Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

A NEUTRAL INFORMATION MODEL FOR SIMULATING MACHINE SHOP OPERATIONS

Y. Tina Lee
Charles McLean
Guodong Shao

Manufacturing Systems Integration Division
National Institute of Standards and Technology
Gaithersburg, MD 20899-8260 U.S.A.

ABSTRACT

Small machine shops typically do not have the resources to develop custom simulations of their operations or data translators to import their data from other manufacturing software applications. This paper presents an overview of an information model currently under development at the National Institute of Standards and Technology (NIST) to address this problem. The model provides neutral data interfaces for integrating machine shop software applications with simulation. The information model provides mechanisms for describing data about organizations, calendars, work, resources, schedules, parts, process plans, and layouts within a machine shop environment. The model has been developed using the Unified Modeling Language (UML) and the Extensible Markup Language (XML).

1 INTRODUCTION

Standard interfaces could help reduce the costs associated with simulation model construction and data exchange between simulation and other software applications -- and thus make simulation technology more affordable and accessible to a wide range of potential industrial users. Currently, small machine shops do not typically use simulation technology because of various difficulties and obstacles associated with model development and data translation. Small shops typically do not have staff with the appropriate technical qualifications required to develop custom simulations of their operations or custom translators to import their data from other software applications.

NIST is working with a number of industrial partners and researchers to develop neutral formats for machine shop data to facilitate simulation and modeling activities. A machine shop data model, as a neutral interface format, has been under development to support both NIST's System Integration of Manufacturing Application (SIMA) program and the Software Engineering Institute's (SEI) Technology Insertion Demonstration and Evaluation (TIDE) Program. SIMA supports NIST projects in applying information technologies and standards-based approaches to manufacturing software integration problems (Carlisle and Fowler 2001). The TIDE Program is sponsored by the Department of Defense and SEI; it is currently engaged in a number of other projects with various small manufacturers in the Pittsburgh, Pennsylvania area. The technical work is being carried out as a collaboration between NIST, SEI, Carnegie Mellon University, Duquesne University, the iTAC Corporation, and the Kurt J. Lesker Company (KJLC).

KJLC is an international manufacturer and distributor of vacuum products and systems to the research and industrial vacuum markets. KJLC manufactures complete, automatically controlled vacuum systems with special emphasis on custom-designed, thin film deposition systems for research in alloys, semiconductors, superconductors, optical and opto-electronics. A small machine shop is contained within the KJLC manufacturing facility. KJLC's machine shop operation has been used to help define the requirements for simulation modeling and data interface specification activities described in this paper. Their facility will also be used as a pilot site for testing and evaluation of the simulation models, neutral data interfaces, and other software developed under this TIDE project. For more information on KJLC, see <www.lesker.com>.

The machine shop information model was developed with two goals in mind: a) support for the integration of software applications at a pilot facility -- KJLC's machine shop, and b) promotion as a standard data interface for manufacturing simulators and possibly for other software applications. The information model is continuing to evolve based on experience and feedback from KJLC's implementations and others involved in this effort.

The objective of the information modeling effort is to develop a standardized, computer-interpretable

representation that allows for exchange of information in a machine shop environment. The information model, when completed, must satisfy the following needs: support data requirements for the entire manufacturing life cycle, enable data exchange between simulation and other manufacturing software for machine shops, provide for the construction of machine shop simulators, and support testing and evaluation of machine shops' manufacturing software. Data structures contained within the information model include organizations, calendars, resources, parts, process plans, schedules, and work orders for machine shops.

An information model provides a sharable, stable, and organized representation of information in a selected domain area. The Integrated Computer Aided Manufacturing (ICAM) Definition Language 1 Extended (IDEF1X), EXPRESS, Unified Modeling Language (UML), and Extensible Markup Language (XML) are most often used by the manufacturing enterprises for information modeling. IDEF1X is a formal graphical language for relational data modeling, developed by the U. S. Air Force (Appleton 1985). EXPRESS (ISO 10303-11 1994) was designed to meet the needs of the STandard for the Exchange of Product model data, commonly called STEP (ISO 10303-1 1994), and it has been used in a variety of other "large-scale" modeling applications. UML is a graphic representation for artifacts in software systems, and is also useful for database design (OMG 2003). XML is a format for structured documents and it helps make possible information exchange in a globally distributed computing environment (W3C 2000).

Section 2 of this document provides an overview of the concepts behind the shop information model. Section 3 explains the constructs used to define the information model, how the model will be used, and gives some detailed examples of a small portion of the model. Section 4 provides conclusions and a discussion of future work.

2 CONCEPT FOR THE DATA MODEL

In this section, we introduce the concept of the shop information model from the user perspective. Our primary objective was to develop a structure for exchanging shop data between various manufacturing software applications, including simulation. The idea was to use the same data structures for managing actual production operations and simulating the machine shop. The rationale was that if one structure can serve both purposes, the need for translation and abstraction of the real data would be minimized when simulations are constructed. The mapping of real world data into simulation abstractions is not, for the most part, addressed in the current data model.

We also recognized that maintaining data integrity and minimizing the duplication of data were important requirements. For this reason, each unique piece of information appears in only one place in the model. Cross-reference links are used to avoid the creation of redundant copies of data.

The machine shop data model contains twenty major elements. Each of the major data elements are italicized in the discussion that follows. The data elements are called: *Organizations, Calendars, Resources, Skill-definitions, Setup-definitions, Operation-definitions, Maintenance-definitions, Layout, Parts, Bills-of-materials, Inventory, Procurements, Process-plans, Work, Schedules, Revisions, Time-sheets, Probability-distributions, References,* and *Units-of-measurement.* Figure 1 illustrates some of the major elements of the conceptual data model and their relationships to each other. Due to space limitations, the entire model is not shown or discussed in detail. For more detailed information on the model, see (McLean et al. 2003). The remainder of this section discusses the data elements and their significance.

Perhaps a good place to start the discussion of the data model is with the customer. Machine shops are businesses. They typically produce machined parts for either internal or external customers. Data elements are needed to maintain information on customers. The types of organizational information that is needed about customers is very similar to the data needed about suppliers that provide materials to the shop. The same types of organizational data are also needed about the machine shop itself. For this reason, an *Organizations* element was created to maintain organizational and contact information on the shop, its customers, and its suppliers.

Organizations can be thought of as both a phone book and an organization chart. The element provides sub-elements for identifying departments, their relationships to each other, individuals within departments, and their contact information. Various other types of information needs to be cross-referenced to organizations and contacts within structure, e.g., customer orders, parts, and procurements to suppliers.

The operation of the machine shop revolves around the production of parts, i.e., the fabrication of parts from raw materials such as metal or plastic. The raw materials typically come in the form of blocks, bars, sheets, forgings, or castings. These materials are themselves parts that are procured from suppliers. The *Parts* data element was created to maintain the broad range of information that is needed about each part that is handled by the machine shop. Part data includes an identifying part number, name, description, size, weight, material composition, unit-of-issue, cost, group technology classification codes, and revision (change) data. Cross-reference links are needed to the customers that buy the parts from the shop and/or the suppliers that provide them as raw materials. Links are also needed to other data elements, documents, and files that are related to the production of parts including: part specification documents, geometric models, drawings, bills-of-materials, and process plans.

The *Bills-of-materials* element is basically a collection of hierarchically-structured parts lists. It is used to define the parts and subassemblies that make up higher level part assemblies. A bill-of-materials identifies, by a part number reference link, the component or subassembly required at each level of assembly. The quantity required for each part is also indicated. Cross-references links are needed between parts that are assemblies and their associated bill-of-materials.

The *Parts* and *Bills-of-materials* elements establish the basic definition of parts produced or used by the shop. Another element, *Inventory,* is used to identify quantity of part instances at each location within the facility. *Inventory* data elements are provided for parts, tools, fixtures, and materials. Materials are defined as various types of stock that may be partially consumed in production, e.g., sheets, bars, and rolls. Structures are provided within inventory to keep track of various stock levels (e.g., reorder point level) and the specific instances of parts that are used in assemblies.

The *Procurements* element identifies the internal and external purchase orders that have been created to satisfy order or part inventory requirements. Cross-reference links are defined to *Parts* to identify the specific parts that are being procured and to *Work* to indicate which work items they will be used to satisfy.

The *Work* data element is used to specify a hierarchical collection of work items that define orders, production and support activities within the shop. Support activities include maintenance, inventory picking, and fixture/tool preparation. *Work* is broken down hierarchically into orders, jobs, and tasks.

Orders may be either customer orders for products or internally-generated orders to satisfy part requirements within the company, e.g., maintenance of inventory levels of stock items sold through a catalog. Orders contain both definition and status information. Definition information specifies who the order is for (i.e., customer cross-references), its relative priority, critical due dates, what output products are required (a list of order items, i.e., part references and quantities required), special resource requirements, precedence relationships on the processing of order items, and a summary of estimated and actual costs. Order items are also cross-referenced to jobs and tasks that decompose the orders into individual process steps performed at workstations within the shop. Status information includes data about scheduled and actual progress towards completing the order

Jobs typically define complex production work items that involve activities at multiple stations and ultimately produce parts. Tasks are lower level work items that are typically performed at a single workstation or area within the shop.

The *Process-plans* element contains the process specifications that describe how production and support work is to be performed in the shop. Major elements contained within *Process-plans* include routing sheets, operation sheets, and equipment programs. Routing and operation sheets are the plans used to define job and task level work items, respectively, in the work hierarchy. These process plans define the steps, precedence constraints between steps, and resources required to produce parts and perform support activities. Precedence constraints defined in a process plan are used to establish precedence relationships between jobs and tasks. Equipment program elements establish cross reference links to files that contain computer programs that are used to run machine tools and other programmable equipment that process specific parts. Each part in the *Parts* element contains cross-reference links to the process plans that define how to make that part. Jobs and tasks contain links back to the process plans that defined them.

The *Resources* element is used to define production and support resources that may be assigned to jobs or tasks in the shop, their status, and scheduled assignments to specific work items. The resource types available in the machine shop environment include: stations and machines, cranes, employees, tool and tool sets, fixtures and fixture sets.

The *Skill-definitions, Setup-definitions, Operations-definitions, Maintenance-definitions,* and *Time-sheets* elements provide additional supporting information associated with resources. *Skill-definitions* lists the skills that an employee may possess and the levels of proficiency associated with these skills. Skills are referenced in employee resource requirements contained in process plans. *Setup-definitions* typically specifies tool or fixture setups on a machine. Tool setups are typically the tools that are required in the tool magazine. Fixture setups are work-holding devices mounted on the machine. Setups may also apply to cranes or stations. *Operation-definitions* specifies the types of operations that may be performed at a particular station or group of stations within the shop. *Maintenance-definitions* specifies preventive or corrective maintenance to be done on machines or other maintained resources. *Time-sheets* is used to log individual employee's work hours, leave hours, overtime hours, etc.

The *Layout* element defines the physical locations of resource objects and part instances within the shop. It also defines reference points, area boundaries, paths, etc. It contains references to external files that are used to further define resource and part objects using appropriate graphics standards. Cross-references links are also provided between layout objects and the actual resources that they represent.

Schedules and *Calendars* data elements are used to deal with time. *Schedules* provides two views of the planned assignment of work and resources. Work items (orders, jobs, and tasks) are mapped to resources, and conversely, resources are mapped to work items. The

planned time events associated with those mappings are also identified, e.g., scheduled start times and end times. *Calendars* identifies scheduled work days for the shop, the shift schedules that are in effect for periods of time, planned breaks, and holiday periods.

The four remaining major data elements are *Revisions, References, Probability-distributions,* and *Units-of-measurement.* The *Revisions* element is used repeatedly throughout many levels of the data model. It provides a mechanism for identifying versions of subsets of the data, revision dates, and the creator of the data. The *References* element identifies external digital files and paper documents that support and further define the data elements contained within the shop data structure. It provides a mechanism for linking to outside files that conform to various other format specifications or standards, e.g., STEP part design files. The *Probability-distributions* element defines probability distributions that are used to vary processing times, breakdown and repair times, availability of resources, etc. Distributions may be cross-referenced from elsewhere in the model, e.g., equipment resources maintenance data. *Units-of-measurement* specifies the units used in the file for various quantities such as length, weight, currency, and speed.

The next section provides a detailed illustration of a small portion of the overall data model, and UML and XML file structures.

3 SPECIFICATION OF INFORMATION MODEL

An information model is a representation of concepts, relationships, constraints, rules, and operations to specify data semantics for a chosen domain of discourse. The advantage of using an information model is that it can provide shareable, stable, and organized structure of information requirements for the domain context. An information model serves as a medium for transferring data among computer systems that have some degree of compliance with this information model. For proprietary data, implementation-specific arrangements can be made when transferring those data (Lee 1999).

In general, the contents of an information model include a scope, a set of information requirements, and a specification. Information requirements serve as the foundation of the specification of the information model. A thorough requirements analysis is a necessity. The initial goal for the machine shop information model is to support data transferring needed for KJLC's machine shop operations. This information model, ultimately, will be promoted as a standard data interface to be used by other machine shops. Thus, the completeness and correctness of the information requirements and a consensus on the data requirements from the industry are also important issues.

The specification of the information model defines elements, attributes, constraints, and relationship between elements for the domain context. The specification should be laid out using some formal information modeling language. An information modeling language provides a formal syntax that allows users to unambiguously capture data semantics and constraints. Three types of methods that implement information models are currently used by the manufacturing community:

- Data transfer via a working form, which is a structured, in-memory representation of data. The method uses a mechanism that accesses and changes data sequentially without actually moving the data around. All shared data are stored in memory.
- Data transfer via an exchange file, which is a file with a predefined structure or format. This method requires a neutral file format for storing the data. The application systems read and write from files.
- Data transfer using a database management system. This method uses a database management system where information is mapped onto and retrieved from databases.

These implementation methods can be accomplished through translators that are developed using programming languages and database management systems. The selection of an implementation method is heavily dependent on the target environment where the application system resides. While the relational database is generally desirable for data transfer, the traditional file-oriented systems are being used continually by many manufacturing applications.

A specification for the machine shop information model has been developed based on the data model concept described in Section 2. Figure 2 shows the top level of the model. The *shop-data* element is represented by a *type*, an *identifier*, and a *number.* Optional elements include: *name*, *description*, *reference-keys*, *revisions*, *units-of-measurement*, *organizations*, *calendars*, *resources*, *skill-definitions*, *setup-definitions*, *operation-definitions*, *maintenance-definitions*, *layout*, *parts*, *bills-of-materials*, *inventory*, *procurements*, *process-plans*, *work*, *schedules*, *time-sheets*, *references*, and *probability-distributions.*

Type is an attribute of *shop-data* and is an enumeration to describe types about *shop-data*. *Identifier* is a key to uniquely identify the object internally within the system, and it is generated automatically by the system when the object is created. *Number* is also a unique key for identifying the object either when taken alone or possibly together with the object type, and the uniqueness is to the user or the user's organization. *Type*, *identifier* and *number* are required attributes. *Name* is used to identify the object by the user or user's organization. It is provided for readability sake. *Description* is used to describe the nature of the subject. *Reference-keys* refers to reference documents or files that are stored external to the model. When a data element's name suffixes with "*-key*" or "*-keys*", these data elements serve as pointers to the model to

avoid redefining the same set of information. All other attributes, such as *organizations*, *calendars*, *resources*, etc., are major elements of the model that were introduced in Section 2.

The machine shop data model specification is documented using both UML and XML structures. XML is chosen to support web users while UML's standard graphical notations provide visual communications. UML is a graphical representation; the language is for specifying, visualizing, constructing, and documenting, rather than processing. XML is a format for structured documents, thus XML documents are decodable.

The current version of the specification includes XML documents that are well-formed, but may or may not be validated. Data should be validated before being imported to a legacy system. An XML schema is a specification of the elements, attributes, and structures; it is not only useful for documentation, but also for validation or processing automation (van der Vlist 2002). Validation is the most common use for schema in the XML environment. The XML documents specification is now being extended to a schema using the World Wide Web Consortium (W3C) XML Schema, an XML schema language.

UML provides several modeling types, from functional requirements and activity analysis to class structures and component description. The modeling type used to map to the XML documents is the UML class diagram. A UML class diagram can be constructed to visually represent the structural and behavioural features. Since the behavioural feature is not relevant to the XML specification, that feature is omitted here (Carlson 2001).

The complete specification is not given here due to its size. Instead a sample data element specification is described. The data element of *orders* is chosen for illustration in this section. *Orders* is a subgroup of work and consists of a set of individual *order* data elements. It specifies a collection of production work orders to be processed within the shop. Each *order* contains the order definition and/or order status section. The order definition contains attributes of the order including a list of order items, i.e., a listing of individual parts that make up a particular order. The order status describes information about scheduled and actual progress toward completing the order. The same part may be listed in the order multiple times in different order items if each instance has unique attributes, e.g., different due dates.

3.1 UML Modeling

As mentioned before, the UML class diagram is one representation for the specification of the information model. A number of software tools are available for generating UML diagrams. The UML class diagrams introduced here have been generated using Microsoft Visio 2000. A UML class diagram can be constructed to graphically represent the classes, attributes, and relationships. A UML class is the abstraction of a concept in the domain of discourse; it is defined by a set of attributes. An attribute is an additional piece of information associated with a UML class. Each attribute defines its type (such as string, integer, date, or user defined data type), relationships, and optionally specifies its default value. A special type of class, named DataType, is used to specify enumeration items.

Relationships between classes are shown with the connecting line; the role and cardinality relationship may be presented along the relationship line. The role describes how the related class is used. There exist cardinality relationships between a class and its attributes, and between classes. The cardinality relationship specifies how many specific instances of an element could be related to another element. The cardinality relationship may be "one" to "zero or one", "one" to "zero or more", "one" to "one or more", or exactly "n" occurrences, and is presented in the Figure 3 as 0..1, 0..*, 1..*, n, respectively. The cardinality relationship used for attributes is enclosed by [ ].

The UML information model for the *orders* element is shown in Figure 3. The *orders* element has the attributes of *type* (which is a string), an *identifier* (which is an "int" or integer value), a *number* (which is a string), an optional *name* (which is a string), an optional *description* (which is a string), and an optional *reference-keys* and *revisions* (they are user defined data types). Figure 3 illustrates the cardinality relationships among *orders, order*, *order-definition*, and *order-status*. An *orders* element contains some *order* elements. Each *order* is defined by *order-definition* and has an *order-status*. *Orders* and *order* has "one" to "one or more" relationship, i.e., there may exist one or many *order* instances for an *orders* instance. Similarly, there may exist zero or one *order-definition* instance and zero or one *order-status* instance for an *order* instance. Each *order-definition* instance is defined by one *customers* instance, one *due-dates* instance, and zero or one of *priority-rating, order-items, precedent-constraints, resources-required,* and *cost-summary* instances.

3.2 XML Specification

XML supports the development of structured, hierarchical data entities that contain a high level of semantic content, that is both human and machine interpretable. There are several supporting standards from W3C that make working with XML easier. These include Document Object Management (DOM) for manipulating XML documents, XML Schema for defining the format of XML documents, and Extensible Style-sheet Language (XSL) for translating XML documents to other formats, see <www.w3.org>. There also exist commercial off-the-self software applications to implement creation, parsing, interpreting, and displaying of XML documents. The current version of

the XML specification of the information model has been developed using Microsoft XML Notepad.

An XML document is a collection of parsed and unparsed pieces. An element is one of the basic type of nodes in the tree represented by a XML document. A well-formed document has one unique root element that contains all other elements. Elements follow one another, or appear inside one another, but may not overlap. All elements must have a start-tag and an end-tag that surround their contents. An element begins with <*name-of-element*> (that is a start-tag) and ends with </*name-of-element*> (that is an end-tag). XML is case-sensitive. The contents of each element may include other elements. An XML element may be defined by a set of attributes and child-elements. (Child-elements are treated as attributes in the UML diagram.) Attributes and child-elements are additional information associated with the element. Attributes are presented in the start-tag, in the form:

<*name-of-element name-of-attribute*=”*value*”>

The same attribute can appear inside the start-tag once only. However, the same child-element may appear in the element more than once if it carries different instances. Attributes are unordered while child-elements are presented in order When an element has no content between the start-tag and end-tag or omits the end-tag and terminates the start-tag with “/>”, the element is an empty element. An empty element may contain attributes, however.

The XML structure for the *orders* element is shown below:

```
<orders type="" identifier="" number="">
    <name />
    <description />
    <reference-keys />
    <revisions />
    <order type="" identifier="" number="">
        <name />
        <description />
        <reference-keys />
        <revisions />
        <order-definition>
            <customers />
            <priority-rating />
            <due-dates />
            <order-items/>
            <precedent-constraints />
            <resources-required />
            <cost-summary />
        </order-definition>
        <order-status>
            <work-scheduled-progress />
            <work-actual-progress />
        </order-status>
    </order>
</orders>
```

In the above structure, the element of *orders* is defined by the attributes of *type, identifier,* and *number*, and the child-elements of *name, description, reference-keys, revisions,* and *order*. Order is further defined by the attributes of *type*, *identifier*, and *number*, and the child-elements of *name*, *description*, *reference-keys*, *revisions*, *order-definition*, and *order-status*. All attribute values are undefined in this case. Child elements are empty elements. Data types, cardinality relationships, constraints, default and values, enumerations are not included in this sample XML document. They will be defined in the XML schema that is currently under development.

4 CONCLUSIONS AND FUTURE WORK

This paper described the work being carried out by NIST in developing a neutral model and data exchange format for machine shop data. The objective of the information modeling effort was to develop a standardized, computer-interpretable representation that allows for the efficient storage and exchange of manufacturing life cycle data in a machine shop environment.

The information model will continue to evolve based on the experience and feedback from others involved in this effort. The model is now being transformed into a schema using an XML schema language. There are also plans to expand the model to include assembly line, supply chain, and other domain areas.

The current model addresses the exchange of real world data between simulation and other manufacturing software applications. Another information model and exchange file format is needed to support the simulation abstraction process. This model would be used to maintain data regarding the mapping of real world objects into their simulated representation. For example, as part of the abstraction process data values may be approximated, different colors may be substituted for real objects, shapes and sizes may be changed, and probabilistic distributions may be substituted for actual arrivals and other time-dependent events.

The information model will be proposed as a candidate standard to be considered by a formal standards body. A preliminary plan is in process for standardization through the Institute of Electrical and Electronics Engineers Standards Association (IEEE SA) 1516 Committee that was responsible for the Department of Defense High Level Architecture (HLA).

There are also experimental development activities underway to test the viability of the model with real world applications. A generic manufacturing simulator is being developed at NIST for the TIDE Program (McLean et al. 2002). The model is also being used in the TIDE Program to integrate a manufacturing execution system with a real-

time adaptive scheduler, and the manufacturing simulator. An aerospace manufacturer is also working on a prototype simulation based on the specification. A database implementation using Microsoft Access is currently underway.

ACKNOWLEDGMENTS AND DISCLAIMER

This project is funded by NIST's SIMA Program and the SEI TIDE Program. SIMA supports NIST projects applying information technologies and standards-based approaches to manufacturing software integration problems. No approval or endorsement of any commercial product by the National Institute of Standards and Technology is intended or implied. The work described was funded by the United States Government and is not subject to copyright.

REFERENCES

Carlisle, M., and J. Fowler. 2001. Systems Integration for Manufacturing Applications Biennial Report. Fiscal Years, NISTIR 6721. Gaithersburg, Maryland: National Institute of Standards and Technology.

Carlson, D. 2001. *Modeling XML Applications with UML: Practical e-Business Applications.* Boston, MA: Addison-Wesley.

D. Appleton Company, Inc. 1985. Integrated Information Support System: Information Modeling Manual, IDEF1-Extended (IDEF1X). Wright-Patterson Air Force Base, Ohio.

ISO 10303-1. 1994. Part 1: Overview and Fundamental Principles. *Industrial Automation Systems and Integration-Product Data Representation and Exchange*. Geneva, Switzerland: International Organization for Standardization.

ISO 10303-11. 1994. Part 11: The EXPRESS Language Reference Manual. *Industrial Automation Systems and Integration-Product Data Representation and Exchange*. Geneva, Switzerland: International Organization for Standardization.

Lee, Y. T. 1999. Information Modeling: From Design To Implementation. *Proceedings of the Second World Manufacturing Congress,* ed. S. Nahavandi and M. Saadat, 315-321. Canada/Switzerland : International Computer Science Conventions.

McLean, C., A. Jones, T. Lee, and F. Riddick. 2002. An Architecture for a Generic Data-Driven Machine Shop Simulator. In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yucesan, C. Chen, J. L. Snowdon, and J. M. Charnes, 1108-1116. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

McLean, C., T. Lee, G. Shao, F. Riddick, and S. Leong. 2003. Shop Data Model And Interface Specification. Draft NISTIR. Gaithersburg, Maryland: National Institute of Standards and Technology.

Object Management Group (OMG). 2003. Unified Modeling Language [online]. Available online via <http://www.omg.org/uml/> [accessed June 27, 2003].

World Wide Web Consortium (W3C). 2000. Extensible Markup Language (XML) 1.0 (second edition) [online]. Available online via <http://www.w3.org/TR/REC-xml.html> [accessed June 27, 2003].

van der Vlist, E. 2002. *XML Schema.* Sebastopol, CA.: O'Reilly& Associates, Inc.

AUTHOR BIOGRAPHIES

Y. TINA LEE is a computer scientist in the Manufacturing Simulation and Modeling Group at NIST. She joined NIST in 1986. Most recently, she has been working on the design and development of interface information models to support the Software Engineering Institute (SEI) Technology Insertion Demonstration and Evaluation (TIDE) project. Previously she worked at the Contel Federal Systems and at the 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>.

CHARLES MCLEAN is a computer scientist and Program Manager of the Manufacturing Simulation and Visualization Program at NIST. He also leads the Manufacturing Simulation and Modeling Group. He has managed research programs in manufacturing simulation, engineer
ing tool integration, product data standards, and manufacturing automation at NIST since 1982. He has authored more than 50 technical papers on topics in these areas. He is on the Executive Board of the Winter Simulation Conference and the Editorial Board of the International Journal of Production, Planning, and Control. He is formerly the Vice Chairman of the International Federation on Information Processing (IFIP) Working Group on Production Management Systems (WG 5.7). He is also the NIST representative to the Department of Defense's Advanced Manufacturing Enterprise Subpanel. He holds an MS in Information Engineering from the University of Illinois at Chicago and a BA from Cornell University. His e-mail address is <mclean@cme.nist.gov>.

GUODONG SHAO is a computer scientist with the Intelligent Automation Inc., and a guest researcher in the Manufacturing Simulation and Modeling Group at NIST. He has participated in research relating to FMS, CIMS, and

manufacturing simulation integration for many years. He holds a Master's Degree from the University of Maryland, College Park. He is a Ph.D. student in the Graphics Laboratory at George Mason University. His e-mail address is <gshao@cme.nist.gov>.

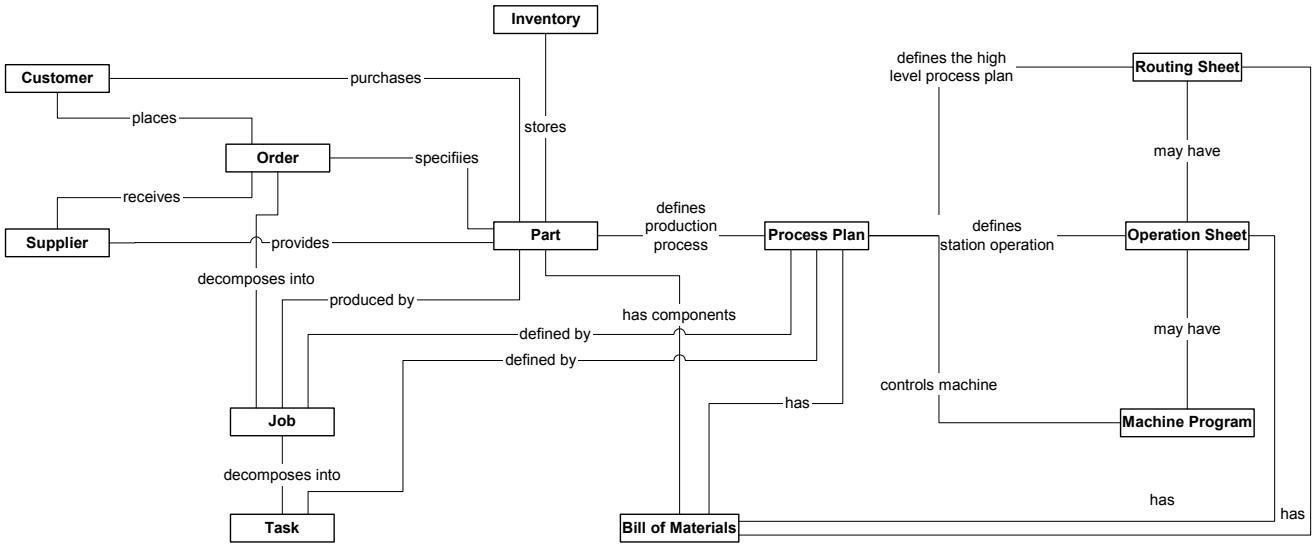

Figure 1: Concept for the Machine Shop Data Model

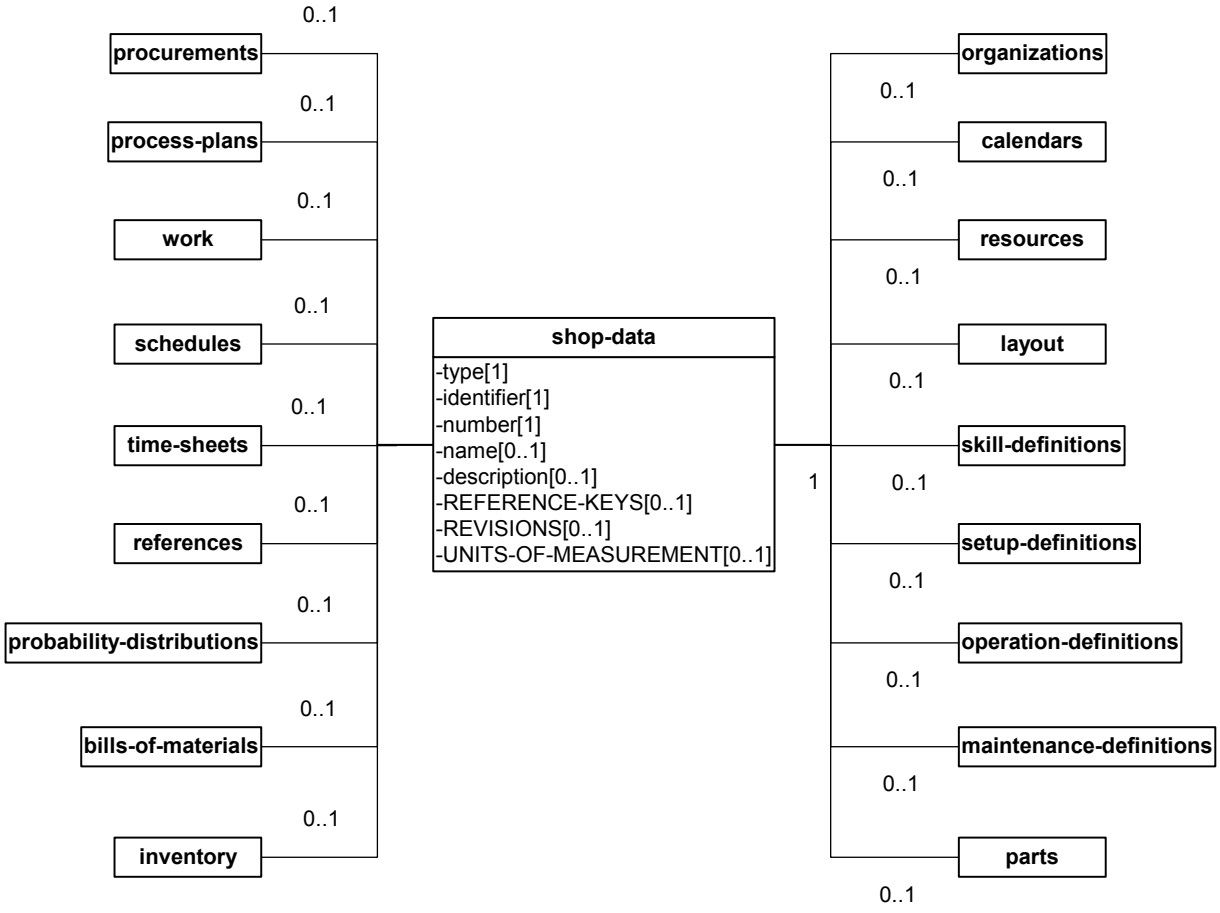

Figure 2: Top Level of the Shop Data Model

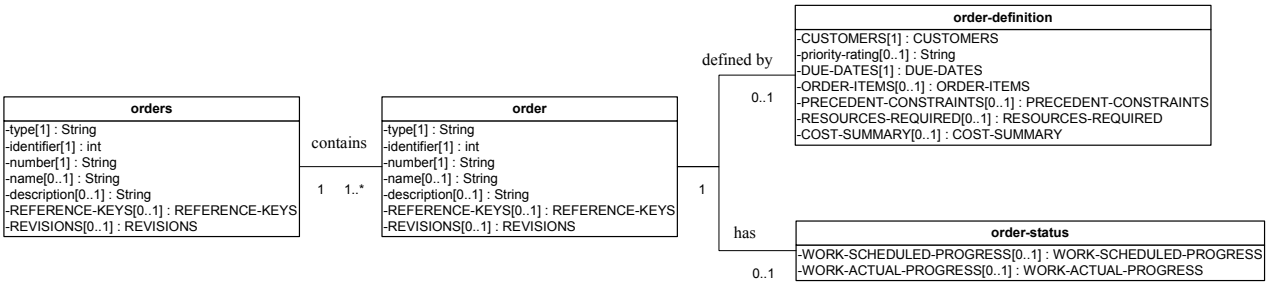

Figure 3: UML Model of the *Orders* Object