 are reconstructed from ‘process mining’ (using the term of [16]) available data for the most common process—a step necessary to build statistical tables of e.g. the time needed to finish one process step. Figure 4 gives an impression of the typical findings when a large fleet of OBUs communicates with the central system: On average a connection is established for a duration of about 6 seconds—stemming mostly from the three message exchanges indicated by bold arrows in Fig. 3, each taking the typical round-trip time observed in GPRS networks [17]. Yet it is obvious that the probability distribution is not a Gaussian distribution: There is a sizable percentage of connections that takes considerably longer than presumed (for transmitting a few hundred bytes over mobile data networks). The log-log plot in Fig. 4 suggests a power law extending to durations of about 60s where a pronounced drop-off occurs (not visible in the figure).

This behavior is typical for TCP/IP connections over high-latency networks [18]: With time, some connections recover from temporary network outages unless a connection time-out (in our case at 60s) closes the connection first.

Regarding a SiL simulation model the empirical distributions need to be incorporated for several reasons: First and foremost the high percentage of connections with considerably longer duration translates into more open connections to be handled by the central server. In reality this behavior is the “real” system requirement—that can only be formulated with prior knowledge and tested by dedicated simulation models as test driver. Furthermore the empirical distribution indicates the scale at which the simulation model needs to respond in “real-time”. An example is the message transfer taking at least one second and on average two seconds—the delays introduced by the scheduling of the discrete events must be negligible on this scale. Of course, the system could contain many more processing steps with more stringent timing constraints. Especially server-side computing, e.g. access to persistent storage or delays due to dedicated cryptographic routines could be simulated but will operate possibly orders of magnitude faster.

In reality the proxy pattern used to translate between the discrete event simulation and the TCP/IP connections (see Sect. 2) serializes the communication over thousands of parallel connections into a single future events list to be processed sequentially by the DES kernel. Therefore the “real time” requirement is equivalent to processing all simultaneous events in the future events list within the allotted time frame. At least for this model element we expect that a parallelization is necessary.

5 Summary

Executable requirements are the underlying idea of the simulation driven design approach: In an extension to the test driven development approach, SDD puts every development effort into the context of the whole system running under normal operating conditions. Of course, the approach incurs additional expenses—a simulation model needs to be developed and maintained—yet it aims to shift many tests to the

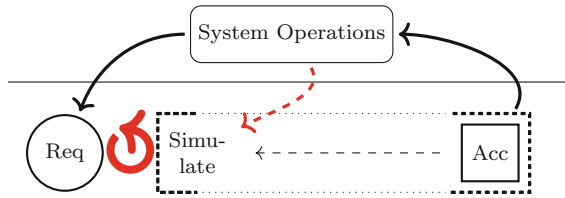


Fig. 5 Simulation driven development shifts the acceptance testing (acc)—at least partially—to the requirements definition phase (req). At all times the simulations put the system under construction into the operational context

very beginning of the software development process. In many respects the operational performance of the simulation model and the tool set determines the applicability of SDD: Requirements from a repository need to be connected to simulation models and model elements, e.g. to keep track of simulation results and for tracing the coverage of the simulation model. Furthermore simulations of large-scale systems at a scale of 1:1 require a high simulation performance (Fig. 5).

Taking a common use case—the simultaneous communication of many slow client computers with a central server—we have shown that a discrete event simulation model is capable to drive a real-world server with many thousand connections in parallel. The remaining challenge was illustrated using the German automatic toll system: In addition to serving a sufficient number of parallel connections, a realistic simulation must not introduce noticeable delays. As a consequence it becomes necessary to parallelize the DES model to keep the ‘artificial’ delays originating from the model execution at a negligible level.

References

1. K. Petersen, M. Khurum, and L. Angelis, “Reasons for bottlenecks in very large-scale system of systems development,” *Information and Software Technology*, vol. 56, no. 10, pp. 1403–1420, Oct. 2014. doi: [10.1016/j.infsof.2014.05.004](https://doi.org/10.1016/j.infsof.2014.05.004)
2. E. Woods, “Operational: The forgotten architectural view,” *IEEE Software*, vol. 33, no. 3, pp. 20–23, May 2016. doi: [10.1109/MS.2016.86](https://doi.org/10.1109/MS.2016.86)
3. B. Pfitzinger and T. Jestädt, *IT-Betrieb: Management und Betrieb der IT in Unternehmen*. Springer, 2016. ISBN 978-3-642-45193-5. doi: [10.1007/978-3-642-45193-5](https://doi.org/10.1007/978-3-642-45193-5)
4. K. Beck, *Test-driven development: by example*. Addison-Wesley Professional, 2003.
5. “Manifesto for agile software development,” [accessed 29-Jul-2015]. [Online]. Available: <http://www.agilemanifesto.org/>
6. J. Roche, “Adopting DevOps practices in quality assurance,” *Communications of the ACM*, vol. 56, no. 11, pp. 38–43, 11 2013. doi: [10.1145/2524713.2524721](https://doi.org/10.1145/2524713.2524721)
7. S. Newman, *Building Microservices*. O’Reilly Media, Inc., 2015. ISBN 978-1-4919-5035-7
8. B. Pfitzinger, T. Baumann, D. Macos, and T. Jestädt, “Using parameter optimization to calibrate a model of user interaction,” in *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems*, ser. Annals of Computer Science and Information Systems, M. P. M. Ganzha, L. Maciaszek, Ed., vol. 2. IEEE, 9 2014. doi: [10.15439/2014F123](https://doi.org/10.15439/2014F123). ISBN 978-83-60810-58-3. ISSN 2300-5963 pp. 1111–1116.

9. T. Baumann, B. Pfitzinger, D. Macos, and T. Jestädt, “Enhanced simulation performance through parallelization using a synthetic and a real-world simulation model,” in *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems*, ser. Annals of Computer Science and Information Systems, M. Ganzha, L. Maciaszek, and M. Paprzycki, Eds., vol. 5. IEEE, 9 2015. doi: [10.15439/2015F226](https://doi.org/10.15439/2015F226) pp. 1335–1341.
10. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., 1995. ISBN 0-201-63361-2
11. D. Alur, D. Malks, J. Crupi, G. Booch, and M. Fowler, *Core J2EE Patterns (Core Design Series): Best Practices and Design Strategies*, 2nd ed. Sun Microsystems, Inc., 2003. ISBN 0131422464
12. Andato GmbH & Co. KG, “MSArchitect,” [accessed 10-Dec-2012]. [Online]. Available: <http://www.andato.com/>
13. B. Quinn, *Windows Sockets Network Programming*, 2nd ed. Addison-Wesley Longman Publishing, 1998. ISBN 0201183943
14. C. Kohlhoff, “Boost.Asio,” 2015, [accessed 02-May-16]. [Online]. Available: http://www.boost.org/doc/libs/1_60_0/doc/html/boost_asio.html
15. A. T. W. Pickford and P. T. Blythe, *Road user charging and electronic toll collection*. Artech House, 2006. ISBN 978-1-58053-858-9
16. W. Aalst, *Process Mining: Data Science in Action*. Springer, 2016. ISBN 978-3-662-49851-4. doi: [10.1007/978-3-662-49851-4](https://doi.org/10.1007/978-3-662-49851-4)
17. G. Xylomenos, G. C. Polyzos, P. Mahonen, and M. Saaranen, “TCP performance issues over wireless links,” *IEEE Communications Magazine*, vol. 39, no. 4, pp. 52–58, Apr 2001. doi: [10.1109/35.917504](https://doi.org/10.1109/35.917504)
18. M. C. Chan and R. Ramjee, “TCP/IP performance over 3G wireless links with rate and delay variation,” *Wireless Networks*, vol. 11, no. 1-2, pp. 81–97, Jan. 2005. doi: [10.1007/s11276-004-4748-7](https://doi.org/10.1007/s11276-004-4748-7)

Advances in Business ICT: New Ideas from Ongoing
Research

Pelech-Pilichowski, T.; Mach-Król, M.; Olszak, C.M. (Eds.)

2017, VII, 135 p. 63 illus., Hardcover

ISBN: 978-3-319-47207-2