A REAL WORLD PILOT IMPLEMENTATION OF THE CORE MANUFACTURING SIMULATION DATA MODEL

Marcus Johansson
Björn Johansson

Department of Product
and Production Development
Chalmers University of Technology
Gothenburg, SE-412 96, SWEDEN

Swee Leong
Frank Riddick
Y. Tina Lee
Manufacturing Systems
Integration Division
National Institute of Standards and Technology
Gaithersburg, MD 20889-8261, USA

ABSTRACT

While software for discrete event simulation (DES) has emerged into sophisticated tools for decision support in a wide range of contexts, the need to integrate DES tools with other applications is increasing. In the industrial engineering context, simulation engineers strive to use real-world data, e.g. logs of machine breakdown, to make behavior of DES models imitate reality. However, the format used for describing simulation data is often specialized to the current situation. The Core Manufacturing Simulation Data (CMSD) is a collaborative effort with academia and industry to standardize the format used for simulation data, to facilitate data exchange among simulation and manufacturing applications. This paper describes the results from a pilot implementation study at Volvo Trucks, where CMSD was utilized as the data exchange format between two data systems and two DES models. The DES tools used were commercial software packages Unigraphics *Plant Simulation* and InControl *Enterprise Dynamics*. Generic and reusable interfaces for CMSD-file communication were developed for each of these tools. The CMSD interfaces were successfully connected to a model in each simulation tool describing the same real-world manufacturing process. A stand alone application was developed to collect and analyze raw data and to create the CMSD file being used as input data for both models. The result is a system including raw data analysis, data reformatting, CMSD interfacing and model execution. Based on the result, a generic methodology for CMSD interface development in DES tools has evolved. The most important conclusion is that CMSD data can be interpretable by both Plant Simulation and Enterprise Dynamics, and that it saves engineering development time during the model building phase.

1 INTRODUCTION

Vendors of discrete event simulation (DES) software market their products as powerful tools to analyze various flow and queue systems. Companies in the automotive industry often demand outcome estimations of major investments to be verified through simulation (Ulgen and Gunal 1998).

However, many reasons make simulation expensive. One of those factors is the need of valid time domain data to base simulation on. Such data is frequently referred to as *input data*. Input data management generally stands for a big part of the total time needed for a simulation study (Umeda and Jones 1997). Because of how simulation often is used: to verify outcome estimation of specific investments, 70-80% of automotive simulation models have a short life-cycle (Ulgen and Gunal 1998). Hence, reducing work associated with input data preparations could lead to economical benefits for industry.

1.1 Background

Modern factories use sophisticated data systems that logs events on the shop-floor. Typically the systems logs start/stop events of machine operation and breakdown cycles. Because of format incompatibility, simulation can seldom access and use the data directly. Excessive efforts have to be spent on manually sorting, analyzing, and formatting the data in a way that fits the used simulation package (Skoogh and Johansson 2007).

Voices from the simulation community have stated the need of interoperability standards (Banks et al. 2003). Experts' forums such as the Winter Simulation Conference host specific tracks for simulation interoperability.

With means of interoperability, simulation and manufacturing data storages could automatically exchange information. To support interoperability, applications need to structure and communicate information uniformly. Several standardization efforts address this need. The Core Manufacturing Simulation Data (CMSD) is one of those efforts (Leong et. al. 2006). The United States' National Institute of Standards & Technology (NIST) leads this CMSD effort.

1.2 Problem description

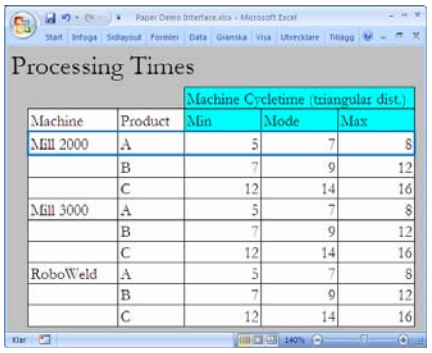
Many simulation packages can already communicate with other data sources. Excel, ODBC, text file, and XML represents commonly supported formats. Such formats can transfer any data and enable basic interoperability. Yet, they do not specify how the data should be structured. When using such formats, the modeler not only has to know *what* data he needs. He also needs to know *where* to get it.

Example 1: When reading input data from Excel files, the modeler have to map specific Excel cells to specific attributes of simulation objects. Requests to retrieve data must specify exactly where the data can be found: in what worksheet, in what column, and on which row.

To gather and compute a cycle time for *Mill 2000* in Figure 1 below, you would have to write a formula for your machine cycle time such as:

```
Triangular(
 ExcelRead(4,4), {min}
 ExcelRead(4,5), {mode}
 ExcelRead(4,6)  {max}
)
```

If data was populated in a common structure, like CMSD, generic functions such as `GetCycleTime(Product A)` could be used instead. The modeler would only need to know *what* data to get. The function would figure out *where* to get it. Manual work, time, and cost would be reduced. However, currently no simulation package provides such `GetCycleTime` functions.



Processing Times

Machine	Product	Machine Cycletime (triangular dist.)		
		Min	Mode	Max
Mill 2000	A	5	7	8
	B	7	9	12
	C	12	14	16
Mill 3000	A	5	7	8
	B	7	9	12
	C	12	14	16
RoboWeld	A	5	7	8
	B	7	9	12
	C	12	14	16

Figure 1: Excel sheet cycle time example. A frame encloses the data addressed by the command written above.

1.3 Previous work

Previously, we have successfully conducted a pilot project to represent input data with CMSD. A paint shop model was built in the simulation package Enterprise Dynamics (Johansson and Zachrisson 2006). The model read input data from a CMSD file by running a specialized script (Johansson et al. 2007). However, the script was not generic. It could only be used for that specific paint shop model.

We concluded that CMSD could represent input data in a way that could be interpreted by Enterprise Dynamics. Still, you had to write your own import script for each model you wanted to connect to a CMSD file.

1.4 Purpose

The purpose of this paper is to go one step further than before, and show that CMSD data is generic and can be used by several simulation packages. The approach is to create generic *CMSD interfaces* for two simulation packages. The interfaces shall be usable for many model building occurrences. This will in this case be completed for two simulation packages: Plant Simulation and Enterprise Dynamics.

We encourage development of CMSD interfaces for other simulation packages. For that purpose, our experiences from this project are documented in this paper as a guideline for others to use.

Figure 2 displays the position of a CMSD interface in a targeted integrated data flow. The data flow encompasses transformation of raw data at a manufacturing process to input data for a simulation model of that process.

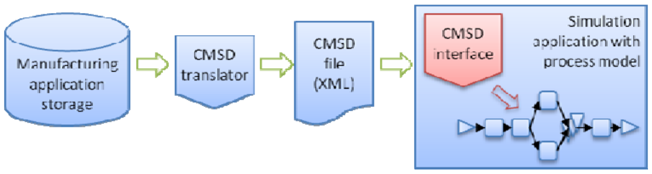

Manufacturing application storage
CMSD translator
CMSD file (XML)
CMSD interface
Simulation application with process model

Figure 2: The figure shows the input data flow that this work is intended to support. The CMSD interface shape represents the focus of this paper.

1.5 Goal

This pilot project reduced the lead-time for modeling a system when CMSD structured input data is available. Novice CMSD skills should be enough for using the CMSD interfaces.

Users experienced of the involved software shall be able to further refine the developed interfaces. Modular architecture of the interfaces shall ensure extensibility.

1.6 Outline

Section two describes the need of standards through an everyday example. Current simulation standardization efforts are then briefed. Section two ends with a short description of CMSD.

Section three shows two examples of implementations: one in Enterprise Dynamics and another in Plant

Simulation. A real world case study where the CMSD interface was used is also presented.

Section four presents our gathered experience, requirements, supports and guidelines for developing CMSD interfaces.

Section five discusses the results and its applicability for future modeling.

Section six describes our future initiatives and development plans. It also invites vendors to utilize CMSD as a part of their software package offerings.

2 INDUSTRIAL NEEDS FOR INTEROPERABILITY STANDARDS

This section describes the need of standards through an everyday example. Current simulation standardization efforts are then briefed. Section two ends with a description of CMSD.

2.1 How standards helps us in everyday life

Imagine you buy some new electronic devices or appliances. It could be an iron, a monitor, or whatever lies close to your imagination. Imagine you would have to do the wiring manually each time you want to connect the device to electric power or peripherals. It would take time and you would have to know exactly how to connect the wires. For electric power it would also be unsafe.

Luckily, you do not face this problem, because the cables have contacts. Depending on the purpose of the cable, the contact looks different and fits into a specific slot or outlet. As an example, the contact that connects your monitor to your computer look different than the one you connect to the power outlet. Different standards specify different contacts and the contact plays the part of an interface between the items you connect.

2.2 Standards and standardization efforts relevant for simulation

When it comes to simulation, there is no “contact” where you can plug in your input data. You have to do the “wiring" yourself. But there are standards and efforts to support development of such contacts or interfaces. Currently we see standards for simulation that evolve with different focus:

- Unigraphics SDX specification focuses on automatic model generation based on layout CAD-drawings (Sly and Moorthy 2001)
- SysML features a UML based neutral language for systems modeling that covers more than just simulation. (Huang et al. 2007; SysML 2007)
- CSPI evolves to supports High Level Architecture for runtime communication among simulation models. (Taylor et al. 2006)
- ISA-95 is a data standard for manufacturing execution systems, which can be related to simulation (ANSI/ISA 2000).

CMSD is an information model for exchanging simulation-relevant manufacturing data between manufacturing applications and simulation systems. CMSD typically addresses the need to structure data for simulation rather than describing the behavior of a simulation model.

Maybe in the future, commercial simulation packages will provide interfaces for each of these standards and efforts. Since our work is focused on reducing work related to input data, we chose to work with CMSD and develop interfaces enabling two simulation packages can connect to CMSD data.

2.3 Brief description of CMSD

Core Manufacturing Simulation Data is intended to be a neutral file format for manufacturing applications that exchange data with simulation models. The file format is based on the extensible markup language, XML. CMSD is defined by an information model which is specified through UML diagrams. Six UML packages group related data. These packages are:

- CMSD package
- Resource Information Package
- Part and Inventory information Package
- Production Operations Package
- Production Planning Package
- Support Package

The packages contain several structures that can be used to structure simulation input data with. Leong et al. (2006) gives a detailed description of these packages. The information model (CMSD 2006) provides the complete specification of CMSD.

3 DEVELOPED CMSD INTERFACES

This section presents the CMSD interfaces developed in our project: One in Enterprise Dynamics and another in Plant Simulation. The section ends by presenting a real-world case study where these CMSD interfaces were used.

3.1 CMSD interfaces – how to use it

The CMSD interfaces are developed as distributable objects. The user loads the interface into the simulation environment using standard routines. Figure 3 shows the interfaces loaded into object libraries in Enterprise Dynamics and Plant Simulation.

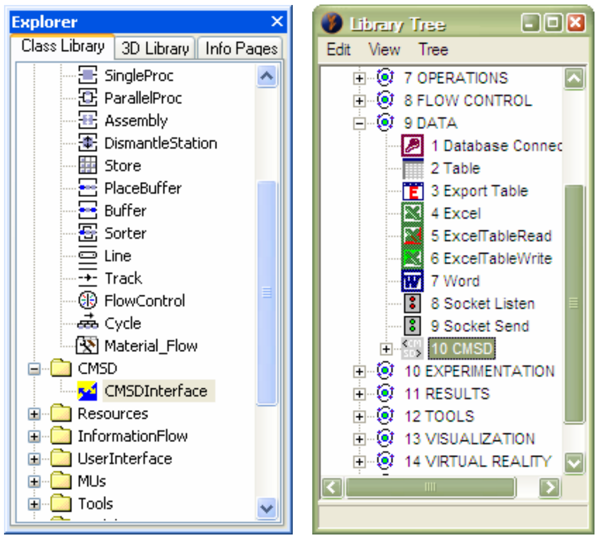


Figure 3 The CMSD interface can be seen in the class library of Plant Simulation to the left, and the atom library of Enterprise Dynamics to the right.

To use the CMSD interface in the model, you simply drag it from the objects library and drop it into the model layout. Figure 4 and Figure 5 show simple example models in both simulation packages where the CMSD interface is used.

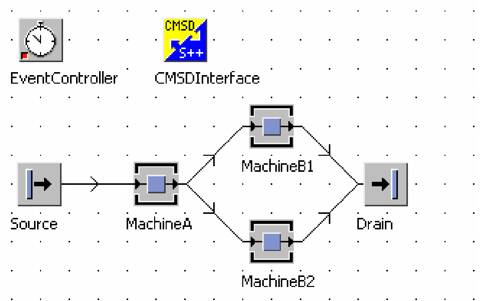


Figure 4 Plant Simulation example model.

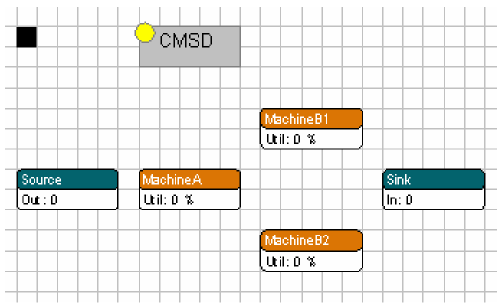


Figure 5 Example Enterprise Dynamics model.

When added to a model, the CMSD interface can be right-clicked to pop up a context menu. From the context menu the user can :

- load data from a CMSD XML file – an “open file” dialog will appear
- reload data from an already chosen file
- examine and edit CMSD data – a table structure or graphic representation of the data will appear.

The CMSD interface for Plant Simulation provides more functionality than the interface for in Enterprise Dynamics. Other than reading and viewing CMSD XML files, the Plant Simulation implementation also gives the user possibility to:

- turn on safe execution mode – handles non-valid CMSD files without hangs, but slows down simulation
- turn on debugging – prints log files based on results from data retrievals and messages from the safe execution mode
- turn on a work-logging function – adds data to CMSD based on the simulation
- write CMSD XML files – a “save file” dialog will appear

Figure 6 shows the context menu for the Plant Simulation CMSD interface.

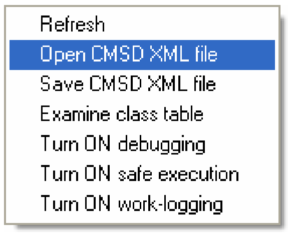


Figure 6 The context menu of the CMSD interface for Plant Simulation.

3.2 CMSD interface - user functions

By adding the CMSD interface to your model, a set of user functions is enabled. Those user functions can be used in other simulation objects to easily retrieve CMSD data. Some user functions instead provide functionality to add data to CMSD.

Depending on the implementation, user functions are named differently. Our intention is to use the same vocabulary as the simulation package to make the user feels familiar with function names. The following sections describes the Plant Simulation CMSD interface in more details. The developed user functions are:

- GetProcessingTime
- GetFailureInterval
- GetFailureDuration
- SetResouceSettings
- SetShiftCalendarSettings
- AddValue

- AddEmptyInstance

3.2.1 Get_ user functions

All user functions named Get_ returns specific data. For example:

GetProcessingTime returns an operation time based on the involved machine and product.

GetFailureInterval returns a Mean Time Between Failure duration for a resource.

GetFailureDuration returns a Mean Time To Repair duration for a resource.

3.2.2 Set_ user functions

All user functions named Set_ changes specific object parameters. These functions are intended to be used in the initialization phases of simulations. SetResourceSettings connects resources to work shift schedules according to Resource definitions in CMSD. SetShiftCalendarSetting sets up work shifts according to Shift definitions in CMSD.

3.2.3 Add_ user functions

All user function named Add_ help *adding* data to CMSD. AddEmptyInstance creates a new data structure according to any of hundreds of data structures defined in the CMSD information model. AddValue is used to set specific attributes on those data structures. To support AddEmptyInstance, templates of all CMSD structures are included in the CMSD interface. Using Add_ functions requires knowledge of CMSD and programming skills in Plant Simulation.

3.2.4 Advanced about Get_ and Add_ functions

For all Get_ functions above, a duration is returned. No matter what unit is used in CMSD to define the specific duration, a value converted to seconds is generated. If the duration was defined by a distribution function in CMSD, a value of that distribution function will be randomly generated each time the function is called.

A support function called *ComputeDuration* handles unit conversion and distribution computations for all user functions requiring such functionality. Get_ functions *locates* the data in the CMSD structure, whereas ComputeDuration *computes* the data.

CMSD interface developers can use the Add_ functions to create their own functions. To demonstrate this, we created the work logging function. The work logging function populates Job and Task structures of the current CMSD data based on generated products in the simulation model. As an example, Job and Task structures define start time, stop time and duration of planned and actual work efforts.

3.3 CMSD interface – real world test case

To test the developed CMSD interfaces, an automotive engine assembly process is modeled in both Enterprise Dynamics and Plant Simulation. The engine line assembly process includes two parallel lines with nine workstations each. Figure 7 shows an outline of the process. Truck engines arrives at workstation one. Gearbox, clutch, servos, turbo, and etc. are mounted at the rest of the workstations.

Both models connected successfully to the same real world input data, represented with CMSD. The user functions provided a fast and accurate way to establish the connections. Compared to writing explicit scripts to manually connect the model to the raw data, using CMSD interface saved engineering time considerably.

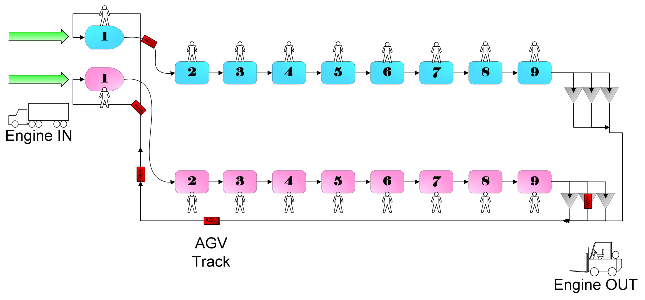


Figure 7: Engine line outline.

3.4 CMSD interface – extensibility

In section 3.2.4 above, we explained how all Get_ functions make use of the support function ComputeDuration. This modular approach is used throughout the CMSD interface architecture. Several support functions further facilitates CMSD interfaces to a state where complete information model could be supported. Interoperability of functions and the CMSD interface source code are highly documented.

4 REUSABLE DEVELOPMENT GUIDLINE

One result of this project is a development guideline. The guideline can be used for other CMSD interface implementations. This section can also be read as an “behind the curtain” experience by interested readers.

This section presents the guideline for CMSD interfaces in general. The section starts with a summary for those interested in high level information only.

4.1 Summary of Recommended Development Guideline Requirements

We have identified some core requirements that CMSD interfaces should fulfill. These requirements include:

- functionality to read XML files
- ability to represent CMSD data internally
- a simple user interface
- user functions that can be used to easily retrieve CMSD data
- functionality to handle data entries of all quantities and units recognized by CMSD
- extensible architecture

Beside these requirements, we also set a list of alleviations. A CMSD interface:

- may assume that a CMSD file fulfills the CMSD specification
- is not required to be able to write CMSD files
- may recognize a subset of the CMSD specification
- may require rich programming and CMSD skills of the developer that furthers the interface

4.2 Functionality to read XML files

CMSD data is meant to be exchanged through XML files. A prerequisite to build a CMSD interface in a simulation package is that XML files can be read. Some simulation packages provide built-in functionality to exchange data with XML files. Some packages can act as ActiveX clients. With means of ActiveX, XML data can be read through calls to AcitveX objects such as MSXML (MSDN 2007a).

We agree that CMSD interfaces can be developed for simulation packages supporting any of the solutions mentioned here. However, ActiveX may require the developer to have programming skills that lies beyond what is normally required for using programming languages in simulation packages. Typically, MSXML can be hard to use (MSDN 2007b).

Both simulation packages used in this study provide built-in functions to read and write XML files.

4.3 Ability to represent CMSD data internally

CMSD data should be viewable and editable on the receiver end of the interface as well as in the CMSD XML-file itself. In this pilot, the CMSD data is represented with nested tables in the simulation package.

4.4 A simple user interface

The user interface has to be simple in order to lessen the requirements on the user and also to enable automatic data input to the simulation. A simple user interface saves engineering time,

4.5 User functions that can be used to easily retrieve CMSD data

Name functions according to software nomenclature so as not to mislead the user. This is also a part of the goals to be user friendliness and to save enginnering time on the input data management side of the simulation project.

4.6 Functionality to handle data entries of all quantities and units recognized by CMSD

The developed CMSD interfaces assume metric (SI) units are used in the simulation.

4.7 Extensible architecture

The developed CMSD interfaces are open ended, extensible, and reconfigurable and further modification of the CMSD structures in the future are welcome. Only one change in the interface module of CMSD is needed to reflect changes allover.

5 ADVANTEGES OF USING THE CMSD INTERFACES

By introducing CMSD while building models in Enterprise Dynamics and Plant Simulation a development time reduction of about 85% can be realized. Decreased time reduction comes with system complexity, which will need further development of the CMSD interface. The interfaces for Enterprise Dynamics and Plant simulation are already prepared to allow further extension as it is based on a modular- structured design.

6 FUTURE WORK

The complete data flow including and surrounding CMSD is large and many items needs to be in place for a complete automatic solution. Data storages need to be in such format which enables CMSD translations. Network and data storage systems need to be accessible online to reach the latest data.

While developing the CMSD interfaces for Enterprise Dynamics and Plant Simulation, a generic development architecture was established. This architecture could be used to develop CMSD interfaces for other simulation packages as well.

Ongoing work will enable data retrieval from many different data sources. An application will provide mapping tools on how to translate the data from these sources to CMSD format. The tool will also include distribution fitting algorithm for stochastic observations of input data. The user of the application configures the mapping once. Next time data is updated, the mapping configuration is automatically reused.

In addition software vendors are welcome to provide CMSD interfacing in their solutions using XML-schema of CMSD.

ACKNOWLEDGMENT AND DISCLAIMER

The funding for this research is granted by VINNOVA (Swedish Agency for Innovation Systems, integrates research and development in technology, transport and working life.) through the MERA-program.

The work described was also funded by the United States Government and is not subject to copyright. No approval or endorsement of any commercial product by the National Institute of Standards and Technology is intended or implied.

REFERENCES

ANSI/ISA 95.00.01-2000, *Enterprise/Control System Integration Part 1: Models and Terminology*, <www.isa.org>

Banks, J., J. C. Hugan, P. Lendermann, C. McLean, E. H. Page, C. D. Pedgen, O. Ulgen, J. R. Wilson. 2003. The future of the simulation industry. In *Proceeding of the 2003 Winter Simulation Conference*, ed. S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, 2033-2043. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

CMSD 2007. *Draft core manufacturing simulation data information model part 1: UML model.* CMSD Product Development Group, Simulation Interoperability Standards Organization. Available via <http://discussions.sisostds.org/default.asp?action=9&boardid=2&read=39532&fid=24> [accessed March 19, 2008]

Banks, J., J. S. Carson, B. L. Nelson, and D. M. Nicol. 2005. *Discrete-event system simulation*. 4th ed. Upper Saddle River, New Jersey: Prentice-Hall, Inc.

Ericsson, U. 2001. *Discrete Event Simulation, The Truth*. Ph.D. thesis, Department of Production Engineering, Chalmers University of Technology, Sweden.

Hopp, W. J., and M. L. Spearman. 1996. *Factory Physics*, Irwin, Chicago, Illinois.

Incontrol 2007. The Enterprise Dynamics simulation software, Incontrol. Available at <http://www.incontrol.nl> [accessed March 19, 2008]

Huang, E., R. Ramamurthy, and L. F. McGinnis. 2007. System and Simulation Modeling Using SysML. In *Proceeding of the 2007 Winter Simulation Conference*, ed. S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 796-803. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Johansson, M., and R. Zachrisson, 2006. *Modeling automotive manufacturing process.* Master's thesis. Department of Product and Production Development, Chalmers University of Technology, Gothenburg.

Law, A. M., and W. D. Kelton. 2000. *Simulation modeling & analysis*. 3rd ed. New York: McGraw-Hill, Inc.

Lee, Y. T., and Y. Luo. 2007. *Machine Shop Information Model Application, Next Step*. Technical report NISTIR 7388, National Institute of Standards and Technology, Gaithersburg, Maryland.

Leong, S, Y. T. Lee, and F. Riddick. 2006. *A Core Manufacturing Simulation Data Information Model for Manufacturing Applications.* Simulation Interoperability Workshop, Simulation Interoperability and Standards Organization, September 10-15, 2006, Orlando.

McLean, C., Y. T. Lee, G. Shao, and F. Riddick. 2005. *Shop Data Model and Interface Specification*. Technical report NISTIR 7198, National Institute of Standards and Technology, Gaithersburg, Maryland.

MSDN 2007a. *Microsoft XML Core Services.* Microsoft Developer Network, Microsoft Corporation. Available via <http://msdn2.microsoft.com/en-us/library/ms763742.aspx> [accessed March 19, 2008]

MSDN 2007b. *Building MSXML Applications.* Microsoft Developer Network, Microsoft Corporation. Available via <http://msdn2.microsoft.com/en-us/library/ms753804(VS.85).aspx> [Accessed March 17, 2008]

SysML 2007. *OMG Systems Modeling Language V1.0*. Object Management Group. Available via http://www.sysml.org/docs/specs/OMGSysML-v1.0-07-09-01.pdf [Accessed March 18, 2008]

Skoogh, A., and B. Johansson, 2007. Time-consumption Analysis of Input Data Activities in Discrete Event Simulation Projects, In *Proceedings of the Swedish Production Symposium 2007*, Gothenburg.

Sly, D., M. Moorthy. 2001. Simulation Data Exchange (SDX) Implementation and Use. In *Proceeding of the 2001 Winter Simulation Conference*, ed. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 1473-1477. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Taylor, S. J. E., S. Strassburger, S. J. Turner, M. Y. H. Low, X. Wang, and J. Ladbrook, 2006. Developing Interoperability Standards for Distributed Simulation and COTS Simulation Packages with CSPI PDG. In *Proceeding of the 2006 Winter Simulation Conference*, ed. L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M Fujimoto, 1101-1110. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Ulgen, O., A. Gunal. 1998. Simulation in the Automobile Industry. In *Handbook of Simulation*. ed. J. Banks, 547-570 City of New York, New York: Wiley-Interscience. Available via <http://www.knovel.com/web/portal/basic_search/display?_EXT_KNOVEL_DISPLAY_bookid=1452> [accessed March 19, 2008]

Umeda, S., A. Jones. 1997. *Simulation in Japan: State-of-the-art update*, Technical Report NISTIR 6040, National Institute of Standards and Technology, Gaithersburg, Maryland

W3C 2007. *Extensible Markup Language*, World Wide Web Consortium. Available via <http://www.w3.org> [accessed March 19, 2008]

AUTHOR BIOGRAPHIES

MARCUS JOHANSSON is from Gothenburg in Sweden where he is employed as a project assistant at the Department of Product and Production Development at Chalmers University of Technology. He is currently a guest researcher in the Manufacturing Simulation and Modeling Group at the National Institute of Standards and Technology where he works with implementation and development of Core Manufacturing Simulation Data. He has a M.Sc. degree in the specialized discipline Automation and Mechatronics Engineering. His e-mail address is <johamarc@gmail.se>.

BJÖRN JOHANSSON is an assistant professor at Product and Production Development, Chalmers University of Technology. His research interest is in the area of discrete event simulation for manufacturing industries. He is interested in the specifics of modular modeling methodology for effective and swift model-building, which is accompanied by software development, user interfaces, and input data architectures. His email address is <bjorn.johansson@chalmers.se>.

SWEE LEONG is a senior manufacturing engineer in the Manufacturing Simulation and Modeling Group at the National Institute of Standards and Technology (NIST) Manufacturing System Integration Division since 1994. Prior to joining NIST, he worked at Ford Motor Company, John Deere, and IBM on different factory automation projects. His research interests are in modeling and simulation activities for the manufacturing industries and engineering tools integration. Currently, Swee manages the Simulation Standards Consortium at NIST. He is Chairman of the Core Manufacturing Simulation Data Product Development Group. He received a Bachelor and Master Degrees in Industrial Engineering from Purdue University in West Lafayette, Indiana. He is a senior member of the Society of Manufacturing Engineers. His e-mail address is <leong@nist.gov>.

Y. TINA LEE is a computer scientist in the Manufacturing Simulation and Modeling Group at NIST. She joined NIST in 1986. Her major responsibility in recent years has been to develop information models to support various manufacturing application areas. Previously she worked at Contel Federal Systems and Sperry Corporation. She received her BS in Mathematics from Providence College and MS in Applied Science from the College of William and Mary. Her e-mail address is <leet@cme.nist.gov>.

FRANK RIDDICK is a computer scientist in the Manufacturing Simulation and Modeling Group in The National Institute of Standards and Technology (NIST) Manufacturing Systems Integration Division. He has participated in research and authored several papers relating to manufacturing simulation integration and product data modeling. He holds a Master's Degree in Mathematics from Purdue. His email address is <riddick@nist.gov>.