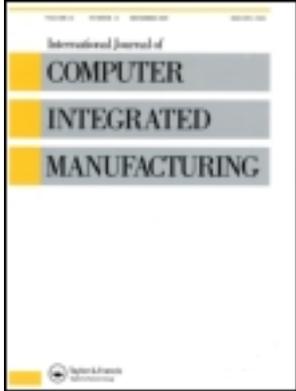


This article was downloaded by: [NIST National Institutes of Standards &]

On: 13 October 2011, At: 14:05

Publisher: Taylor & Francis

Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK



International Journal of Computer Integrated Manufacturing

Publication details, including instructions for authors and subscription information:

<http://www.tandfonline.com/loi/tcim20>

Core Manufacturing Simulation Data - a manufacturing simulation integration standard: overview and case studies

Yung-Tsun Tina Lee ^a, Frank H. Riddick ^a & Björn Johan Ingemar Johansson ^b

^a Manufacturing Systems Integration Division, National Institute of Standards and Technology, Gaithersburg, USA

^b Product and Production Department, Chalmers University of Technology, Gothenburg, Sweden

Available online: 21 Jul 2011

To cite this article: Yung-Tsun Tina Lee, Frank H. Riddick & Björn Johan Ingemar Johansson (2011): Core Manufacturing Simulation Data - a manufacturing simulation integration standard: overview and case studies, International Journal of Computer Integrated Manufacturing, 24:8, 689-709

To link to this article: <http://dx.doi.org/10.1080/0951192X.2011.574154>

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: <http://www.tandfonline.com/page/terms-and-conditions>

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae, and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand, or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.

Core Manufacturing Simulation Data – a manufacturing simulation integration standard: overview and case studies

Yung-Tsun Tina Lee^{a*}, Frank H. Riddick^a and Björn Johan Ingemar Johansson^b

^aManufacturing Systems Integration Division, National Institute of Standards and Technology, Gaithersburg, USA; ^bProduct and Production Department, Chalmers University of Technology, Gothenburg, Sweden

(Received 29 September 2010; final version received 15 March 2011)

Standard representations for information entities common to manufacturing simulation could help reduce the costs associated with simulation model construction and data exchange between simulation and other manufacturing applications. This would make simulation technology more affordable and accessible to a wide range of potential industrial users. To foster the more widespread use of manufacturing simulation technology through the reduction of data interoperability issues, the Core Manufacturing Simulation Data (CMSD) specification was created. CMSD is a standardised, computer-interpretable representation that allows for the efficient exchange of manufacturing shop floor-related data in a manner that it can be used in the creation and execution of manufacturing simulations. The work has been standardised under the auspices of the Simulation Interoperability Standards Organization (SISO). CMSD defines an information model that describes the characteristics of and relationships between the core manufacturing entities that define shop floor operations. This enables greater integration and data exchange possibilities for manufacturing simulations and other manufacturing applications. This article presents an overview of CMSD, its motivation, structure, and content. Descriptions of case studies using CMSD to integrate real world manufacturing applications are also presented.

Keywords: CMSD; information model; interoperability; manufacturing; simulation; standard

1. Introduction

Interoperability between business applications is a significant problem for the today's companies, large and small, that must compete in the current globally scoped marketplace. Interoperability, as defined by the Institute of Electrical and Electronics Engineers, Inc. (IEEE Standards Board 1990), is 'the ability of two or more systems or components to exchange information and to use the information that has been exchanged.' This issue affects businesses of all kinds, with losses due to interoperability problems estimated at about \$15.8B (Gallaher *et al.* 2004). It is a problem that especially affects manufacturing enterprises, where losses have been estimated at about \$1B (Brunnermeier and Martin 1999). Manufacturing enterprises typically employ a large number of software applications that allow them to implement complex business and manufacturing strategies, and to track and analyse the execution of those strategies. Defining, deploying and managing the complex set of applications needed to support modern manufacturing enterprises is extremely difficult.

For manufacturing enterprises, a specific area affected by interoperability issues is the area of

production operations. This area of manufacturing features a variety of complex applications, each with its own notion of what information is important, how that information should be constructed, which applications manage which kind of information, and how much of its information should be shared and exchanged with other applications.

An important category of applications used to understand and manage production is simulations of production operations. Simulations are used to create virtual representations of production operations that can then be manipulated and analysed. These applications provide a means to visualise and evaluate current production operations, diagnose production problems, estimate production capacity, etc. One of the most useful aspects of simulation is that it provides the ability to evaluate changes in the production environment 'virtually' before physically making those changes (Gartner Group 2010). Simulation has repeatedly been identified by the National Research Council (1995, 1998) as a technology that should be deployed by manufacturers to improve their efficiency and profitability.

Unfortunately, manufacturing simulation applications are especially affected by interoperability

*Corresponding author. Email: leet@nist.gov

problems. For simulations, as well as other production operations-related applications, there is generally no agreement on how information important to the area should be defined, stored, managed, or exchanged. This problem makes using simulation technology expensive and time consuming, leading to missed opportunities for manufacturers to use simulations to help them understand, optimise, and streamline their production operations.

To address interoperability issues between simulations and other manufacturing applications, the Core Manufacturing Simulation Data (CMSD) [Simulation Interoperability Standards Organization (SISO) 2010] specification has been developed. It facilitates the definition of manufacturing information related to production operations, in a manner such that the information can be exchanged between simulations and other software applications that are used to manage or analyse production floor operations.

The organisation of this article is as follows. In Section 2, an overview of CMSD is presented. Topics such as the motivation, the multilanguage modelling approach used, the Unified Modeling Language (UML) package structure, and the UML class design approach used for CMSD are discussed. In Section 3, CMSD and two related manufacturing data specifications are compared. In Section 4, three case studies that used CMSD to integrate simulations and other manufacturing applications are described. The article concludes with Section 5, where a summary of the information about CMSD described in this article is presented.

2. CMSD overview

The CMSD specification addresses interoperability between simulations and other manufacturing applications. The model focuses on defining representations for the essential manufacturing entities needed for simulation. It provides neutral structures for the efficient exchange of the manufacturing data needed to support simulation analyses. These neutral structures can be used to support the integration of simulations with other manufacturing applications.

2.1. Modelling objectives

The CMSD information model describes the essential entities in the manufacturing domain and the relationships between those entities needed to create manufacturing-oriented simulations. This model facilitates the exchange of information between manufacturing-oriented simulations and other applications in the manufacturing domain. The primary objective of this standard is to provide a data specification that enables

the efficient exchange of manufacturing life cycle data in a simulation environment. The objective is intended to

- foster the development and use of simulations in manufacturing operations.
- facilitate data exchange between simulation and other manufacturing software applications.
- enable and facilitate better testing and evaluation of manufacturing software.
- increase manufacturing application interoperability.

2.2. Modelling languages

The specification of the CMSD information model is presented using two different methods: (1) the information model defined using UML (Object Management Group, OMG 2010); and (2) the information model described using a schema language for eXtensible Markup Language (XML) documents.

UML, a widely accepted standard of the OMG, is a graphical model representation. It is used for specifying, visualising, constructing and documenting. UML defines a variety of modelling constructs to support several different modelling tasks including several functional requirements specification, activity analysis, class structure definition and component description.

UML diagrams can support object-oriented programming and analysis methodologies (Muller 1997), and its class diagramming feature is frequently used to model both the structure and behaviour of modelled entities. The CMSD specification defines an information model (Lee 1999) (and not an object-oriented model). Therefore, in CMSD only the structure and interrelationships of the modelled entities are defined.

XML, a specification supported by the World Wide Web Consortium (W3C 2010), is a tag-based format for machine interpretable structured documents. An XML schema is a specification of the elements, attributes and structures allowable in a kind of document. It is not only useful for documentation but also useful for validation and processing automation. To facilitate automated validation of CMSD information in XML documents, the format for such documents will be defined using a standard schema language for XML, such as the W3C XML Schema Definition language (Van der List 2002) or the REgular LAnguage for XML Next Generation (RELAX NG) (RELAXNG.ORG 2010).

2.3. UML package design

The CMSD model is designed as a suite of interrelated collections of information modelled as UML classes

contained within UML packages, presented visually as a series of UML class and package diagrams. The primary function of the UML packages in the CMSD model is to partition and group, by major areas of manufacturing, the class definitions that realise related manufacturing concepts. When necessary to improve model clarity, some packages may be further partitioned into sub-packages. The manufacturing concepts within each package are modelled as UML classes and the characteristics associated with each entity are modelled as UML class attributes. Attributes associated with a class may be defined directly in the class or may be defined through an aggregation association with another class defined in the model. Within each package, a series of UML class diagrams is used to present details about the content of each class and its relationship to the other classes in the model. The packages that make up the CMSD model are shown in Figure 1.

The *CMSD* package defines packages that contain definitions for all of the classes and relationships that make up the CMSD information model. Although it does not directly define any class, several nested sub-packages are defined within the scope of the *CMSD* package. It is within these

nested packages that the classes and relationships that define CMSD information are directly defined. Each of the *CMSD* package's sub-packages defines a focused, cohesive set of classes and relationships for a specific subset of CMSD information. The packages directly defined by the *CMSD* package are the *Layout* package, *Part Information* package, *Production Operations* package, *Production Planning* package, *Resource Information* package and *Support* package.

- The *Layout* package defines classes and relationships that facilitate the creation of manufacturing layout information. A manufacturing layout is a specification of the spatially oriented characteristics and interrelationships for the logical and physical entities that are used to carry out production activities.
- The *Part Information* package defines classes and relationships for describing the raw materials, work-in-progress components and finished products that are the inputs to and outputs of manufacturing processes. In addition, information about the component structure of parts and about the kinds and amounts of parts either

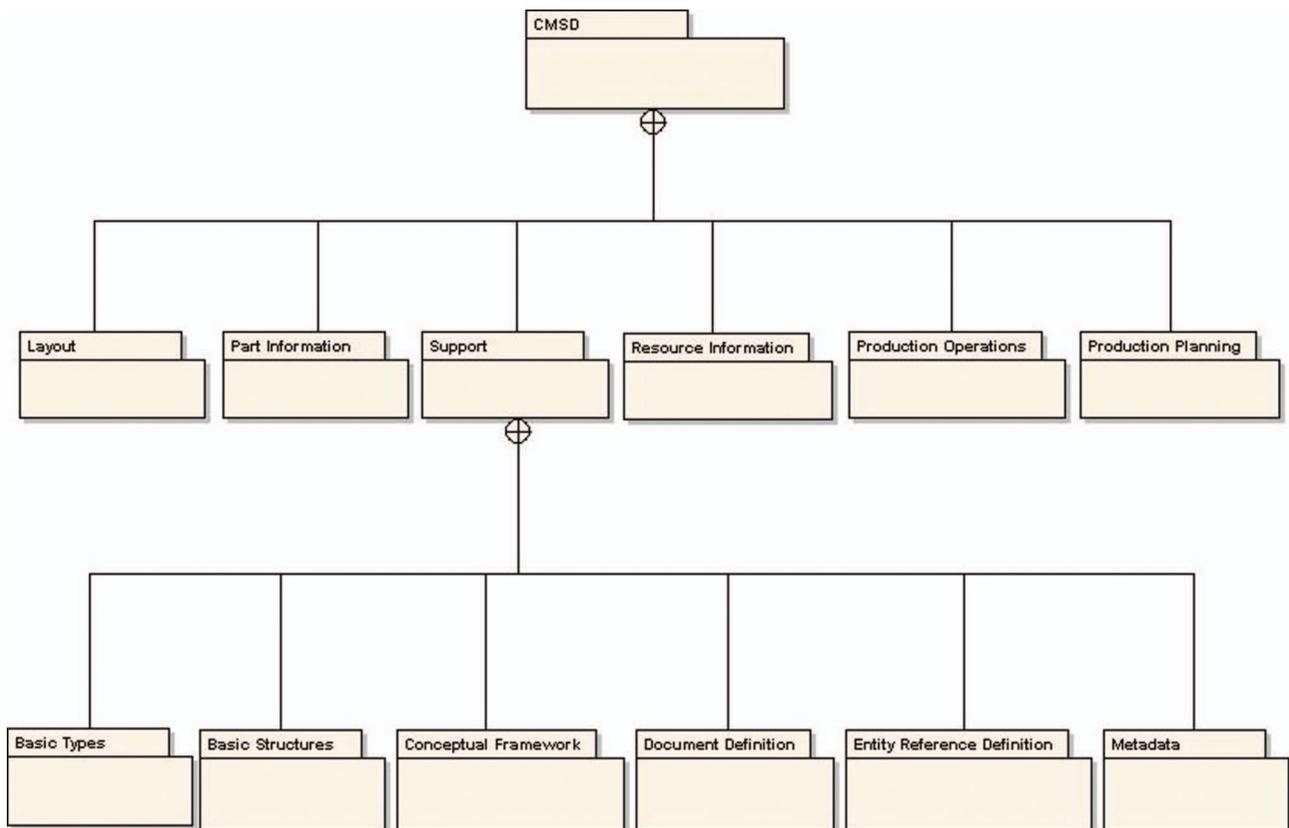


Figure 1. The UML Packages of the CMSD model.

completed or available to be used in production activities can also be defined.

- The *Production Operations* package defines classes and relationships describing customer requests for products, production requests to produce those products, and the effort needed to produce those products.
- The *Production Planning* package contains classes and relationships that can be used to create plans describing the dates and hours of operation for production resources and plans describing the sequence of processing steps necessary to manufacture products using the available production resources.
- The *Resource Information* package contains classes for creating definitions of the characteristics and capabilities of the equipment and employees used in the manufacturing process, the skills associated with employees, and setup information required for the making efficient use of non-employee resources.
- The *Support* package defines packages that contain definitions for simple types and other basic structures that are used in other CMSD packages to define more complex structures.

2.4. CMSD UML class example

To illustrate how classes are defined within the packages of the CMSD specification, the *Resource* class and the *ResourceClass* class are depicted in Figure 2 and described in this section. The *Resource* class defines information about a manufacturing resource, which is a piece of equipment, an employee, or a collection of pieces of equipment and/or employees that are used in the manufacturing process. All resources have a *ResourceType* attribute that describes in general the kind of manufacturing asset or assets the

resource represents. In CMSD, the *ResourceType* attribute may take on one of the values *machine, station, employee, conveyor, fixture, tool* or *other*. Other information that can be defined for a resource includes the current operational state of the resource, the gross dimensions of the resource, the ‘class’ to which a resource belongs and information about the group members for resources that are resource groups.

The *ResourceClass* class provides a means to create a classification scheme for resources based on descriptions of the characteristics that those resources possess. Like the *Resource* class, the *ResourceClass* class has a *ResourceType* attribute indicating the kind of manufacturing resource being described. The distinguishing characteristics for the class of resources being described by a *ResourceClass* are defined using one or more *Property* attributes.

3. Related manufacturing and simulation standards

The scope of the CMSD specification overlaps with some existing manufacturing data specifications, especially in the area of manufacturing operations management. As a part of the effort to develop CMSD, several related manufacturing data specifications were examined. In this section, a brief overview of CMSD, ISA-95 (ISA-95.COM 2010) and the Open Applications Group Integration Specification (OAGIS) (OAGi 2010a) are presented. A comparative analysis of the specifications and a discussion of the need for CMSD, given there are existing standards in this area, are also presented.

3.1. Specifications overview

3.1.1. Core manufacturing simulation data

CMSD, developed by the SISO, is an international standard supporting the integration of simulations

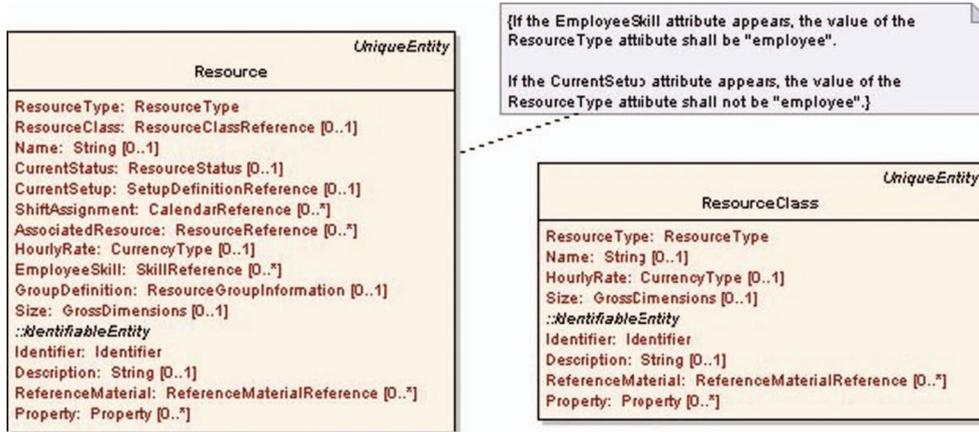


Figure 2. The *Resource* class and *ResourceClass* class.

with other manufacturing software applications. The key features of CMSD are as follows: (1) it specifically facilitates the integration of simulation applications by providing a means to define aspects of manufacturing entities that are governed by stochastic processes, in such a way that the information can be exchanged and shared and (2) CMSD information elements may be extended with additional properties and other information. This subsection provides a description of some of the major manufacturing entities included in the CMSD.

- *Resource* information describes the people and equipment that perform manufacturing activities. Resources in the CMSD are used to represent stations, machines, cranes, employees, tools, fixtures, carriers, transporters, conveyors, power and free conveyors, and paths.
- *Order* information provides a means to specify a request for products or services originating from a person or organisation external to the manufacturing enterprise.
- *Calendar* information provides a means to specify a long-term focused collection of shift and holiday information that, taken together, specify the time periods during which production is and is not expected to take place.
- *Skill definition* information describes the skills that an employee resource may possess and the levels of proficiency associated with those skills.
- *Setup definition* information describes how resources may be configured to perform a task, how long it takes to configure the resource, and how long it takes to change from one configuration to another.
- *Part* information provides a means to specify the characteristics of the materials and subcomponents that are used in some stage of production, or an end product that is the final objective of production.
- *Bill-of-materials* information provides a means to specify the subcomponent parts and the quantities of those parts that are needed to make an end product.
- *Process plan* information provides a means to specify the set of production activities needed to transform materials and subcomponents into finished products.
- *Maintenance plan* information provides a means to specify a collection of maintenance processes that provide the necessary instructions for maintaining a machine or other maintained manufacturing resource.
- *Job* information provides a means to specify a request for production-related activities to take

place, originating from a person or organisation internal to the manufacturing enterprise.

- *Schedule* information provides a means to specify a plan containing a time-ordered collection of production activities, and/or the results obtained by carrying out such a plan.
- *Distribution* information provides a means to specify statistical distributions that can be used to indicate process variability. A statistical distribution is a mathematical function where (1) the range of possible values of the function is known, and (2) the probability that a random input to the domain of the function will produce an output value in a subset of the range that is known.
- *Layout* information provides a means to specify spatially relevant characteristics of, and relationships between, the manufacturing resources that are a part of a manufacturing facility.

3.1.2. ISA-95

ISA-95, developed by the Instrumentation, Systems, and Automation Society (ISA), is the international standard for the integration of enterprise and control systems. The Enterprise/Control Integration Committee (ISA-SP95) of ISA has developed and is continuing to work on a multipart series of standards that define the interfaces between enterprise activities and control activities, based upon the Purdue Reference Model for Computer Integrated Manufacturing (Williams 1992). The goal is to reduce the risk, cost and errors associated with implementing these interfaces. The standard can be used for several purposes, for example, as a guide for the definition of user requirements, for the selection of Manufacturing Execution System (MES) suppliers and as a basis for the development of MES systems and databases.

ISA-95 consists of models and terminology. The ISA-95 information models are defined using UML diagrams, and these models can be used as the basis for the development of standard interfaces between Enterprise Resource Planning and MES systems. There are five parts of the ISA-95 standard.

- *Part 1: Models and terminology* consists of standard terminology and objective models that can be used to decide which information should be exchanged.
- *Part 2: Object model attributes* consist of attributes for every object that is defined in Part 1.
- *Part 3: Models of manufacturing operations management* defines production activities and information flows. It focuses on the

functions and activities at the Production/MES layer level of the Purdue Enterprise Reference Architecture.

- *Part 4: Object models and attributes for manufacturing operations management* defines object models that determine which information is exchanged between MES activities. The models and attributes are the basis for the design and implementation of interface standards.
- *Part 5: Business to manufacturing transactions* defines business-to-manufacturing transactions and manufacturing-to-business transactions that may be used in relation to the object models defined in Parts 1 and 2.

3.1.3. Open Applications Group integration specification

The OAGIS standard (OAGi 2010a), developed by the Open Applications Group Inc. (OAGi), is an effort to provide a canonical business language for information integration. The OAGIS framework supports service-oriented architecture definition, web services and is compatible with the Electronic Business using XML (ebXML) standard. OAGIS provides a means to define business messages in the form of Business Object Documents (BODs) and example business scenarios that provide example usages of the BODs. BODs are the business messages and information that are exchanged between software applications, between companies, across supply chains and between supply chains. In OAGIS, information about a business object, referred to as a 'Noun', is defined separately from information about actions that may be applied to a business object. The actions are referred to as 'Verbs.' A BOD represents a merging of noun and verb information to define a common business transaction. *Shipment*, *PurchaseOrder*, *Quote*, and *Requisition* are examples of Nouns. *Acknowledge*, *Cancel*, *Get*, *Show*, *Sync*, and *Update* are examples of Verbs. These Nouns and Verbs facilitate the definition of BODs such as *AcknowledgeShipment*, *ShowPurchaseOrder* and *GetQuote*. As different industries have different needs, OAGIS must provide a means for information that is relevant in different industry vertical markets to be defined. For this reason, BODs have been designed to be extensible, while providing a common architecture and content for integration.

In OAGIS, business scenarios identify the business applications being integrated and the BODs that are used. OAGIS provides example scenarios that can be used as a starting point for integration. By identifying a scenario that most closely matches the user's needs, it is possible to identify the messages needed to achieve the needs.

OAGIS covers data exchange requirements for business systems and applications, including manufacturing and operations management systems. The current release, OAGIS 9.4 (OAGi 2010b), includes 80 nouns, 498 business messages and 61 business scenarios that can be used to integrate business applications. OAGIS has expanded its scope to include functionality to allow the footprint of the specification directly support Process Manufacturing. Additional Nouns, such as *ProductionPerformance* and *ProductionSchedule*, were defined based on the ISA-95 specification as a part of an ongoing convergence effort to normalise manufacturing interoperability standards for process, discrete and mixed-mode manufacturing using the ISA-95 and OAGi models.

3.2. Specifications comparison

This subsection compares the CMSD specification with the ISA-95 and OAGIS specifications. The viewpoint of the comparison is how manufacturing simulation data representation can best be supported. Table 1 summarises the specifications' background information including responsible organisation, standardisation level, availability, modelling language or tool used and file exchange format. Table 2 compares the specifications' content information including scope, domain coverage (that is major data categories defined by the specification), application supported, attribute definition accuracy and remarks.

The key findings of this comparison study are:

- With respect to specification availability, CMSD is free to the general public. The ISA-95 specification is free to ISA members, but it requires a royalty payment from non-members. OAGIS offers free specification downloads after registration, which is also free.
- CMSD is designed to support job shop manufacturing, but does not directly support flow shop manufacturing. ISA-95 and OAGIS focus on flow shop or continuous process manufacturing, but they claim to support both modes.
- ISA-95 defines a framework, including a common terminology, abstract models and transactions, for enterprise-control system integration. The emphasis is on good practices for integrating manufacturing systems with other enterprise systems. The data models provided are in UML.
- ISA-95 does not provide a machine interpretable information exchange form. An XML representation of ISA-95 was created named the Business to Manufacturing Markup Language (B2MML) (WBF 2010). B2MML, known as IEC/ISO 62264, was developed by the World Batch Forum.

Table 1. Specifications comparison – background.

Specification name	Responsible organisation	Standardisation	Availability	Modeling language or tool	File exchange format
CMSD	SISO	SISO-STD-008–2010 (UML Model)	Free to general public	UML XML schema	XML
ISA-95	ISA	ANSI/ISA ISO/IEC	Free to ISA members Requires a royalty payment from non-member	UML	B2MML
OAGIS	OAGi	OAGi standard Compliant with ISO 11179, ISO 15000–5, and UN/ CEFACT TBG17 Will compliant with UN/CEFACT: NDR 3.0 and CCTS 3.0	Free download with registration	XML schema	XML ebXML

Table 2. Specifications comparison – content.

Specification name	Scope	Domain coverage	Application supported	Attribute definition accuracy	Remarks
CMSD	Simulation of manufacturing operations in job shop environment	Core set manufacturing data including: parts and inventory, production planning, Resources, production operations and 2-D plant layout	Discrete event simulation of manufacturing operations and 2D layout	Major attributes are clearly defined	No specific support for message exchange
ISA-95	Business to Manufacturing (B2M) integration framework in process manufacturing environment	Resource Product Production capability Production schedule Production performance	Logistics operations support: ERP, MES, PLC, MRP	Attribute's data type is not defined Only certain key attributes are named and defined Allow users to define most of attributes	B2MML is an XML implementation of ISA-95 Interpreting the content of an exchange requires pre-agreement on the semantic meaning of each attribute in the exchange
OAGIS	Canonical business language for Application to Application (A2A), Business to Business (B2B), and Business to Consumer (B2C) for Inter-enterprises	Business scenario Business message container for supporting: enterprise, commerce and manufacturing	SAP, ERP, CRM, MRP Supply chain transaction	Allow users to specify any message content	Interpreting the content of an exchange requires pre-agreement on the semantic meaning of each attribute in the exchange Does not contain the in-depth semantic information

- OAGIS defines business messages and identifies business process scenarios that allow business applications to communicate. The OAGIS framework includes enterprise, commerce and

manufacturing functionality with emphasis on business process interoperability.

- OAGIS, ISA-95 and B2MML often allow users to define attributes of the objects or entities as

needed. This flexible approach makes it hard to develop software tools to interpret the data. When exchanging data, these attributes cannot directly be interpreted without providing pre-negotiated definitions.

- CMSD provides the ability to define characteristics of manufacturing entities that are governed by stochastic processes. It provides this feature by allowing statistical distribution information to be defined and associated with different properties of manufacturing entities. This feature enables distribution information to be used in and exchanged between discrete event and other types of simulations of manufacturing operations.

Both ISA-95 and OAGIS specifications provide a structured way for creating exchangeable content for the manufacturing industry; however, they are not neutral data formats for storing the manufacturing data needed by simulation models. They define many different classes of manufacturing-related data, but many of the key attributes of these classes are not clearly defined. Place holders reserve space for users to specify the custom data elements. This flexible approach makes it hard to develop software tools to interpret the data. From this analysis, CMSD's unique role is identified. CMSD facilitates the exchange of manufacturing life cycle data in a simulation environment, whereas ISA-95 and OAGIS are not focused on this problem.

4. Case studies

This section presents three case studies that were undertaken as a part of the CMSD standardisation effort.

4.1. Car manufacturer

The industrial needs of car manufacturers are used as the starting point for this case study. Two important issues must be resolved to be able to perform a simulation analysis of an entire car manufacturer plant:

- Modelling an entire plant takes too long and requires significant simulation expertise.
- Input data management consumes too much time and requires a lot of manual work.

These problems indicate a need for a technology solution that enables the rapid creation of easily understood abstract models that can be used by car manufacturing programme managers. The solution must enable managers to quickly represent and

perform analysis of their plant's capabilities because they are usually the ones that make important decisions in early stages of programme development. There is also a need for standardising and automating the collection and preparation of data for input into relevant analysis tools. Typically, a vast amount of raw production data is available to car manufacturers, but these data are usually unsuitable for direct input into any analysis tool. Furthermore, employing abstract models for analysis generally requires some manipulation of input data to fit the chosen level of abstraction. Car manufacturers have a clear need for this work to be standardised, with clear definitions for the representation and exchange of the shop floor data needed to carry out simulation at all abstraction levels.

4.1.1. Goals of the study

The main goal of the study was to evaluate CMSD's feasibility for robustly representing the data needed to support the abstractions essential to represent a car manufacturing plant's operations. A secondary goal was to develop reusable objects and generic solutions to enable simulation engineers at the car manufacturer to analyse the plant's operations. To carry out this study, a detailed model of a car assembly plant was developed using Enterprise Dynamics (ED) simulation software. The extensible and object-oriented structure of ED made it possible to create tailored modelling objects for common resources used by the car manufacturer. The ability to create custom objects and reusable logic functions sped up the modelling of some of the plant's functions, such as the paint shop, as well as supported the modelling of other factories within the same company. An additional goal was to develop the application in such a way that it could be used by factory management staff for planning on a weekly basis. They were interested in a tool that could help them forecast necessary shop floor staffing. Because of this goal, the user interface to the application was designed using Microsoft Excel, a technology that was familiar to the intended user. To enable standardised input data management, CMSD was used as a basis for representing data related to resources and work processes. This presentation of the case study focuses on how CMSD was involved in the application's implementation. Information about other aspects of the case study can be found in the studies by Johansson and Zachrisson (2006) and Kibira and McLean (2007).

4.1.2. Method and realisation

To accomplish the study, the paint shop manufacturing process was analysed in detail, a simulation model

was built based on this analysis, and a user interface based on Microsoft Excel was created. Input for the modelling effort was solicited from personnel involved in the car manufacturer operations. This approach was taken to provide a high level of realism and validation for the model and to foster buy-in for the personnel who might use the model. The methodology used during the model-building phase was derived from the one developed by Banks *et al.* (2005). The main difference is that Banks proposes complete conceptualisation before simulation modelling, whereas Johansson and Zachrisson (2006) use iterations of gradual conceptualisation and simulation modelling by dividing the whole factory in smaller areas modelled one at a time.

Since one of the goals of the study was to create a useful tool for factory managers, an Excel user interface was created to facilitate easy data entry. In the interface, it is possible to enter values for cycle time, disturbance data, material handling equipment speed and resource availability. The user interface provides functions for data collection and configuration, and for initiating the creation of CMSD data files.

Once the simulation model of the plant's operations was created, XML instance documents were created containing information that described the resources and work processes defined in the simulation model. The format of these documents was based on a mapping of CMSD to XML. A script to interpret and transfer the XML data into the simulation model was written based on these documents.

4.1.3. Overview of the modelled manufacturing process

The factory that was modelled contains several processes associated with the painting of a car body: sealing, washing, painting, hardening and controlling. A simple flow diagram of the factory can be seen in Figure 3. A more detailed flow diagram and a conceptual model are included in the study by Johansson and Zachrisson (2006).

The resources that carry out the production process can be characterised in several different ways. Some processes are carried out on resources that can be described as *continuously driven lines*. The processes carried out on these resources are the hardening, washing and painting processes. The resources that carry out the other work in the factory can be divided in two resource types: *manual workstation* and *automatic workstation*. For both types of workstation, the total cycle times can be determined empirically by analysing their process logs, but cycle times for individual workstation tasks cannot. Therefore, workstations provide the lowest level of operation that can

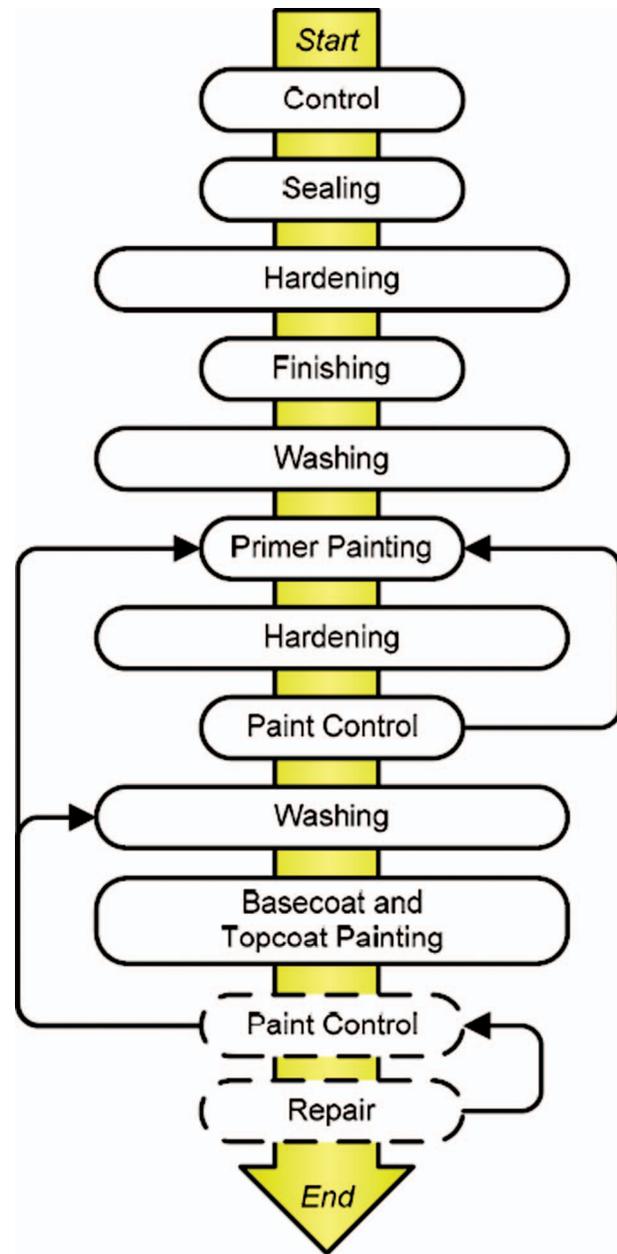


Figure 3. A simple flow chart of the factory.

Note: Steps with dashed lines are present in the factory but not included in this case study. Steps with wide symbols, such as *Hardening* and *Washing*, represent continuously driven lines.

be modelled. Historical data in the form of process logs were available in a shop floor database. Disturbance data for workstations and other factory resources were retrieved from this source.

In this factory, car bodies are carried on skids as they move from resource to resource. The skids are moved on special skid-conveyors as depicted in Figure 4. The number of conveyor resources is extensive compared with the work-related resources. There are also several elevators in the factory. In contrast to the

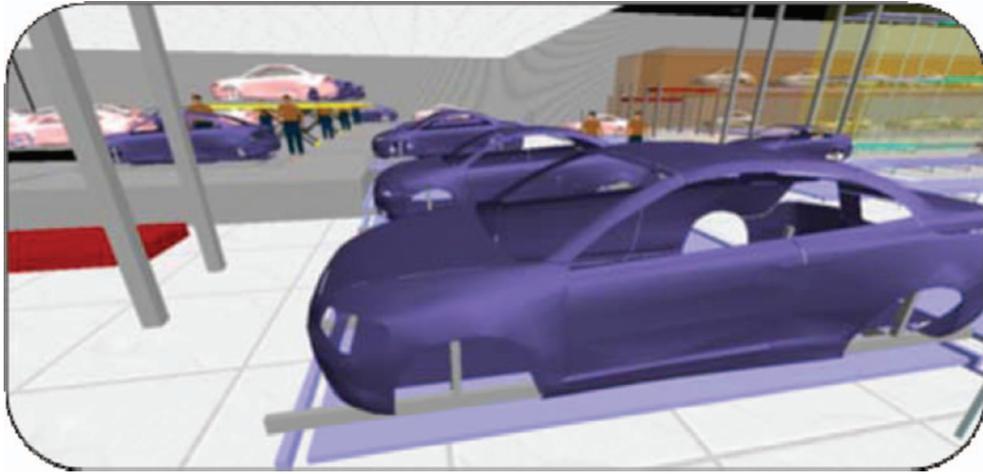


Figure 4. Car bodies on skids, moving on skid-conveyors after primer-paint control stations.

situation with disturbance data for work resources, historical disturbance data for conveyors and elevators were scarce. Therefore, conveyors and elevators are modelled as reliable resources, and the ability to set breakdown data for them in the model is not provided.

4.1.4. User interface overview

The user interface was created as a series of Microsoft Excel spreadsheets that allow the entry of resource definitions, process definitions and disturbance data. It provides a flexible means to enter information about existing or prospective production scenarios in a form familiar to the manufacturing engineers that would be using it. Separate worksheets are provided for defining workstation, conveyor and elevator data. For conveyors, only the speed can be set. For elevators, two different speeds can be set, indicating the speed for an empty elevator and the speed for a loaded elevator. For workstations, several worksheets are used to allow the entry of staffing, cycle time and disturbance data. In the staffing worksheet, the number of employees assigned to each workstation on a certain shift can be set. In the cycle time sheets, the distribution type and its parameters can be set for each combination of workstation, product and staffing (workstation staffing is only a parameter for manual stations). In the disturbance sheet, the user can select distribution type and parameters for Mean Time To Failure (MTTF) and Mean Down Time (MDT).

4.1.5. Approach to resolving discrepancies in data representations

While modelling the resource and process data to be used in the study, several conflicts were discovered between the way some data needed to be modelled in

CMSD and how the same data needed to be modelled in the ED simulation. In general, the conflicts were resolved by creating tentative extensions to CMSD specification rather than by changing the simulation model. This approach was undertaken because (1) recreating and reverifying the finished simulation model would be extremely burdensome, (2) there was limited time available to finish the study and (3) creating tentative extensions to CMSD was feasible given the time available. One of the interests of the CMSD development team was to use CMSD in integration projects and case studies such as this to validate the structures already present in CMSD and to gather requirements for future enhancements. With this in mind, temporary extensions to CMSD were created solely to carry out the integration efforts associated with this project.

Since the conclusion of this study, modifications to CMSD that resolve issues uncovered by this study have been proposed and accepted as a permanent part of CMSD. Therefore, the XML examples presented in the following sections have been updated to indicate how the data would be represented using the current version of the standard.

4.1.6. Mapping input data to XML elements

Since the user interface contains all necessary resource data, the CMSD-based XML document was created based on that data. Working with real world data from companies such as the car manufacturer provides a means to evaluate the CMSD specification's usefulness in providing a framework for fostering the exchange of data between manufacturing applications. An XML instance document was created containing the data that were modified through the user interface and that were to be read into the ED simulation. To automate this process, an Excel macro was written using

Microsoft Visual Basic for Applications (VBA) and the Microsoft XML Core Services Library (MSXML) (MSDN 2010). The complete data flow is visualised in Figure 5. ED provides a basic set of functions to read and write XML documents, and these were used to import the data into the ED simulation. The ED functions provide a means to access specific XML elements in a document directly by node name, and this capability was used to create a translation script in ED that worked irrespective of the order of the data in the XML document.

One kind of information that needed to be imported into the simulation involved the operational characteristics of the conveyors and elevators that were a part of the production system. Initially, CMSD did not directly support material handling equipment such as conveyors and elevators but was extended to support these kinds of resources as a result of this case study. Figures 6 and 7 are examples of how resource information for a conveyor and an elevator are modelled in the study. CMSD Property elements are used to model the ‘speed’ characteristic of the conveyor and the ‘unloadedSpeed’ and ‘loadedSpeed’ characteristic of the elevator.

In the model, each resource must be associated with data for staffing, cycle time and disturbance. These data are entered in different worksheets in the user interface and translated into different CMSD elements in the XML file. Examples of how this is accomplished are presented below.

For each manual workstation, different numbers of workers may be assigned to work at that workstation depending on the shift. That information can be entered in the user interface as shown in Figure 8.

Initially, CMSD did not provide a means to associate a specific number of employees for each resource for each shift. CMSD was extended so that such information could be entered as a part of the *ShiftSchedule* element. An example of the XML to represent the number of employees assigned to work on each resource for specific shift is presented in Figure 9.

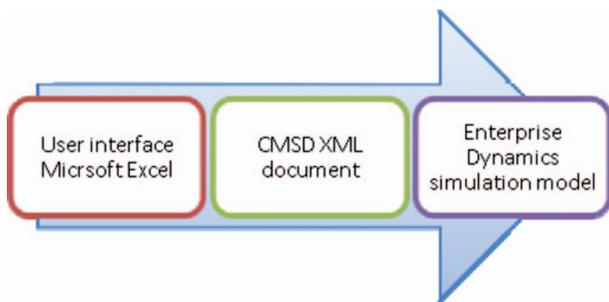


Figure 5. The simulation input data flow.

The next kind of information that needed to be specified for the study was the cycle time for each resource. Normally, in CMSD, the cycle time for a given process to be executed on a given resource is specified in a *ProcessPlan* element. This approach allows a given process, including its cycle time and other processing characteristics, to be defined independently and then to be associated with any resource that can perform the process. This approach better supports the definition for information relating job shop-oriented manufacturing. Many simulation systems, including ED, require that the cycle time for a process be directly specified on the resource that will carry out the process. This approach is most suitable for flow shop manufacturing.

In CMSD, custom *Property* elements can be added to *Resource* elements. Therefore, for this study, cycle time information for each process was encoded as a *Property* element on each *Resource* element. This

```
<Resource>
  <Identifier>Part_13:C_341.05</Identifier>
  <Description>ALONG</Description>
  <ResourceType>conveyor</ResourceType>
  <Property>
    <Name>speed</Name>
    <Unit> meterPerSecond</Unit>
    <Value>0.33</Value>
  </Property>
</Resource>
```

Figure 6. XML for a conveyor resource.

```
<Resource>
  <Identifier>E_52-11-341.30</Identifier>
  <ResourceType>other</ResourceType>
  <Description>elevator</Description>
  <Property>
    <Name> unloadedSpeed </Name>
    <Unit> meterPerSecond</Unit>
    <Value>0.06</Value>
  </Property>
  <Property>
    <Name> loadedSpeed </Name>
    <Unit> meterPerSecond</Unit>
    <Value>0.06</Value>
  </Property>
</Resource>
```

Figure 7. XML for an elevator resource.

Workstation	A-shift	B-shift	C-shift
WM_11-140	3	3	2
WM_11-150	0	0	0
WM_11-160	2	2	2

Figure 8. The simulation input data of *Resource Staffing Levels* in Microsoft Excel.

approach enables the XML information to be more easily read in by the simulation and allows CMSD to be used as the transport format for the data without the need for new CMSD elements. The user interface for entering the cycle time data and an example of the associated XML are presented below.

In the user interface, two spreadsheets were used to specify cycle time data. In this example, all of the cycle time distributions are normal distributions for which mean and standard deviation information must be specified. In one spreadsheet of the user interface, the cycle time mean for each resource for each operation can be specified. In an associated spreadsheet, the standard deviation for each resource for each operation is specified. Also, since some distributions take more than two parameters (such as Triangular distributions that have minimum, maximum and most likely parameters), an additional spreadsheet to hold cycle time parameter data is supported in the user interface.

Figure 10 shows how the cycle time data for the first workstation, as shown in Figures 11 and 12, would be defined in XML. For this workstation, the cycle time for a given product may differ depending on the number of assigned operators. Because there are three different parts (Body1, Body2 and Body3) and one to two operators working at the station, there are six different cycle times that need to be specified. For brevity, the example below only shows the cycle time for workstation *WM_GRIND1* when one operator is working on part Body1.

In addition to the cycle time and speed information, resources in the study need to have disturbance data specified. In a similar manner to how the speed information was specified, MTTF and MDT information can be added to CMSD Resource elements in the form of Property elements. In the user interface, spreadsheets were defined so that disturbance distribution information could be entered for each resource in much the same manner as was done for the cycle time information. Figure 13 is an example of how that information would appear in an XML document.

```

<ShiftSchedule>
  <Identifier>ShiftStaffing:AShift</Identifier>
  <Description>Number of employees assigned to each manual station</Description>
  <Property>
    <Name>StaffingLevel:station:WM_11-140</Name>
    <Value>3</Value>
  </Property>
  <Property>
    <Name>StaffingLevel:station:WM_11-150</Name>
    <Value>0</Value>
  </Property>
  <Property>
    <Name>StaffingLevel:station:WM_11-160</Name>
    <Value>2</Value>
  </Property>
</ShiftSchedule>

```

Figure 9. The simulation input data of *Resource Staffing Levels* in XML.

4.1.7. Summary of study results

The implemented system involving an Excel-based user interface, CMSD export and import functions, and production system simulation was suitable for allowing manufacturing engineers to evaluate different production optimisation strategies. The integrated system facilitated experimentation involving multiple simulation runs using different input data without requiring simulation experts to modify the simulation. With respect to CMSD, requirements for additional flexibility in representing Resource characteristics were identified, and the specification was modified to support these requirements.

4.2. Truck manufacturer 1

This section presents an overview of the CMSD interfaces developed using ED and Plant Simulation as well as how they are used in a real world case study for a truck manufacturing plant.

4.2.1. CMSD interfaces – how to use it

The CMSD interfaces were developed as reusable objects that could be used in multiple simulations. The user loads the interface into the simulation environment using standard routines. Figure 14 shows the interfaces loaded into object libraries in ED and Plant Simulation.

To use the CMSD interface in the model, the user simply drags the CMSD interface icon from the objects library and drops it into the model layout. Figures 15 and 16 show simple example models in both simulation packages where the CMSD interface is used.

```

<Resource>
  <Identifier> WM_GRIND1</Identifier>
  <Description>Finish_grinding</Description>
  <ResourceType>station</ResourceType>
  <Property>
    <Name>cycletime:Body1:1</Name>
    <Unit> second</Unit>
    <Distribution>
      <Name>normal</Name>
      <DistributionParameter>
        <Name>mean</Name>
        <Value>412.8</Value>
      </DistributionParameter>
      <DistributionParameter>
        <Name>standardDeviation</Name>
        <Value>17</Value>
      </DistributionParameter>
    </Distribution>
  </Property>
  ...
</Resource>

```

Figure 10. The simulation input data of *Resource Cycle Time* in XML.

Workstation Name	Distri-bution	1 Op.			2 Op.		
		Body 1	Body2	Body 3	Body 1	Body2	Body 3
WM GRIND1	Normal	412,80	412,40	436,00	206,40	206,20	218,00
WM GRIND2	Normal	412,80	412,40	436,00	206,40	206,20	218,00
WM GRIND3	Normal	412,80	412,40	436,00	206,40	206,20	218,00

Figure 11. The simulation input data of *Resource Cycle Time Mean*.

Workstation Name	Distri-bution	1 Op.			2 Op.		
		Body 1	Body2	Body 3	Body 1	Body2	Body 3
WM GRIND1	Normal	17,00	25,00	15,00	11,00	15,00	10,00
WM GRIND2	Normal	17,00	25,00	15,00	11,00	15,00	10,00
WM GRIND3	Normal	17,00	25,00	15,00	11,00	15,00	10,00

Figure 12. The simulation input data of *Resource Cycle Time Standard Deviation*.

Once added to a model, the CMSD interface can be right clicked to pop up a context menu. From the context menu, a user has several options:

- Load data from a CMSD XML file. An 'open file' dialog appears to allow a file to be chosen to load.
- Reload data from an already chosen file.
- Examine and edit CMSD information using a table representation of the data.

The CMSD interface for Plant Simulation provides more functionality than the interface in ED. In addition to reading and viewing CMSD XML files, the Plant Simulation implementation gives the user the following options:

- Turn on safe execution mode. This option allows CMSD files containing format errors to be manipulated without hanging the simulation.
- Turn on debugging mode. This option generates log files based on the results of data retrievals and messages created because of safe execution mode activation.
- Turn on a work-logging function. This option modifies the internal representation of the CMSD information read from a CMSD file with updates generated as a part of a simulation execution.
- Write CMSD XML files. This option initiates a 'save file' dialog to allow the internal CMSD information to be saved to disk. Figure 17 shows the context menu for the Plant Simulation CMSD interface.

```

<Resource>
  <Identifier>WM_11-160</Identifier>
  <Description>Manual_sealing</Description>
  <ResourceType>station</ResourceType>
  <Property>
    <Name>MTTF</Name>
    <Unit> second</Unit>
    <Distribution>
      <Name>exponential</Name>
      <DistributionParameter>
        <Name>mean</Name>
        <Value>20688</Value>
      </DistributionParameter>
    </Distribution>
  </Property>
  <Property>
    <Name>MDT</Name>
    <Unit> second</Unit>
    <Distribution>
      <Name>lognormal</Name>
      <DistributionParameter>
        <Name>mean</Name>
        <Value>1130</Value>
      </DistributionParameter>
      <DistributionParameter>
        <Name>standardDeviation</Name>
        <Value>2024</Value>
      </DistributionParameter>
    </Distribution>
  </Property>
  ...
</Resource>

```

Figure 13. The simulation input data of *Resource Speed Disturbance* in XML.

4.2.2. CMSD interface – user functions

By adding the CMSD interface to the simulation model, a library of user functions is enabled. The user functions can be used in other simulations to easily retrieve CMSD data and add data to CMSD documents.

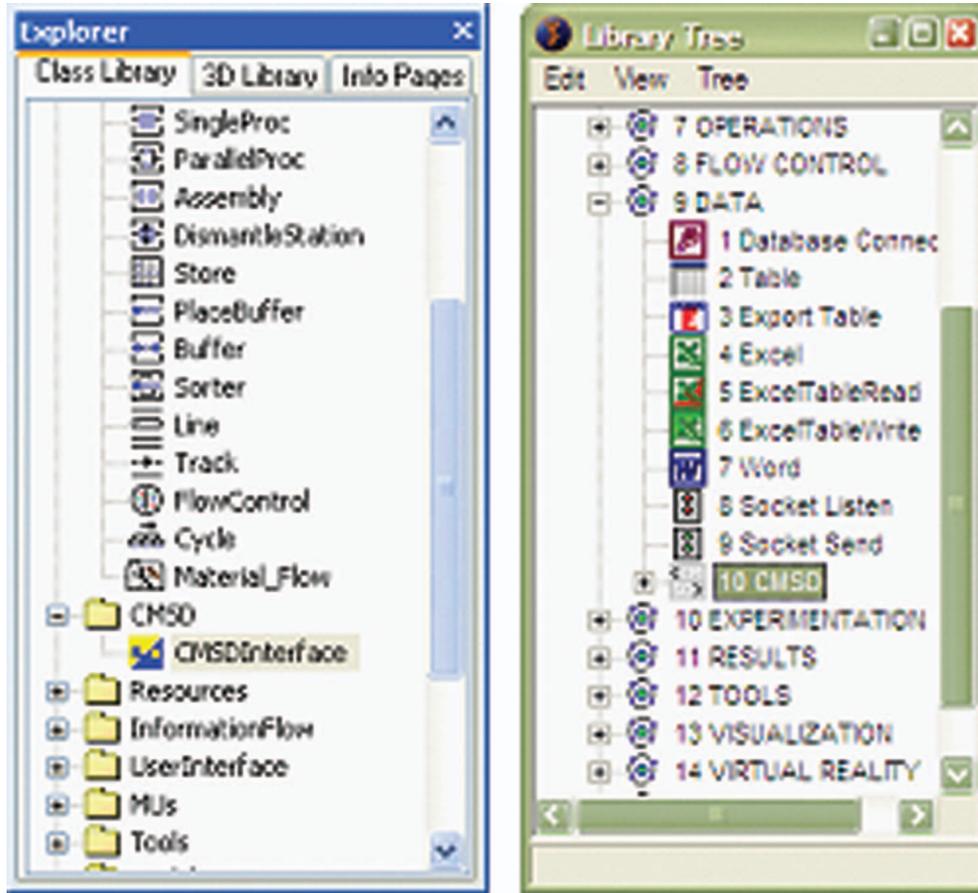


Figure 14. CMSD interfaces in the class library of plant simulation (left) and in the class library of enterprise dynamics (right).

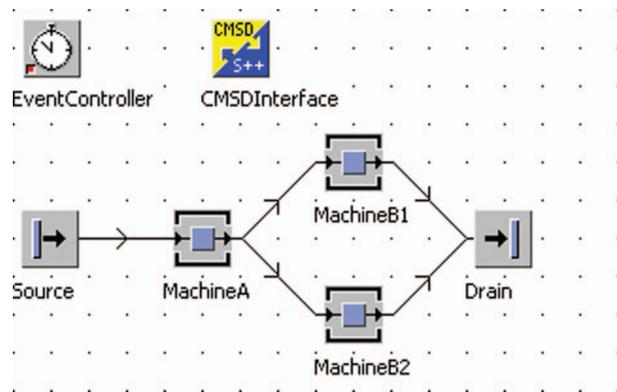


Figure 15. Plant simulation example model.

Depending on the simulation package being used in the implementation, user functions are named differently. The intention is to use the same function naming approach as used in the simulation package to make the user feel familiar with the function names.

Several of the user functions start with the prefix 'Get'. These functions return specific data.

- *GetProcessingTime* returns an operation time based on the machine and product involved.
- *GetFailureInterval* returns a Mean Time Between Failure (MTBF) duration for a resource.
- *GetFailureDuration* returns a Mean Time To Repair (MTTR) duration for a resource.

Two user functions start with the prefix 'Set'. These functions change specific object parameters and are intended to be used in the initialisation 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.

There are also two user functions that start with the prefix 'Add'. These functions help add data to CMSD. Using Add_ functions requires knowledge of CMSD and programming skills in Plant Simulation.

- *Add Empty Instance* creates a new data structure according to any of hundreds of data structures defined in the CMSD information model. To support *AddEmptyInstance*, templates of all CMSD structures are included in the CMSD interface.
- *AddValue* is used to set specific attributes on those data structures.

For all 'Get' functions, 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 sampled from that distribution function will be randomly generated each time the function is called.

There is also a function called *ComputeDuration*. It handles unit conversion and distribution computations for all user functions requiring such functionality. In the CMSD interface implementation, the 'Get' functions locate the data in the CMSD structure, whereas

ComputeDuration computes the data. Usually, this function is not directly called by the end user.

CMSD interface developers can use the *Add_* functions to create their own functions. To demonstrate this, a work-logging function was created. The work-logging function creates and adds Job data to the current internal CMSD data store based on generated products in the simulation model. For example, Job data might include start time, stop time and duration of planned and actual work efforts generated by the execution of the associated simulation model.

4.2.3. CMSD interface – real world test case

To test the CMSD interfaces, an automotive engine assembly process is modelled in both ED and Plant Simulation. The engine line assembly process includes two parallel lines with nine workstations each. Figure 18 shows an outline of the process. Truck engines arrive at workstation 1. The gearbox, clutch, servos and turbo components are mounted at other workstations in the production line.

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 the CMSD interface saved development time considerably.

4.2.4. CMSD interface – extensibility

In the previous subsection, how all the 'Get' functions make use of the support function *ComputeDuration* is described. This modular approach is used throughout the CMSD interface architecture. Several other support functions could be defined for the CMSD interfaces so that it could be extended to support the complete CMSD information model.

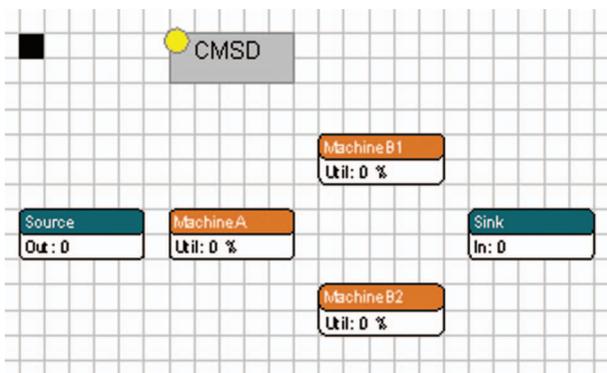


Figure 16. Enterprise dynamics example model.

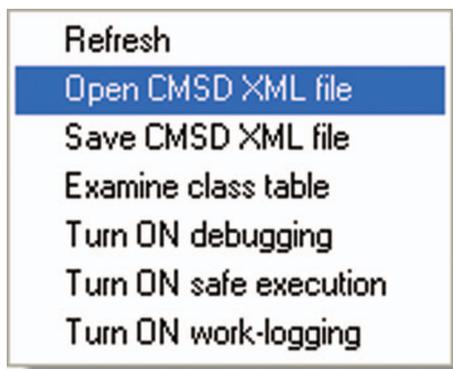


Figure 17. The context menu of the CMSD interface for plant simulation.

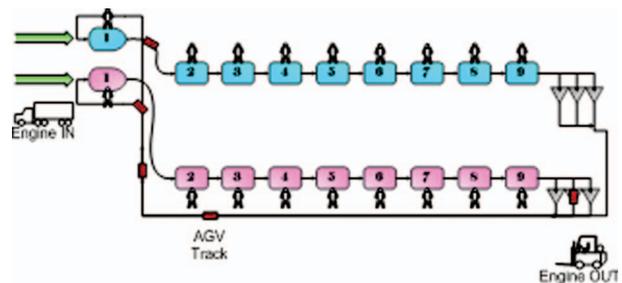


Figure 18. Engine line outline.

4.2.5. Summary of results

In this case study, two simulations of the same production process were created. A CMSD interface was created to allow each simulation to read the same information from the same CMSD file. An examination of the results of simulation runs for each simulation showed that they produced similar results. CMSD was able to represent the manufacturing data required for this study, and the creation of reusable data import objects based on CMSD reduced the effort for creating the simulations. This reduction in the required effort for simulation creation allowed not one but two production line simulations to be created. These results suggest that it is feasible to use CMSD as the basis for simulation-based shop floor analysis applications, and that the combination of CMSD and reusable import libraries can lead to reduced effort in simulation creation.

4.3. Truck manufacturer 2

This section presents a case study for an architecture that manages and makes available for analysis resource, process and simulation-related manufacturing data. A system based on this architecture was implemented at a truck engine manufacturing plant.

Several individual components make up the data management architecture and each component is intended to provide a functional capability that works synergistically with the other components. Different aspects of the CMSD specification were used to enable greater consistency in component structure and interface, and easier inter-component communication.

The CMSD specification provides a common conceptual model that allows each component to be

developed using the same notion of manufacturing entities such as resource, job and process, enabling a greater level of consistency for the internal representations of the data. In addition, the CMSD XML representation enables component data exchange by providing a format that is consistent with the component implementations and that is easily implementable over commonly available communication methods. The common manufacturing entity representation and data exchange formats provided by CMSD facilitate component interoperability and enable each component to be independently developed without affecting the other components in the architecture. Figure 19 presents a diagram that shows the four main parts of the data management architecture. Brief descriptions of the components are provided in this section. In the following sections, each of the components of the architecture will be described in detail.

- *The Resource Information Management (RIM) Database:* The RIM is composed of a relational database for data storage and a user interface for data selection and generation of XML files. The RIM database can be populated from the XML files using the *Generic Data Management (GDM)-Tool* (see below). It is also possible to manually populate the database through user interface.
- *The GDM-Tool:* The GDM-Tool provides the capabilities for extracting, converting and formatting data from a variety of relational, spreadsheet and document-based data sources. For additional information about this tool see the study by Balderud and Olofsson (2008).
- *The CMSD Standard Interface:* CMSD serves as a neutral mechanism to link the RIM database, the GDM-Tool, and the simulation models by providing an efficient and standardised structure for production data exchange.

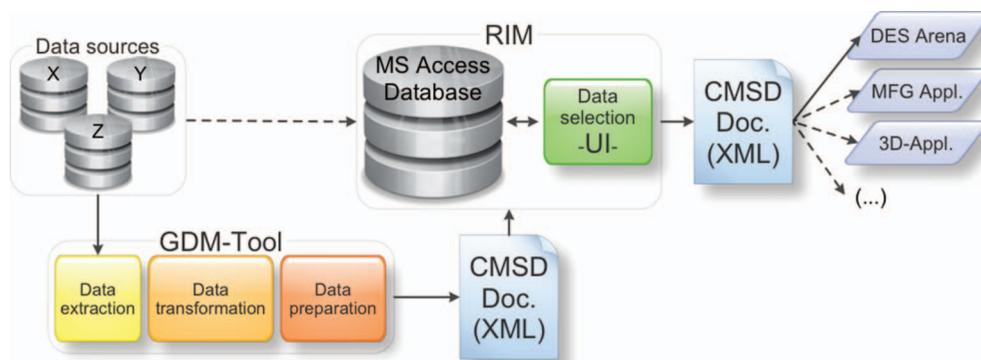


Figure 19. An outline of the data management architecture.

- *Discrete-Event Simulation (DES) Models*: The last element of the architecture is the DES models. The DES models might be created using any of a widely available array of DES packages.

4.3.1. Data storage: RIM Database

The RIM component consists of two parts: a relational database and a user interface. The database provides the means to store and retrieve generic blocks of resource descriptions created using the user interface. This component has three main objectives: (1) store information about resources and their properties, jobs and products, (2) reuse information for new DES projects and (3) structure information in a way that enables generation of standardised XML files describing factory resources. Figure 20 shows the entity-relationship diagram with all of the tables needed for database implementation. This database is designed to facilitate version management for resources by providing the capability to define reusable combinations of specifically configured resources.

4.3.2. User interface: RIM-UI

Figure 21 presents an image of the RIM-UI. The RIM-UI is a front end to the RIM-Database. It provides a means for invoking the functionality of the database and for generating XML files according to the CMSD specification. The RIM-UI provides a user-friendly means to define and store resource configuration

information and to access previously defined and stored resource configuration information. The RIM-UI works with the RIM-Database to generate XML files containing production line data. Its interface consists of four main parts:

- *Resource Selection (Part A)* provides means to associate a resource reference to a resource name. The resource reference is a string that uniquely identifies a resource in the DES model or another manufacturing application.
- *Resource Information (Part B)* gives the user access to all information concerning a resource, regardless of *Resource Configuration* choices. Information such as the resource provider, technical characteristics of the resource, energy consumption and a CAD drawing may be displayed for the selected resource.
- *Resource Configuration (Part C)* provides a means to define or modify the information about a resource or its characteristics. Each resource is associated with a product-job couple that links it to a set of properties, such as cycle times, MTBF and MTTR. This allows for aggregations of resource information to be created for different purposes. It is possible to select, create and modify information in order to define a resource. In fact, there are three possibilities to design a resource: (1) manual data entry, (2) reuse existing data from the database and (3) import historical production data processed by the GDM-Tool.

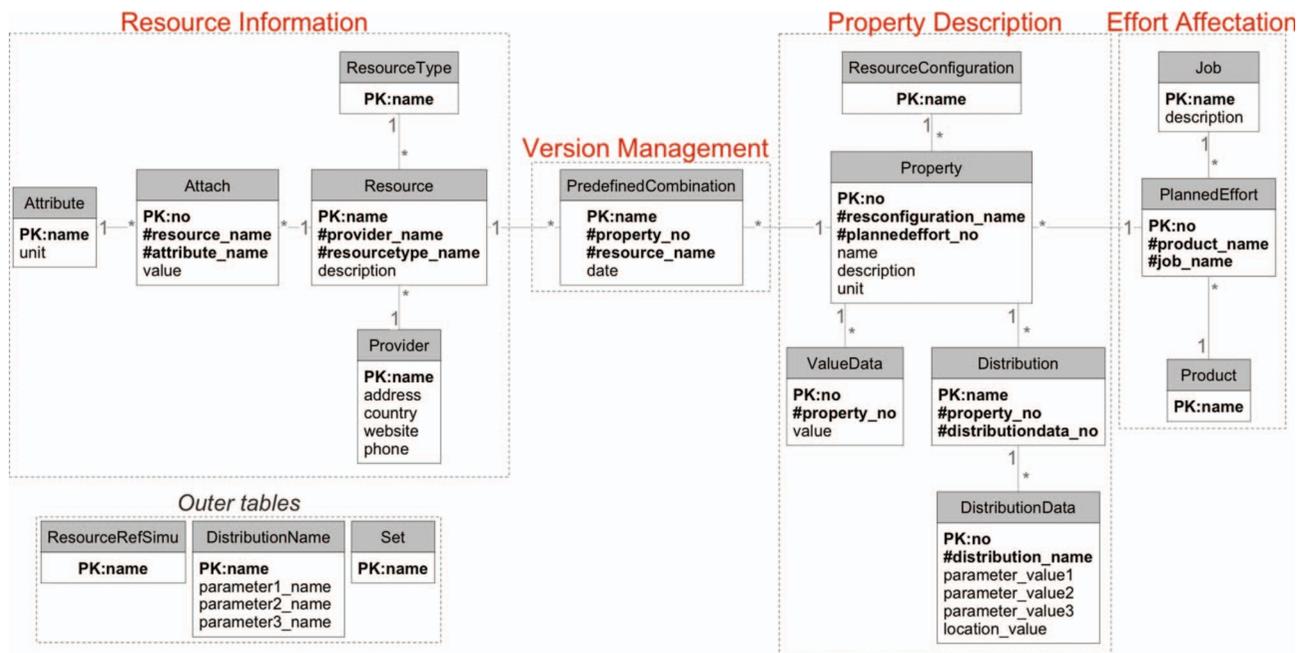


Figure 20. Enhanced entity-relationship diagram.

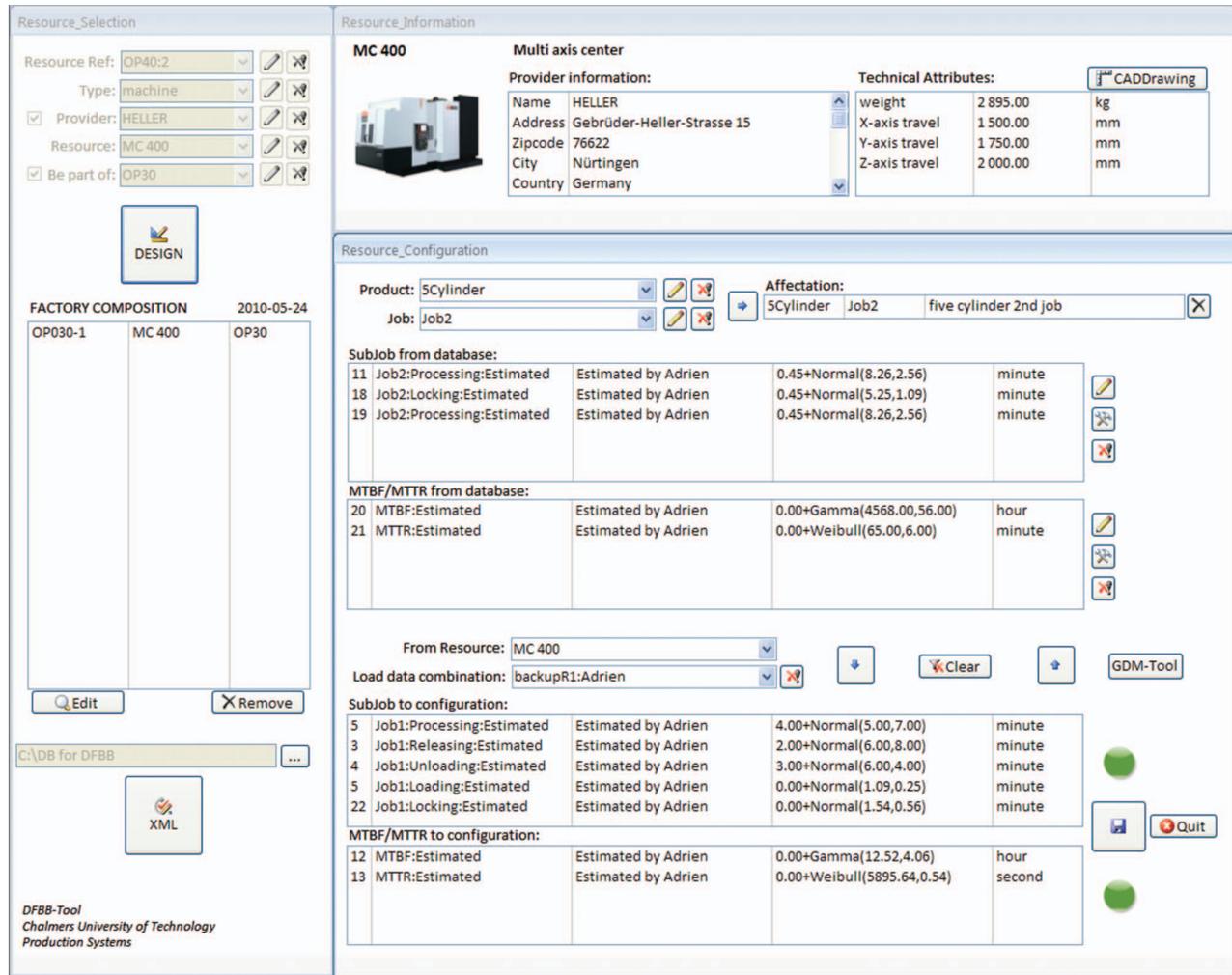


Figure 21. RIM user interface.

- *Factory Composition and XML Generation (Part D)* allows the user to change the factory configuration and create new XML instance documents. A new production line is added to the factory list each time a resource configuration, associated to a product-job couple, is saved. The user can add resources without defining specific logic because factory configuration is only focused on generating XML files with definitions for the resources and their characteristics. The simulation logic must be defined and implemented in the DES software.

4.3.3. Data exchange: CMSD document

In the CMSD specification, the logical boundary for an aggregation of CMSD information is called a CMSDDocument (ISO 2010). To realise this in the context of an XML instance document, the root element of the document will be a CMSDDocument

element. Nested within the root element is a DataSection element. Nested within this element will be multiple instances of Job and Resource elements. In Section 3.1.1, many other kinds of data that can be defined exchanged using CMSD were mentioned, but in this study, CMSD is used to exchange mainly Resource- and Job-related data.

Figure 22 shows a graphical depiction of the structure of the DataSection of a CMSD document generated for exchange between RIM components. The information is extracted from the RIM-Database using Structured Query Language (SQL) requests. VBA code and the Chilkat ActiveX Components (Chilkat Software 2010) are used to create the CMSD compliant XML instance documents from the extracted data.

4.3.4. Engineering application: ARENA simulation model

The purpose of generating the CMSD XML document is that it can be used to populate a DES model with

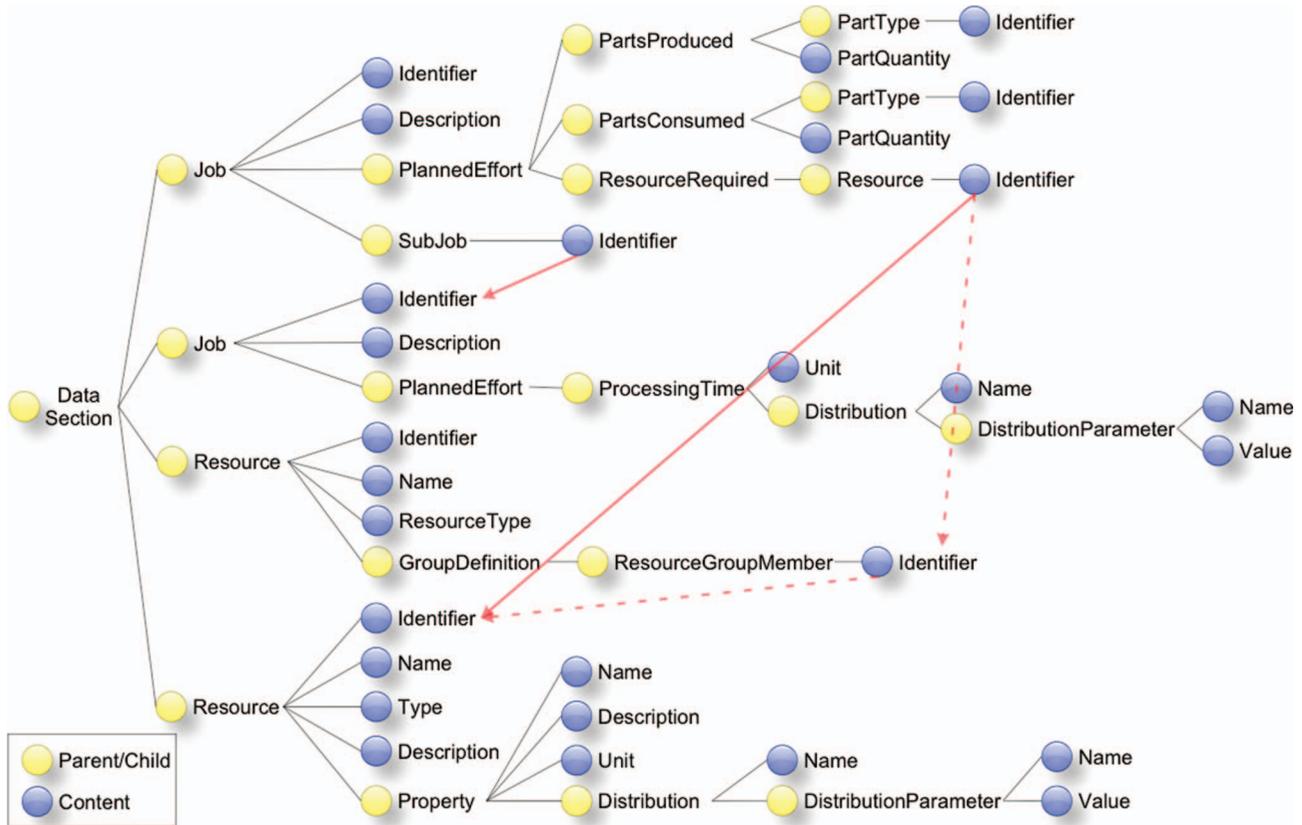


Figure 22. CMSD document – tree view.

information. In this case, the Rockwell Automation (2010) ARENA simulation product was chosen to test the data exchange between CMSD and simulation models. A model of a production line at a truck manufacturing company was built as a part of the test scenario. For each resource in the model, cycle times are specified in five steps (loading, locking, processing, realising and unloading). In addition, downtime patterns for machine, transporter and conveyor resources are specified. The CMSD XML files generated from the RIM-Database stores the input information necessary to run the simulation model. An image of the model is depicted in Figure 23. The main challenge is in creating a data-driven model such as this is parsing the input XML file, identifying which model elements are associated with the data in the file and updating the model elements with the corresponding data from the file.

4.3.5. Summary of results

In this study, an integrated system for conducting simulation analysis of production operations was created. A database for storing production-related data was designed based on a mapping of the CMSD

specification to a relation schema. The relation schema was extended to enable different configurations of production information to be created/modified to support different simulation studies. The XML representation of CMSD was used both to load historical information into the database and to exchange data with the simulation. The successful completion of this study proved that it is feasible to use CMSD as the basis for creating integrated simulation-based analysis applications.

5. Summary

Developing methods for the efficient exchange of information between simulations and other manufacturing tools have been a critical problem for many years. Standard representations for key manufacturing entities could help to address the issue. ISA-95, OAGIS and CMSD specifications offer some integration solutions. This article briefly describes and compares these specifications. From the comparison, a unique role for CMSD has been identified. Its ability to represent the characteristics of manufacturing entities that are governed by stochastic processes sets CMSD apart from other existing shop floor data

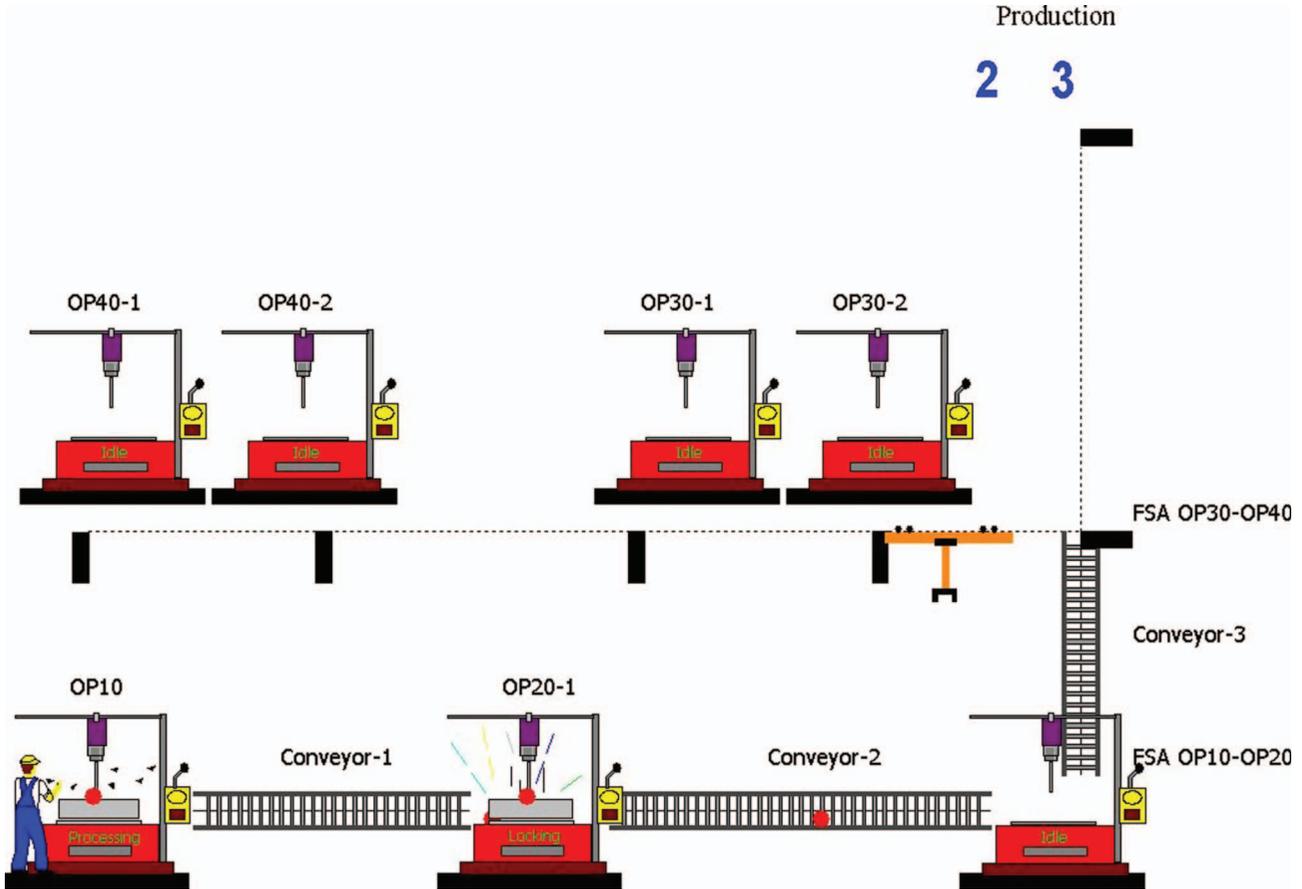


Figure 23. An ARENA snapshot of the test model used for the RIM-Tool development.

standards. This capability is crucial for developing applications that involve integrating DES with other manufacturing applications and technologies. This article also describes the pilot implementations of the CMSD specification for some car manufacturer's paint shop operations, some truck manufacturer's truck engine assembly line and another truck manufacturer's production line. These implementations successfully demonstrated the application of CMSD in manufacturing simulations.

This article also describes case studies that involved using CMSD as a basis for creating simulation-based applications for analysing manufacturing operations. The successful completion of these studies demonstrates the feasibility of using CMSD to represent the data and to provide the integration framework for simulation-based applications involving vastly different areas of manufacturing, such as simulations of paint shop operations, truck engine assembly and multi-line truck assembly.

The UML representation of CMSD is an approved SISO standard. Future work on CMSD includes creating and standardising a set of schemas to describe

the XML representation of CMSD. Multiple schema languages, such as RelaxNG, the XML Schema Definition Language 1.0, the XML Schema Definition Language 1.1 and Schematron, might be used to create the CMSD XML specification. In addition, researchers at the National Institute of Standards and Technology are working on C#- and Java-based support tools for validating and manipulating XML instance documents created based on CMSD. These activities will make integrating simulation and other manufacturing applications using CMSD effective, efficient and inexpensive.

Disclaimer

A number of software products are identified in the context of this article. This does not imply a recommendation or endorsement of the software products by the authors or the National Institute of Standards and Technology, nor does it imply that such software products are necessarily the best available for the purpose. The work described was funded by the United States Government and is not subject to copyright.

References

- Balderud, J. and Olofsson, A., 2008. *A plug-in based software architecture for generic data management*. Thesis (MS). Chalmers University of Technology.
- Banks, J., et al., 2005. *Discrete-event system simulation*. 4th ed. Upper Saddle River, NJ: Prentice-Hall.
- Brunnermeier, S.B. and Martin, S.A., 1999. *Interoperability cost analysis of the U.S. automotive supply chain*, Research Triangle Park, NC: Research Triangle Institute, Project Number. 7007-7003.
- Chilkat Software, 2010. *Chilkat ActiveX Components* [online]. Available from: <http://www.chilkatsoft.com/activeX.asp> [Accessed 5 September 2010].
- Gallaher, M.P., et al., 2004. *Cost analysis of inadequate interoperability in the U.S. capital facilities industry*. Gaithersburg, MD: National Institute of Standards and Technology (NIST). NIST GCR, 04-867.
- Gartner Group, 2010. *Gartner identifies the top 10 strategic technologies for 2010* [online]. Available from: <http://www.gartner.com/> [Accessed 20 May 2010].
- IEEE Standards Board, 1990. *IEEE standard glossary of software engineering terminology*. New York City: The Institute of Electrical and Electronics Engineers. IEEE Std 610.12-1990.
- ISA-95.COM, 2010. *ISA-95: The international standard for the integration of enterprise and control systems* [online]. Available from: <http://www.isa-95.com/> [Accessed 20 May 2010].
- Johansson, M. and Zachrisson, R., 2006. *Modeling automotive manufacturing process*. Thesis (MS). Chalmers University of Technology.
- Kibira, D. and McLean, C., 2007. Generic simulation of automotive assembly for interoperability testing. In: *Proceedings of the 2007 winter simulation conference*, Piscataway, NJ: Institute of Electrical and Electronics Engineers, 1035-1043.
- Lee, Y.T., 1999. Information modeling: from design to implementation. In: S. Nahavandi and M. Saadat, eds. *Proceedings of the second world manufacturing congress*. Canada/Switzerland: International Computer Science Conventions, 315-321.
- Microsoft Developer Network (MSDN), 2010. *Microsoft Corporation: building MSXML Applications* [online]. Available from: [http://msdn2.microsoft.com/en-us/library/ms753804\(VS.85\).aspx](http://msdn2.microsoft.com/en-us/library/ms753804(VS.85).aspx) [Accessed 5 September 2010].
- Muller, P.A., 1997. *Instant UML*. Birmingham: Wrox Press.
- National Research Council (NRC), 1995. *Committee to study information technology and manufacturing information technology for manufacturing: a research agenda*. Washington, DC: National Academy Press.
- National Research Council (NRC), 1998. *Committee on visionary manufacturing challenges. visionary manufacturing challenges for 2020*. Washington, DC: National Academy Press.
- Object Management Group (OMG), 2010. *Introduction to OMG's Unified Modeling Language* [online]. Available from: http://www.omg.org/gettingstarted/what_is_uml.htm [Accessed 20 May 2010].
- Open Applications Group (OAGi), 2010a. *Open applications group - open standards that open markets* [online]. Available from: <http://www.oagi.org/dnn2/> [Accessed 3 September 2010].
- Open Applications Group (OAGi), 2010b. *Open applications group - opening plenary, May 5, 2010* [online]. Available from: http://www.oagi.org/oagi/downloads/meetings/2010_0505_Gaithersburg/2010_0428_OAGi_Spring_Plenary.pdf [Accessed 3 September 2010].
- RELAXNG.org, 2010. *RELAX NG home page* [online]. Available from: <http://www.relaxng.org/> [Accessed 20 May 2010].
- Rockwell Automation, 2010. *Arena simulation software* [online]. Available from: <http://www.arenasimulation.com/> [Accessed 3 September 2010].
- Simulation Interoperability Standards Organization (SISO), 2010. *Standard for: core manufacturing simulation data - UML model, SISO-STD-008-2010*, 20 September 2010 [online]. Available from: http://www.sisostds.org/DigitalLibrary.aspx?Command=Core_Download&EntryId=31457 [Accessed 28 April 2011].
- Van der List, E., 2002. *XML Schema*. Sebastopol, CA: O'Reilly & Associates.
- Williams, T.J., 1992. *The Purdue enterprise reference architecture: a technical guide for CIM planning and implementation*. Research Triangle Park, NC: Instrument Society of America.
- World Batch Forum (WBF), 2010. *Business to manufacturing markup language (B2MML)* [online]. Available from: <http://wbfforg.affiniscap.com/displaycommon.cfm?an=1&subarticlenbr=99> [Accessed 28 April 2011].
- World Wide Web Consortium (W3C), 2010. *Extensible Markup Language (XML)* [online]. Available from: <http://www.w3c.org/XML> [Accessed 20 May 2010].