Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A.M. Uhrmacher, eds

A NEW WEB BASED METHOD FOR DISTRIBUTION OF SIMULATION EXPERIMENTS BASED ON THE CMSD STANDARD

Sören Bergmann
Sören Stelzer
Steffen Straßburger

Ilmenau University of Technology
Helmholtzplatz 3
98684 Ilmenau, GERMANY

ABSTRACT

This article introduces a novel methodology for web based distribution of simulation experiments. The approach is related to themes such as web based applications, cloud computing or applications as a service, which have been recurring topics in scientific papers for years. The methodology is based on automatic model generation, initialization, and result analysis under usage of the CMSD standard. All user interactions are performed in web based user interfaces. Of special importance is that different simulations tools can be used in parallel without any additional effort. Furthermore the simulation tool actually used is transparent to the user. The applicability of our methodology is demonstrated for different production scenarios.

1 INTRODUCTION

Simulation, especially discrete-event simulation, is used in many different disciplines and application areas. In the area of production and logistics, simulation is a well-accepted tool for the planning, evaluation and monitoring of relevant processes. Fowler and Rose (2004) have discussed future challenges for modeling and simulation of complex production systems. Among others, they identified the reduction of the time and effort for simulation studies and the integration of simulation techniques with general manufacturing applications as future research areas.

A well-known approach to reduce the time of simulation experiments is the parallelization of simulation experiments, using multiple instances of one or more simulators which are executed on a sufficient number of computers. This approach is in theory easier to implement than parallel simulation, as the individual simulation runs can be executed independently. In practice, the implementation of an adequate experimentation environment can lead to highly complex solutions, often involving extra work. Also, the implemented solutions are often bound to a special simulation tool. This may lead to new problems concerning the number of available licenses for concurrent simulation runs.

Cloud computing and software as a service are among the most discussed themes in the last years, e.g., Gartner (2011; 2012) forecasts cloud computing as one of the top 3 technologies for the next years. In this context, and taking into account the challenges stated above, web based simulation (or implementing “simulation as a service”) seems to be very appealing. The promise of such approach is that, if successful, it can reduce the amount of time needed to execute simulation experiments. Also completely new business models, like effort or time-based billing, are possible through web-based simulation. Such business models can aim at an economically better use of the needed hardware and software for the simulation, especially concerning the sometimes very pricy cost for simulation licenses. Precise accounting for the time a certain license has been used offers the possibility for renting and paying for software licenses on an actual use basis.

The combination of GRID computing and simulation has been investigated by a number of authors. Taylor et al. (2011) present a solution for the distribution of simulation experiments on a desktop GRID. This solution is based on distributing Flexsim models on so-called worker clients, which execute a model in the background and deliver results back to a dedicated manager. Characteristic for this solution is the requirement to have Flexsim licenses available on each of the dedicated client machines. The distribution of the experiments is even bound to the requirement of using the same release version of the simulation system.

The web based simulation approach presented in our paper can use an unlimited number of *different* simulation tools for simulation runs. So is it possible to use one or more instances of a component based simulation tool, like Plant Simulation (Siemens 2012), and one or more instances of a language based simulation tool, like SLX (Henriksen 1999), at the same time. For the end user the web based simulator is transparent, because theoretically the type and count of the used simulator instances is unknown and also unimportant for the end user. The end user only defines the simulation model in a simulation system independent manner and the parameters for the simulation run. Using this approach, the actual simulation model can be automatically generated in the available simulation tools. The execution environment is responsible for distributing and executing the models in the available clients and for collecting and aggregating results of the experiments. This is possible without permanent allocation of hardware resources and software licenses.

In the first step of the work presented here we use the outlined approach only to distribute independent replications of one simulation experiment. In future versions, we also aim at defining entire sets of experiments, e.g., to test the effect of different buffer sizes, strategies or production plans.

The remainder of the paper is organized as follows. In Section 2, we discuss the requirements for web based simulation and also the distribution of simulation experiments through a web application. In Section 3 we show how web based simulation can be implemented and we introduce a prototype. Finally, Section 4 contains some concluding remarks and future research directions.

2 REQUIREMENTS FOR WEB BASED SIMULATION AND DISTRIBUTION OF SIMULATION EXPERIMENTS

The requirement for web based simulation and distribution of simulation experiments can be classified into three sub preconditions. The first precondition is the automatic model generation, the goal is the creation of the simulation model in a simulator on the server side. The second precondition is the initialization of this model with system load data and data about the current resource states. The third and last precondition refers to how the results of an experiment will be presented to the user on the client side.

Before we show our solution for the three preconditions, we introduce the CMSD standard which is used as the data model for exchange and storage of all information in the complete work flow.

2.1 The Core Manufacturing Simulation Data (CMSD) Information Model as base for the standardized data exchange

The core manufacturing simulation data (CMSD) information model is an open standard developed within the simulation interoperability standards organization (SISO). It has been developed in cooperation with academic and industrial partners under the leadership of the National Institute of Standards and Technology (NIST). The primary objective of the CMSD Information Model is to facilitate interoperability between simulation systems and other manufacturing applications (SISO 2010).

Two different methods are used for representing the CMSD standard: the Unified Modeling Language (UML), and an XML schema definition language, in this case RELAX NG and Schematron (SISO 2011). The UML representation has been organized using packages shown in Figure 1.

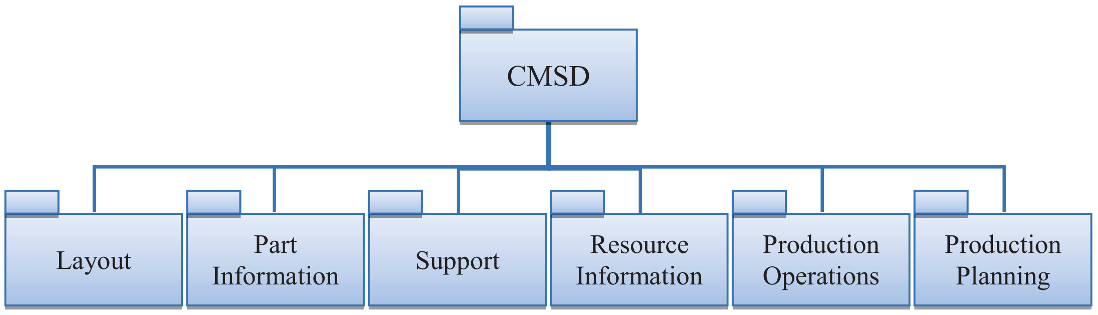


Figure 1: The packages of the CMSD Information Model (SISO 2010)

For more detailed information about CMSD, see SISO (2010), SISO (2011), and Johansson et al. (2007). It was shown in several papers that the CMSD standard can be used for many different applications in the context of simulation of production systems (Johansson et al. 2007). In the next subsections we first introduce our related work in the field of CMSD based model generation and initialization. Afterwards we introduce a CMSD based methodology for result data representation, transportation, storage, and visualization.

2.2 Automatic model generation as first precondition for web based simulation

Typically, one of the first steps of a simulation study (after definition of study goals and data collection) is the creation of the simulation model. In the context of web based simulation we can build the model in a specific web frontend that emulates the modeling tools of the chosen simulator or we can use approaches of "data-driven model generation". Data-driven model generation is characterized through model generation without using the modeling tools of the chosen simulator, rather the model is generated from external data sources using algorithms and interfaces of the simulator (Eckardt 2002; Bergmann and Strassburger 2010; Strassburger, Bergmann, Müller-Sommer 2010).

As discussed in some of our previous work we suggest the use of the CMSD standard as data standard for such a data-driven model generation. In our approach we firstly map all needed objects of a production system, e.g., machines, workers, but also process plans and job lists, into a CMSD compliant description. Secondly, we use this CMSD XML representation to automatically generate all required model elements in the selected simulator. This model generation can be based on different techniques. The spectrum ranges from XML Stylesheet Transformation for creating simulator source code to model generation performed from the simulator itself, i.e., by executing suitable model generation scripts in the simulator which reads the CMSD input data file and accordingly generates the appropriate model elements.

For more detailed information about CMSD based model generation see Bergmann, Fiedler and Straßburger (2010) or Bergmann et al. (2012).

2.3 An automatable mechanism for automatic initialization as second precondition for web based simulation

The second step of a web based simulation study is the initialization of the just generated simulation model. We define initialization as a way to setup simulation models in such a way that the model's internal control structures (event lists, random number generators, simulation clock, component states, etc.) reflect the current state of a real system with sufficient accuracy. This approach is often required in the context of online or symbiotic simulation (Aydt et al. 2008). For the initialization only data about the system load and the state of all resources is of interest (Bergmann, Stelzer and Strassburger 2011). All other aspects, like technical data (system layout, topology), are already reflected in the automatically generated simulation model, see Subsection 2.2.

In our approach we also used the CMSD standard for the initialization, so it is possible to map data for generation and initialization in one CMSD XML file. What data is most important for the initialization and how it is mapped in CMSD was described in detail in Bergmann, Stelzer and Strassburger (2010).

2.4 Last precondition for web based simulation – standardization of result data

The missing component to perform web-based simulation concerns the standardized transfer of the simulation result data to the end user. The result data must be represented in a way that all possible information needed for analyzing the simulated system are contained, since the goals and requirements of a simulation study can be very different. So we must select an abstraction level in which all information can be represented adequately. One can assume that the end user does not necessarily need all the detailed technical, organizational and management data, and that depending on the use case certain key figures will be favored, e.g., utilization, cycle time etc. Furthermore we can safely assume that most of the typically used key figures can be calculated from a small set of raw data. Some examples for relevant key figures and their calculation are shown in table 1.

Table 1: Examples for key figures

Name	**Formula**
Cycle time	$t_{cycle,i} = \|t_{start,i} - t_{end,i}\|$
Average cycle time	$t_{avg\ cycle} = (\sum_{i=1}^{n} t_{cycle,i})/n$
Setup time	$t_{setup,i} = \sum (\|t_{setup,i} - t_{start,i}\|)$
Adherence to delivery dates	$t_{adh,j} = t_{due\ date,j} - t_{end\ date,j}$

This mental model is analogous to a real production with data acquisition (PDA). In the production typically data about events on jobs, resources etc. are collected. These event data sets are often stored in relational databases. Typically, the events occur as a result of a status change of an object, e.g., a job starts working on a machine and is allocated a worker, or a machine fails. All kinds of these events can be described by a timestamp, an identifier, and, if necessary, references to related objects, like jobs or resources.

In the context of web based simulation we want to use this pattern in the way that we store all needed event data sets also in the CMSD data structure. The CMSD standard, however, is mainly designed to describe a system with its current state. Therefore the standard is well suited for describing a state of an element, e.g., the current status, current setup of a machine or an end time of a specific job. For the analysis of simulation results this kind of information alone does not suffice. For the storage of further result data, all information not directly representable in a CMSD class can be stored in the CMSD Class "Event" (Figure 2).

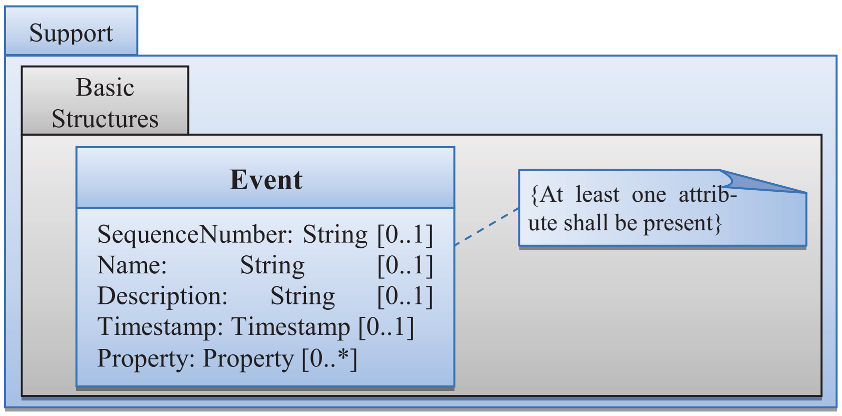

Support
Basic Structures
Event
SequenceNumber: String [0..1]
Name: String [0..1]
Description: String [0..1]
Timestamp: Timestamp [0..1]
Property: Property [0..*]
{At least one attribute shall be present}

Figure 2 : The CMSD Event Class, as Part of the Support/BasicStructures Package (SISO 2010)

The Event Class is part of the basic structure package and so part of the Support package (see Figure 1). The class Event is, according to the CMSD standard, only used by the JobEffortDescription Class which is located in the Production Operations Package. It therefore seems to be reasonable to use this class for result documentation purposes. The Event Class has only five kinds of attributes, in our approach we used four of these, see Figure 3.

```
… <Job>
   <Identifier>Job01</Identifier>
   <Status>released</Status>
   <PlannedEffort> …
      <DueDate>2012-02-26T07:00:00</DueDate>
      <ReleaseDate>2012-02-25T07:00:00</ReleaseDate>
      <ProcessPlan>
         <ProcessPlanIdentifier>ProcessPlanA1</ProcessPlanIdentifier>
      </ProcessPlan>
   </PlannedEffort>
   <ActualEffort> …
      <Event>
         <SequenceNumber>4</SequenceNumber>
         <Name>start setup</Name>
         <Timestamp>2012-02-26T09:07:23</Timestamp>
         <Property>
            <Name>ProcessStep</Name>
            <ProcessReference>
               <ProcessIdentifier>PP1Step1</ProcessIdentifier >
            </ProcessReference>
         </Property>
         <Property>
            <Name>usedResource</Name>
            <ResourceReference>
               <ResourceIdentifier>Ma3</ResourceIdentifier>
            </ResourceReference>
         </Property>
      </Event>
      <Event>
         <SequenceNumber>5</SequenceNumber>
         <Name>start work</Name>
         <Timestamp>2012-02-26T09:17:32</Timestamp>
         <Property>
            <PropertyDescription> … </PropertyDescription>
            <Name>ProcessStep</Name>
            <ProcessReference>
               <ProcessIdentifier>PP1Step1</ProcessIdentifier >
            </ProcessReference>
         </Property>
         <Property>
            <Name>usedResource</Name>
            <ResourceReference>
               <ResourceIdentifier>Ma3</ResourceIdentifier>
            </ResourceReference>
         </Property>
      </Event>
   </ActualEffort>
</Job> …
```

Figure 3: Snapshot of a CMSD XML Job Description with Events

First the SequenceNumber orders the events in a logical order. Every event has a unique number. Second the Name classifies the type of Event, we use a enumeration for possible values. The enumeration contains values such as start setup, start work, end work, machine broken, machine repaired, etc. Third the Timestamp contains the date and time when the event occurred, the representation is defined according to ISO 8061. Finally we use at least one user-property. This property is used to build a relation to one or more objects involved in this event, e.g., an involved worker or machine. The involved job is identifiable through the hierarchy of the CMSD document, because the event class is used inside a Job or more precisely a JobEffortDescription of a Job. An example of how we use the event class within a JobEffortDescription is shown in Figure 3.

In the example, a part of the description of the job "Job01" is shown, its release date is February 25, 2012 at 7 am. The delivery date is set to February 26, all production steps of the job are described in process plan "ProcessPlanA1".

In the example 2 events have been registered. At 7:23 am, the first example event with the sequence number 4 has been recorded. It characterizes the start of a setup process on machine "Ma3" which is needed for the process step "PP1Step1" (a step of the deposited process plan "ProcessPlanA1"). The second event entry was created after completion of the setup process, i.e., at the start of the actual processing of the process step.

As mentioned above, the use of the Event Class is according to CMSD strictly made in conjunction with jobs only. This may be problematic, when event information must be conveyed that does not directly relate to a job, e.g., when a machine breaks while no job is currently working on it.

If this type of event is considered relevant for result evaluation, we have different options to circumvent this limitation. For instance, a simple way for managing such events is to co-locate this event with the last known job on this machine. For this alternative we do not need any extra property or the like, but it is not a 100% logically correct representation. A second way is to convert from a job oriented view of events to a machine oriented view as it would occur in real PDA. This could be done for all events or only for special events. This alternative is logically correct, but is problematic as resources, like machines, in the CMSD standard do not have an Event attribute. For this approach, we must use a user-property to extend the standard, e.g., a reference property to the Event Class.

For first tests of our web based distribution of simulation experiments we have assumed that events without an involved job are negligible.

3 A PROTOTYPE FOR WEB BASED SIMULATION AND DISTRIBUTION OF SIMULATION EXPERIMENTS

After we have discussed all preconditions for web based simulation, we introduce our actual prototype for web based simulation and the distribution of experiments. The basic architecture for web based distribution of simulation experiments, based on the CMSD standard, is shown in Figure 4.

In the prototype, only two components in the complete workflow are visible for the end user. One is the "web based user interface" and the other is the "web based simulation monitor/statistics". Both are implemented as ASP.Net applications.

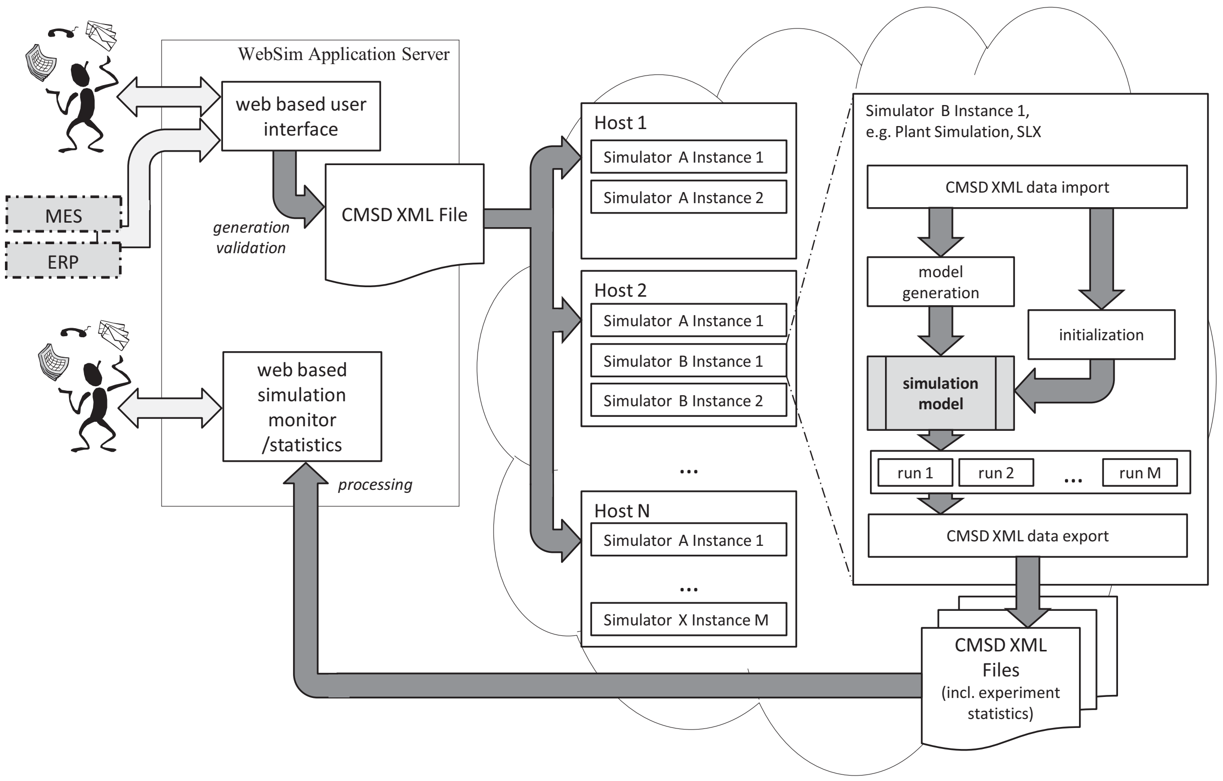


Figure 4: Architecture of a web based prototype for distribution of simulation experiments based on the CMSD standard

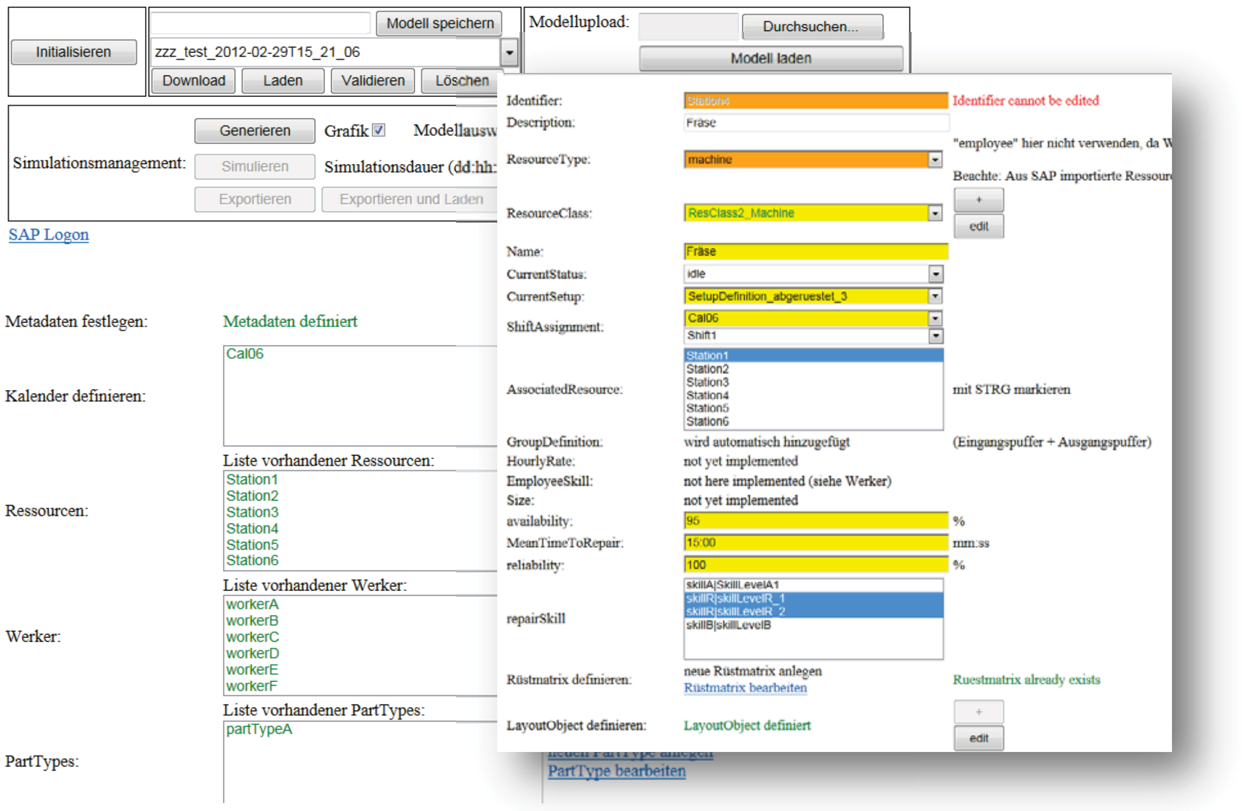
The web based user interface is our central tool for the generation and administration of CMSD files. In this tool it is possible to import and refine data from external applications, like Enterprise Resource Planning Systems (e.g., SAP R/3), Manufacturing Execution Systems or other Planning or Execution Systems, see Figure 5. Typically the data which is imported has not the required detailing and quality for the next steps, i.e., for the simulation model generation. Therefore the web based user interface offers features for editing the raw data and for checking its completeness. Missing data can so be detected and completed, e.g., a decision rule on a buffer. This functionality of the web based user interface allows to complete the data and can be used to generate additional data entries, e.g., additional machines or workers. Furthermore, the web based user interface has various functions to manage, import, and export of CMSD files.

Finally, a CMSD compliant XML file can be generated and passed to a defined set of simulator instances. For statistical reasons and due to the stochastic behavior of simulation the results of a single simulation run are often not meaningful enough. Therefore often a number of independent simulation runs are required.

The initialization of random number sequences in the respective simulation tools relies on simulator specific randomization techniques. In Plant Simulation, for instance, we use its internal mechanisms to randomize the seed values. For other simulation systems, randomization can be performed in the simulation system based on system time as well as unique system properties.

Three basic kinds of approaches are possible to implement this. The first approach uses a single simulator instance which sequentially perform all desired simulation runs. This is often supported by the simulation systems themselves, but may take significant amounts of execution time. The second approach is to use as many simulator instances as simulation replications are needed, and let each instance perform exactly one simulation run. The third approach is a hybrid of the previous two approaches in which multiple simulator instances are used but every instance shall perform one or more simulation runs.

Although our CMSD based solution can support all three approaches, for reasons of this paper we focus on the third approach. This approach is best suited to convey the basic intention of our paper of giving optimal use to all available simulator instances, even when they are instances of different simulation tools. A simulator instance is in this case a standalone simulation environment, it is identified by a host and operation system task of the simulator tool. Each instance must have a CMSD compliant model generator component (see chapter 2.2), a CMSD compliant model initialization component (see Subsection 2.3) and must have abilities to create CMSD compliant result files (see Subsection 2.4). If these prerequisites are fulfilled it is easily possible to call such a simulator instance by passing a CMSD file to it. The result is also represented as a CMSD file which is passed back asynchronously.



zzz_test_2012-02-29T15_21_06

Simulationsmanagement:

Metadaten festlegen:	Metadaten definiert
Kalender definieren:	Cal06
Ressourcen:	Liste vorhandener Ressourcen: Station1, Station2, Station3, Station4, Station5, Station6
Werker:	Liste vorhandener Werker: workerA, workerB, workerC, workerD, workerE, workerF
PartTypes:	Liste vorhandener PartTypes: partTypeA

Identifier:	Station4	Identifier cannot be edited
Description:	Fräse	"employee" hier nicht verwenden, da W
ResourceType:	machine	Beachte: Aus SAP importierte Ressour
ResourceClass:	ResClass2_Machine	
Name:	Fräse	
CurrentStatus:	idle	
CurrentSetup:	SetupDefinition_abgeruestet_3	
ShiftAssignment:	Cal06; Shift1	
AssociatedResource:	Station1, Station2, Station3, Station4, Station5, Station6	mit STRG markieren
GroupDefinition:	wird automatisch hinzugefügt	(Eingangspuffer + Ausgangspuffer)
HourlyRate:	not yet implemented	
EmployeeSkill:	not here implemented (siehe Werker)	
Size:	not yet implemented	
availability:	95	%
MeanTimeToRepair:	15:00	mm:ss
reliability:	100	%
repairSkill	skillA|SkillLevelA1, skillR|skillLevelR_1, skillR|skillLevelR_2, skillB|skillLevelB	
Rüstmatrix definieren:	neue Rüstmatrix anlegen; Rüstmatrix bearbeiten	Ruestmatrix already exists
LayoutObject definieren:	LayoutObject definiert	

Figure 5: Screen shot of the "web based user interface"

There are different approaches to the detailed procedure of distributing simulation runs .We investigate three cases (A1, A2 and B) and discuss their pros and cons, see Figure 6. In all variations the web based user interface calls the simulation instances, always all technically required data (e.g., host and socket) is available in a database.

In the variation A1 and A2 every single simulation instance receives exactly one CMSD file and an additional parameter, which specifies the count (rc) of the requested simulation runs. In variant A1 all runs are equally distributed between the available instances, in variant A2 the quota of runs depends on the performance of each instance, represented by an indicator in the database. The general handicap of the variant A1 and A2 is that every instance needs additional functionalities for managing multiple simulation runs. Additionally, variant A1 is only as fast as the slowest instance, variant A2 often has a higher overall performance but detailed performance key figures for each instance are needed.

In variant B, all logic for distributing the simulation runs are implemented in the web based user interface, every simulator instance needs only functionalities for single runs. In this variant each run will be separately triggered, i.e. in the first cycle, all available simulator instances will be called, also a CMSD file will be transferred. If a simulation run on an instance is finished it will return a CMSD result file to the server. On the server side a simple counter is used to identify if a further simulation run on this now empty instance is necessary. Variant B has a lot pros and less important cons, except the higher data volume for exchanging CMSD files. Positive is the lower implementation effort (only on server side), and that no performance key figures for the instances are needed and that the impact of a slow instances, if applicable, is not huge. Due to the fact that bandwidth is typically not a bottleneck, our prototype only implements variant B at the moment.

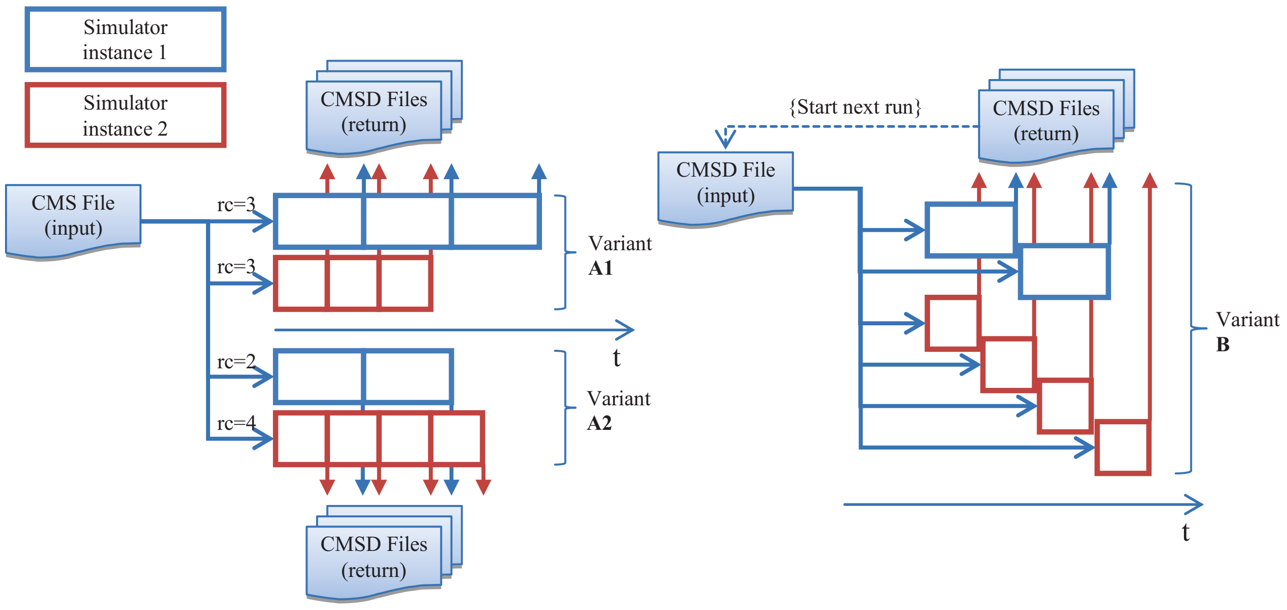


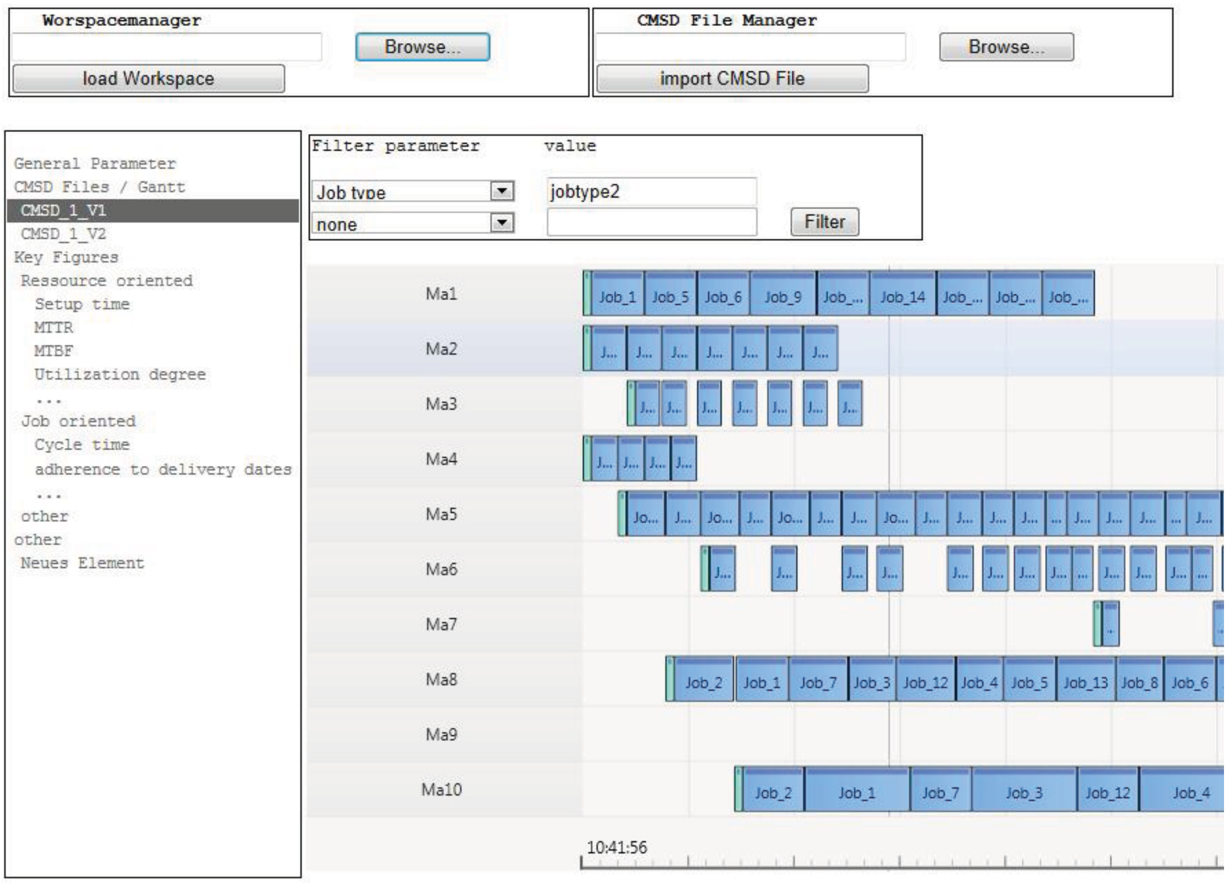
Figure 6: Variants for CMSD based distribution

The final component in the workflow and the second tool visible to the end user is the "web based simulation monitor". In this also web based user interface all functionalities for the analysis of simulation runs are integrated. It is possible to import single or multiple CMSD files and to analyze the included result data (see Subsection 2.4).

In this prototype already a lot of functionalities are available. Before detailed analysis functions can be used, a set of associated CMSD XML files must be imported to the workspace. All further functions use only the data in the workspace. In the current version of the tool a set of methods to calculate key figures are implemented. These methods can be applied for a single simulation run, for sets of multiple simulation runs, or for only a subset of the contained data items. For example, the method to calculate the "average cycle time" can use a single run, a set of all performed runs, or only a selected job type in all performed runs. The sub sets are generated by a general filter mechanism. Such filters can be used in many ways in the prototype.

If it is of interest to the user, there are functions available for building arithmetic average, confidence intervals, standard deviation, etc. of the key figures. The most calculated values are displayable in corresponding diagrams, a screenshot of the prototype is shown in Figure 7.

It is very helpful to present single runs as "Gantt diagram", as these allow a quick insight into the overall system performance. Therefore the "Gantt" functionality was implemented in the prototype.



Filter parameter	value
Job type	jobtype2
none	

Figure 7: Screenshot of the web based simulation monitor prototype

The prototype has been tested with several scenarios implementing different job shop problems. Simulation results were further validated against results from replications performed in a traditional manner, i.e., on a single computer with subsequent sequential simulation runs. The results of the simulation by both methods yields similar results. The simulator combination used in the test cases had no significant impact on the statistics of the results. So it is asserted that the methodology for web based simulation enables end users to simulate production systems only over a web frontend without a need to decide for a certain simulator. It should be noted that in the current state of development the used model generators do not support all CMSD object classes. Some elements not typical to job shop problems have been omitted (e.g., conveyor systems). This is not a limitation of our conceptual framework, but rather attributed to the complexity of CMSD.

4 SUMMARY AND DISCUSSION

This paper introduced a new methodology for the distribution of simulation experiments on the web. The methodology assumes a CMSD based description of the simulated manufacturing system. Since all user interactions are made in the web based user interfaces, no further knowledge about the simulation tool used is needed. With this approach the usage of several simulators can be combined into one workflow without the need of additional user interaction.

The prerequisite for the suggested methodology is the implementation of suitable model generators, which use the CMSD standard for all data input and output. The advantage of the methodology is its flexibility in regard to the count and kind of simulation tools.

Future work includes testing the methodology with larger manufacturing systems. Depending on these scenarios, additional functionality will be added to the model generator, e.g., for supporting conveyors. Another venue of future work may include the extension of the web based simulation monitor. Finally, procedures have to be implemented that make sure that the simulation always terminates within a certain time limit. This will also involve the detection and treatment of failed simulation runs, e.g., with time out mechanisms.

REFERENCES

Aydt, H., Turner, S.J., Cai, W., Low, M. Y. H. 2008. "Symbiotic simulation systems: An extended definition motivated by symbiosis in Biology". In *Proceedings of the 22nd Workshop on Principles of Advanced and Distributed Simulation*: 109–116.

Bergmann, S., Fiedler, A., Straßburger, S. 2010. "Generierung und Integration von Simulationsmodellen unter Verwendung des Core Manufacturing Simulation Data (CMSD) Information Model" (Generation and integration of simulation models using the Core Manufacturing Simulation Data (CMSD) information model (in German)). In *Proceedings of the 14th ASIM Dedicated Conference on Simulation in Production and Logistics - integration aspects of simulation referring to equipment, organization and personne*l, Edited by G. Zülich and P. Stock, Karlsruhe Institute of Technology (KIT): 461-468.

Bergmann, S., Strassburger, S. 2010. "Challenges for the Automatic Generation of Simulation Models for Production Systems." In *Proceedings of the 2010 Summer Simulation Multiconference*, July 11-14, 2010. Ottawa, Canada: 545-549.

Bergmann, S., Stelzer, S., Strassburger, S. 2011. "*Initialization of Simualtion Models using CMSD*". In *Proceedings of the 2011 Winter Simulation Conference*, Edited by S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, December 11-14, 2011. Phoenix, AZ: 2228-2239.

Bergmann, S., Stelzer, S., Wüstemann, S. and Strassburger, S. 2012. *"Model generation in SLX using CMSD and XML Stylesheet Transformations"*. In *Proceedings of the 2012 Winter Simulation Conference*, December 9-12, 2012. Berlin, Germany.

Eckardt, F. 2002. *"Ein Beitrag zu Theorie und Praxis datengetriebener Modellgeneratoren zur Simulation von Produktionssystemen (A contribution to theory and praxis of data drive model generators for the simulation of production systems (in German))."* Aachen, Shaker, 2002.

Fowler, J. W., Rose, O. 2004. "*Grand Challenges in Modeling and Simulation of Complex Manufacturing Systems*." In *SIMULATION: The Society for Modeling and Simulation International*, 80(9): 469–476, September 2004.

Gartner 2011. "*Gartner Executive Programs Worldwide Survey of More Than 2,000 CIOs Identifies Cloud Computing as Top Technology Priority for CIOs in 2011*". Gartner Inc. Accessed February 14, 2012. http://www.gartner.com/it/page.jsp?id=1526414.

Gartner 2012. "*Gartner Executive Programs' Worldwide Survey of More Than 2,300 CIOs Shows Flat IT Budgets in 2012, but IT Organizations Must Deliver on Multiple Priorities*". Gartner Inc. Accessed February 14, 2012. http://www.gartner.com/it/page.jsp?id=1897514.

Henriksen, J. O. 1999. "SLX - The X is for eXtensibility." In *Proceedings of the 1999 Winter Simulation Conference*, Edited by P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, December 5-8, 1999. Phoenix, AZ: 167-175.

Johansson, M., Leong, S., Lee, Y. T., Riddick, F., Shao, G., Johansson, B., Skoogh, A., and Klingstam, P. 2007. "*A Test Implementation of the Core Manufacturing Simulation Data Specification*". In *Proceedings of the 2007 Winter Simulation Conference*, Edited by S. G. Henderson, B. Biller, M.-H Hsieh, J. Shortle, J. D. Tew, and R. R. Barton. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.: 1673-1681.

Siemens 2012. *"Plant Simulation*". Product Lifecycle Management Software Inc. Accessed March 28, 2012. http://www.plm.automation.siemens.com/en_us/products/tecnomatix/plant_design/plant_simulation.shtml.

SISO 2010. "*Standard for: Core Manufacturing Simulation Data – UML Model*". Core Manufacturing Simulation Data Product Development Group. Accessed February 14, 2012. http://www.sisostds.org/DigitalLibrary.aspx?Command=Core_Download&EntryId=31457.

SISO 2012. "*Standard for: Core Manufacturing Simulation Data – XML Representation*". Core Manufacturing Simulation Data Product Development Group. Accessed February 14, 2012 http://www.sisostds.org/DesktopModules/Bring2mind/DMX/Download.aspx?Command=Core_Download&EntryId=32733&PortalId=0&TabId=105.

Strassburger, S., Bergmann, S., Müller-Sommer, H. 2010. *"Modellgenerierung im Kontext der Digitalen Fabrik - Stand der Technik und Herausforderungen"* (Model Generation in the Digital Factory Context- State-of-the-Art and Challenges (in german)). In *Proceedings of the 14th ASIM Dedicated Conference on Simulation in Production and Logistics - integration aspects of simulation referring to equipment, organization and personne*l, Edited by G. Zülich and P. Stock, Karlsruhe Institute of Technology (KIT): 35-42.

Taylor, S., Mustafee, N., Kite, S., Wood, C., Turner, S., Strassburger, S. 2010. *"Improving Modeling and Simulation Through Advanced Computing Techniques: Grid Computing and Distributed Simulation".* In: *Proceedings of the 2010 Winter Simulation Conference*, Edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan. December 5-8, Baltimore, MD: 216-230.

AUTHOR BIOGRAPHIES

SÖREN BERGMANN is a Ph.D. student at the Ilmenau University of Technology. He is a member of the scientific staff at the Department for Industrial Information Systems. He received his diploma degree in Information Systems from Ilmenau University of Technology. Previously he worked as corporate consultant in various projects. His research interests include generation of simulation models and automated validation of simulation models within the digital factory context. His email is soeren.bergmann@tu-ilmenau.de.

SÖREN STELZER is a Ph.D. student at the Ilmenau University of Technology. He is a member of the scientific staff at the Department for Industrial Information Systems. He received his diploma degree in Computer Science from the Ilmenau University of Technology. During his study he worked in the Neuroinformatics and Cognitive Robotics Lab of the Ilmenau University of Technology. After his study he worked in optimization of power plants in several projects. His research interests are simulation based optimization, model predictive control, artificial learning and discrete event simulation. His email is soeren.stelzer@tu-ilmenau.de.

STEFFEN STRASSBURGER is a professor at the Ilmenau University of Technology in the School of Economic Sciences. Previously he was head of the "Virtual Development" department at the Fraunhofer Institute in Magdeburg, Germany and a researcher at the DaimlerChrysler Research Center in Ulm, Germany. He holds a Ph.D. and a Diploma degree in Computer Science from the University of Magdeburg, Germany. He is a member of the editorial board of the Journal of Simulation. His research interests include distributed simulation as well as general interoperability topics within the digital factory context. He is also the Vice Chair of SISO's COTS Simulation Package Interoperability Product Development Group. His web page can be found via www.tu-ilmenau.de/wi1. His email is steffen.strassburger@tu-ilmenau.de.