2005-07-07

Suitability of the NIST Shop Data Model as a Neutral File Format for Simulation

Gregory Brent Harward
Brigham Young University - Provo

SUITABILITY OF THE NIST SHOP DATA MODEL AS A NEUTRAL FILE FORMAT FOR SIMULATION

by

Gregory Brent Harward

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

School of Technology

Brigham Young University

August 2005

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Gregory Brent Harward

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Charles R. Harrell, Chair
Date	A. Brent Strong
Date	Val Hawks

BRIGHAM YOUNG UNIVERSITY

I have read the thesis of Gregory Brent Harward in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative material including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

_____ _____

Date

Charles R. Harrell
Chair, Graduate Committee

Accepted for the Department

_____

Thomas L. Erekson
Director, School of Technology

Accepted for the College

_____

Alan R. Parkinson
Dean, Ira A. Fulton College of Engineering and Technology

ABSTRACT

SUITABILITY OF THE NIST SHOP DATA MODEL AS A NEUTRAL FILE FORMAT FOR SIMULATION

Gregory Brent Harward

School of Technology

Masters of Science

Due to the successful application in internet related fields, Extensible Markup Language (XML) and its related technologies are being explored as a revolutionary software file format technology used to provide increased interoperability in the discrete-event simulation (DES) arena. The National Institute of Standards and Technology (NIST) has developed an XML-based information model (XSD) called the Shop Data Model (SDM), which is used to describe the contents of a neutral file format (NFF) that is being promoted as a means to make manufacturing simulation technology more accessible to a larger group of potential customers.

Using a two step process, this thesis evaluates the NIST SDM information model in terms of its ability to encapsulate the informational requirements of one vendor's simulation

model information conceptually and syntactically in order to determine its ability to serve as an NFF for the DES industry. ProModel Corporation, a leading software vendor in the DES industry since 1988, serves as the test case for this evaluation. The first step in this evaluation is to map the contents of ProModel's information model over to an XML schema file (XSD). Next, the contents of this new XSD file are categorized and compared to the SDM information model in order to evaluate compatibility. After performing this comparison, observations are made in relation to the challenges that simulation vendors might encounter when implementing the proposed NIST SDM.

Two groups of limitations are encountered which cause the NIST SDM to support less than a third of the ProModel XSD elements. These two groups of limitations are: paradigm differences between the two information models and limitations posed due to the incomplete status of the NIST SDM specification.

Despite these limitations, this thesis shows by comparison that XML technology does not pose any limitation which would invalidate its ability to syntactically represent a common information model or associated XML NFF. While only 28% of the ProModel element are currently supported by the SDM, appropriate changes to the SDM would allow the information model to serve as a foundation upon which a common information model and neutral file format for the DES industry could be built using XML technologies.

ACKNOWLEDGEMENTS

To the woman that makes me feel like no other

To the woman who is my favorite mother

To the woman who makes my heart beat fast

My forever friend, lets make it last

I watch my son, and he me.

- Humility, trust, flexibility -

Open mind, is it me or he?

Scooter Scoot what a cutie little boot

So happy, I love your smile

Thanks so much for finally coming to stay a while

“And when you dream, dream big” – Ryan Shupe

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

There exists a niche market of software vendors who distribute software products that are designed to address the need to simulate existing and proposed processes in an effort to make intelligent decisions about how to improve those processes without affecting existing systems (Bateman, 1997). Among this group of simulation software vendors are at least a dozen software companies that market software products which utilize discrete-event simulation as a method of modeling real world events in the manufacturing arena. In 2003 at a simulation conference it was stated, "While the art of simulation continues to flourish, the continuity and the consistency of the simulation data usage may have not kept pace" (McLean, 2003). "Hence, it takes real effort when exchanging simulation models during scenario iterations among project participants" (Lu, 2003). One proposed solution to the problem of exchanging dissimilar simulation model information is to develop a neutral file format (NFF). "The development of neutral, vendor-independent data formats for storing simulation models could greatly improve the accessibility of simulation technology to industry by enabling the development of reusable models" (McLean, 2002). This type of solution would help to "make simulation technology more affordable and accessible to a wide range of potential industrial users" (Lee, 2003).

1.2 Statement of the Problem

As a step toward the creation of an NFF, "NIST has been developing an information model in an XML-based exchange file format that facilitates the exchange of information between manufacturing simulation applications and other manufacturing applications and/or data sources" (McLean, 2002). This information model is designed to conceptually outline model information in an extensible markup language schema (XSD). Utilizing this XSD, the contents of an information model could then be exported to a portable NFF using extensible markup language (XML) syntax. The purpose of this research is to analyze the ability of the NIST XSD information model to represent simulation model information conceptually. This evaluation also includes an inherent test of the XML syntax and its ability to represent simulation model information syntactically.

1.3 Thesis Statement

This thesis focuses on four questions which are:

1. Can the data elements found in a typical discrete-event simulation (DES) modeling tool be represented using XML syntax?
2. Can these elements be mapped to the XSD information model proposed by NIST?
3. When mapping the data of a typical manufacturing DES modeling tool to the NIST XSD information model, what percentage of the data elements can be represented?
4. What challenges might simulation vendors encounter when implementing the proposed NIST XSD?

1.4 Justification

Simulation models are complex and require time and technical skill to build. Often the time invested in simulation modeling is spent by companies that are trying to optimize processes in an effort to save money. With this desire to save money comes the desire to reuse existing model technology in order to increase the return on investment of funds already spent on existing simulation technology. In addition, in order to allow model reusability some simulation vendors have implemented the ability to build model components which contain all of the required information to represent a particular operation or process. These components or sub models can then be recombined and reused at a later date to form new simulation models thereby increasing the return on original investment through the reuse of model technology while also saving time. Currently, software packages that produce simulation models and sub models maintain information by storing model information in binary files. These binary files are proprietary in nature and can not be used across different discrete-event simulation products or shared with other non-simulation related software packages unless a custom translator is first developed by the simulation vendor. Because simulation vendors do not share a common file type, the ability to share domain knowledge contained within simulation models is restricted. The inability to reuse model information makes the decision to use simulation technology an expensive alternative and therefore less attractive to companies that would benefit from the technology (McLean, 2002).

One approach to increasing the accessibility of simulation technology is to create a standard information model for the simulation industry. This common information model

defines the contents of an NFF which provides the information model in a portable format to be shared and reused across the industry. This approach alleviates the need to create binary file translators in order to use simulation models created with different software products. An NFF also gives simulation model technology more usage flexibility thus providing more utility.

The creation of a standard information model and associated NFF is a difficult process that requires the mutual collaboration of industry leaders. To serve effectively as a common standard, an information model must also be integrated with existing software vendor technology. In order for the proposed NIST XSD information model to be promoted across the simulation industry as a standard, it behooves the simulation vendors affected by the proposed standard to make an evaluation to determine the ability of the proposed information model and associated NFF to serve as an effective standard. Such an evaluation helps to determine if the current direction of the methodology undertaken by NIST is progressing in a direction that is conducive to vendor needs in terms of supporting the transfer of simulation model file information using XML technologies. This thesis evaluates the proposed NIST XSD information model in terms of its ability to encapsulate the informational requirements of one vendor's simulation model file conceptually and syntactically in order to determine the ability of the NIST XSD to serve as an NFF for the DES industry.

1.5 Methodology

In the software industry, when an NFF is not already supported as a means for transferring information between two dissimilar information model formats, automated file translators are created. Accordingly, as part of the roadmap for implementation of the NIST XSD information model, the Shop Data Model (SDM) proposed by NIST includes the creation of XML parsers designed to automatically convert XML code to and from data structures which reside in computer memory (Lee, 2003).

In order to evaluate the ability of the NIST XSD to serve as an NFF for the DES industry, the process of manually creating a file translator is utilized as a method of evaluating what elements of a sample vendor's information model are supported in the NIST XSD. The process of manually creating a file translator reveals the ability that the NIST XSD has in supporting model elements from a simulation vendor's information model.

Currently, only two documented cases of interaction between the NIST SDM and DES vendor software packages exist. These two vendors consist of ProModel software from ProModel Corporation and Quest from DELMIA Corporation. These interactions are covered in more detail in Chapter 2, but generally consist of the population of the vendor's information model with an existing XML information model created utilizing the NIST XSD. Neither of these interactions deals with the translation of a vendor's information model into the NIST XSD for the purpose of evaluating it as an NFF candidate.

ProModel Corporation, a leading software vendor in the DES industry since 1988, was chosen as the test case for this evaluation. Currently ProModel software utilizes a binary file format for storing simulation model information. Because of this, a two step information model transformation is required. These two steps are:

1. A ProModel XSD is created that represents the contents of the ProModel information model using XML syntax. This step allows for the ProModel information model to be stored in an XSD format rather than in a binary format.
2. The contents of this new ProModel XSD are then categorized to determine what model elements are vital and thus belong in an NFF. The resultant set of XSD elements are then compared with the contents of the NIST XSD information model to manually test the percentage of data elements that are supported in the NIST XSD.

The execution of these two steps supplies insight into the specific challenges that a simulation vendor might face when implementing support for the NIST XSD information model. Additionally, the results of the comparison, described in step two, forms a compatibility reference to help measure the ability of the NIST XSD to serve as a viable NFF for the DES industry.

1.6 Delimitations

In order to effectively evaluate and develop an XSD information model that will be used to generate an NFF to serve as an industry standard, a consensus on the file format and

structure is necessary from the majority of affected interests in order for the resultant XSD information model and associated NFF to successfully serve the requirements of a common file format. Due to the large scope of such an evaluation within the DES industry, this thesis is limited to the ability for the NIST XSD information model to accurately represent the information contained in one vendors software product; namely a ProModel version 6.2 binary file. Additionally, this evaluation is made at the lowest file level in an attempt to explore all possible information mappings associated with importing the binary ProModel (*.mod) model file into the NIST XSD information model. This approach is preferred as a method to make a complete evaluation which is not based on the contents of any one particular model or models which have the possibility of not serving as a completely robust sample. The evaluation does not make any assumptions related to the exporting of the NIST XSD information model into the proposed ProModel XSD. Also, the evaluation does not attempt to explore all of the possible combinations of model contents or the limits of these data mappings when considering the volume of data that can be stored utilizing the NIST XSD information model. As mentioned previously, the parameters of this evaluation are set in accordance with the contents of the NIST XSD information model draft specification currently at the time of this review. In situations when the currently available NIST XSD information model provides no possible mappings or the specification is incomplete, recommendations are given as to possible mappings that could be used.

1.7 Definition of Terms

ActiveX and COM (Component Object Model):

Component Object Model is the Microsoft framework for developing and supporting program component objects. This technology provides the ability to dynamically load components in an as-needed fashion in order to use requested features.

DES (Discrete – Event Simulation):

Discrete- event simulation is a type of simulation modeling that models the effects of the events that occur within a system as they occur at particular points in time. (Harrell, Ghosh, & Bowden, 2000)

NFF (Neutral File Format):

A neutral file format is one that is used to interchange data between different manufacturing software applications at a common level. The term is generic and used by industry to signify any number of standard formats.

NIST (National Institute of Standards and Technology):

A non-regulatory federal agency of the United States Department of Commerce; NIST was formerly known as the National Bureau of Standards (NBS) until 1988 and was founded for building the foundations for technological progress in 1901.

ProModel:

ProModel is a commercially available simulation software product distributed by ProModel Corporation. It is used for evaluating, planning, or redesigning manufacturing, warehousing, and logistics systems. This discrete-event simulator allows users to build graphical representations of systems and perform what-if scenarios via multiple simulated iterations. (McLean, 2001)

SDM (Shop Data Model):

The Shop Data Model and Interface Specification (SDM) describe the contents of the NIST XSD information model. The contents of this information model constitute what would then be included in an eventual NFF for simulation models. This study was performed using the latest available draft of the specification dated February 24, 2003. (McLean, 2003)

SDX (Simulation Data eXchange):

SDX is an NFF developed by Engineering Animation Incorporated used to transfer data between different simulation applications and CAD software packages.

SGML (Standard Generalized Markup Language):

SGML was developed and standardized by the International Organization for Standards (ISO) in 1986 and specifies the rules for marking text with tags. These tags are then interpreted to supply formatting for the associated text.

W3C (World Wide Web Consortium):

The World Wide Web Consortium (W3C) develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential. W3C is a forum for information, commerce, communication, and collective understanding. (http://www.w3.org/)

XML (Extensible Markup Language):

Extensible Markup Language, abbreviated XML, describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language. By construction, XML documents are conforming SGML documents (W3C, 2004).

XSD (XML Schema Definition) or XML Schema:

Published as a recommendation in May 2001 by W3C, a schema defines how data types are assigned to each tag and any attributes that are contained in the XML document. A schema is also a structured document which must obey XML syntax rules. It is composed of a series of predefined tags and attributes that are part of the XML language and are used to set the data types for the values associated with custom tags (Devguru, 2004).

CHAPTER 2

REVIEW OF THE LITERATURE

2.1 Introduction

Driven by its success in integrating successfully with internet related technologies, the XML file format it is now being used as an NFF in office applications such as Microsoft Office 12 and Sun Star Office 7 (Sun Microsystems, 2005; Microsoft 2005). Microsoft has shown particular interest in XML technologies and is currently in the process of integrating Office with Computer Aided Design (CAD) applications. Even with the emerging interest that XML technologies has gained, the use of XML in the DES arena remains largely unexplored. A survey conducted in August 2003 listed forty-nine simulation software products, only four of which were utilizing XML technologies in their software products (Lionheart, 2005). Despite this, the reputation of XML technologies due to successful implementation in other arenas has caused the DES industry to take notice.

This chapter covers the history of XML technologies as it relates to the creation of an NFF within the DES industry. Each of the topics covered has served a preparatory role in creating an overall interest in the creation of an NFF to serve the DES industry based on XML technologies. Following these topics, a background of XML history and XML syntax is provided as it relates to the objectives of this thesis. Last, the NIST Shop Data Model and its proliferation are discussed.

2.2 SDX a Legacy NFF for DES

The Simulation Data eXchange (SDX) file format is an NFF that has been successfully used in order to improve data exchange issues between dissimilar simulation models. Originally developed by Shreekanth Moorthy of Engineering Animation, Inc, the SDX file format was intended to be open source in nature and enable the transfer of DES model information via an NFF. Currently, four simulation companies, namely AutoMod, WITNESS, Quest, and Simul8 support SDX for import functionality, while only one software package, FactoryCAD, supports the creation of the SDX file format for the purpose of exporting.

While originally intended to serve as an NFF, the usage of the SDX file format has primarily been limited to the transfer of routing information and environment layout information such as equipment, machines, and conveyors with their associated breakdown, cycle time, and scrap rate information from FactoryCAD software to one of the DES software packages listed. In this limited scope, the SDX file format has been used as an efficient means of transferring model information. On a larger scale, the acceptance of the SDX file format as a viable NFF in the DES industry has been limited. This is due to the inability that the SDX file format in transfering all of the information required by DES software packages, the necessity of creating creating custom software translators before it can be used, and limited support for the file format from industry vendors (Kim, 2003).

The SDX file format has contributed to the formation of an NFF for the DES industry by supplying specification for representation of layout elements. However, the SDX file format is not robust enough to provide a comprehensive NFF solution for the DES industry. For this reason XML technologies are now being explored as a future solution.

2.3 XML in Computer Aided Design

Currently in the Computer Aided Design (CAD) industry the XML file format is supported minimally via file translators or proprietary XML schemas. Similar to the DES industry, there is focus within the CAD industry to utilize XML technologies to create an NFF. This effort is currently being pursued by two separate groups. Each of these groups is working to unlock the CAD information which is currently hidden inside proprietary file formats and making this information easily sharable while at the same time maintaining data protection of sensitive content. It is thought that an NFF will allow for the sharing of CAD information with other applications within the industry and also allow for the information to be shared with unrelated business verticals though a common file format.

The first of two groups focused on the creation of an XML based NFF is the Web3D consortium whose members include Hewlett Packard, Sun Microsystems, and 3Dlabs. The Web3D consortium is working to develop an XML file format called CAD Distillation Format (CDF) which is geared to help drive productivity, cut costs, and raise revenue streams in sales and marketing departments that desire to use CAD graphics without exposing sensitive design information (Boulton, 2004). The CDF format is based

on an existing 3D file format called X3D which is in the final stages of ratification by the International Standards Organization. The goal of the file format architecture is to extract out the common information contained in CAD files that is useful throughout the industry. This common information can then be shared throughout the industry using the CDF file format represented using XML syntax.

The second group that is interested in the development of a CAD XML NFF is comprised of Microsoft Corporation and a CAD industry market leader Dassault Systemes. These two companies have formed a non-exclusive strategic business alliance. One of the goals of this alliance is the advancement of XML technology within the CAD industry to enable CAD files to be integrated seamlessly with Microsoft Office products. Currently, Dassault Systemes has integrated the 3D-XML file format into its existing line of software products which include CATIA, DELMIA, ENOVIA, SMARTEAM, SolidWorks, and Spatial (Rowe, 2004).

There is obvious interest within the CAD industry to create an NFF so that years of intellectual investment contained in proprietary CAD formats can more freely be shared. XML and its related technologies are being explorer as a means to provide this NFF capability.

2.4 Open SML an Open Source NFF for DES

The LINUX (Linux, 2005) computer operating system has thrived under the open source development paradigm. Open source software development allows for any member of

the open source community to contribute to the body of computer source code while being governed by a set of organization rules and a governing central body. The net result of this type of development is a product which is royalty free, because it is the culmination of free member contributions, and flexible, as it can be changed by any contributing member. The concept of open source software development has been applied to the problem of developing an NFF for DES industry (Wiedemann 2002).

During the 2001 Winter Simulation Conference, OpenSML-project was presented. This project focuses on an open source development approach in order to create Simulation Modeling Language (SML) as a method to supply common core simulation functionality for the DES industry. Originally developed as a Java based programming language, the program is set up to leverage the open source development process in order to develop a standard DES language that is readable, modular, and extensible (Kilgore 2001).

The open source software development model used by OpenSML-project is another approach that is being explored in order to create an NFF for the DES industry.

2.5 The Use of XML as an NFF in Simulation Modeling

This section provides information relating XML technologies to simulation modeling projects. The initiatives listed apply XML to varied simulation approaches some of which are discrete-event in nature.

2.5.1 DEVS

Discrete-Event System Specification (DEVS) is a method of describing a system via mathematical language or formalism. Using DEVS, the components of a system are defined which are then coupled together to form a system representation which is hierarchical in nature. This method of specifying model information facilitates the ability of DEVS to serve as an object-oriented simulation language with the ability to create modular models. Because the models created using DEVS are hierarchical in nature, the components have the ability to be shared and reused by other DEVS based systems. In order to support model sharing, an XML translator has been created to translate DEVS models into portable XML files. These XML files can then be reused to serve as components to other models in additional simulation projects. For DEVS, XML successfully serves as a standardized NFF which contains all of the necessary model information for a DEVS model to be portable and reusable (Wang 2002).

2.5.2 BOM

SimVentions (SimVentions, 2005) is a simulation company specializing in Distributive Interactive Simulations (DIS) and High Level Architecture (HLA). The implementation of DIS is accomplished with the use of the Base Object Model (BOM), also developed by SimVentions. The concept of Base Object Model (BOM) is that simulation activity consists of common patterns and interplay between simulated entities. These commonalities can be extracted into stand-alone building blocks and stored in an XML format, which can then be reused and regrouped with other building blocks to form new simulation models. This methodology allows for the reuse of time tested modeling

components which provides advantages such as interoperability, reusability, adaptability, and the use of repositories. Because the model components are stored in a platform independent XML format, they can be easily gathered or distributed via web services in an on-demand fashion from an internet based repository. The concept also specifies that once the required XML source files are on the host computer, the files are recompiled into the native code language of the host application. This set up allows for web repositories which host BOM models in the XML file format which serve as an NFF for a host system and application configurations of any flavor (Gustavson, 2004).

BOM serves as an example of how XML technologies can be used to create an NFF which has the ability to be reused and regrouped to form new simulation models. The ability to reuse model information in this fashion provides financial incentives that support the use of XML technologies. Additionally, BOM shows that the use of XML based files includes the ability to distribute files via existing internet technologies which already inherently support the XML file type.

2.5.3 SRML

Simulation Reference Markup Language (SRML) is an XML- based markup language designed at Boeing for the representation of simulation models. The language is used in combination with the Simulation Reference Simulator (SR Simulator), also developed at Boeing, to run SRML simulation models. The creation of SRML and SR Simulator has been developed with the goal of leveraging existing internet technologies in order to overcome existing simulation model obstacles. With this goal in mind, SR Simulator was

developed as a plug-in that can be used in any internet enabled application or other supporting application to permit the execution of simulations in many different host applications. Using SR Simulator as a plug-in to a popular internet browser, such as Microsoft Internet Explorer, provides the added benefit of instantly inheriting the additional capability provided by embedded internet communication protocols. This approach enables the running of simulation models on the same computer or across a network in a distributed fashion.

The creation of simulation models using SRML is patterned after the process of web page authoring. SRML syntax includes the use of concepts such as embedded logic defined in javascript, vbscript, or another simulator-supported language, item classes which support hierarchical containment relationships, item quantities for describing large nested numerical items, links for use in referencing external items, and external files to describe simulated item behavior in a separate stand alone file (Reichenthal, 2002).

The development of SRML is an example of how XML is being used to represent simulation model information in an XML file format. Wisely, this implementation is developed to leverage existing proven internet technologies in order to benefit the simulation industry.

2.5.4 Simul8 XML Schema

Founded in 1994, Simul8 Corporation is one of the few DES software vendors that utilize the XML file format to support model information in its flagship software product called

Simul8. SIMUL8 began using the XML file format in November of 2000. Since that time they have matured an XML standard that they would like to be adopted or used as the foundation of a wider standard. The feeling is that the use of the SIMUL8 file format as a foundation would allow a new standard to inherit previous lessons learned from a simulation vendor which would provide inherent advantages when compared with an NFF developed by a non-vendor third party. In an effort to proliferate the Simul8 XML schema with the DES industry, Simul8 provides the complete XML schema (XSD) at the Simul8 web site for review and implementation (Simul8).

Simul8 serves as a unique example of a DES software vendor that already supports the use of XML technologies for maintaining model information. Because of this, Simul8 has completed the first step of what is referred to in this thesis as the two step process. In this way Simul8 sets an example for other vendors to follow in the process of supporting an XML based NFF.

2.6 XML History

Standard Generalized Markup Language (SGML) is a programming language which was created in an attempt to have code which is easily readable by both machines and humans. SGML was designed to include meta-data which is descriptive data that describes the other information contained in the file. File formats which contain meta-data are also referred to as self describing because of the information about information relationship. Originally SGML was successfully used in large document management

systems, however it is best known for its application in creating web pages with a language called Hypertext Markup Language (HTML).

2.6.1 XML Syntax

While still successfully used to create HTML pages, SGML is not well suited for the transfer of data over the internet. HTML is also limited in that it is only intended to be used to display information viewed via a web browser such as Microsoft Internet Explorer. In order to address these limitations, Extensible Markup Language (XML) was created in 1996 by the World Wide Web Consortium (W3C) as a subset of SGML (W3C, 2004). XML syntax has the advantage of being less complex than SGML which makes it easily readable, while also containing meta-data (Hunter, 2000).

2.6.2 XML Schema

Along with XML syntax, W3C created a recommendation for an XML Schema. An XML schema (XSD) is a file that utilizes XML syntax to describe the organization and structure of data within an XML file. This description includes a listing of data items along with data item types and constraints which can be viewed as an outline to understand what data elements an XML file should included. Because an XSD is an outline of the type of information a file should contain, XSD files are also used to validate the contents of an associated XML file (Hunter, 2000).

2.6.3 XML Syntax

XML is not actually a programming language but rather used to describe syntax for the creation of custom languages which meet the XML criteria. It is beyond the scope of this thesis to provide a comprehensive review of XML and XSD syntax; however a brief understanding of the associated syntax is necessary in order to understand the syntax used in mappings later described between the ProModel XSD file and the SDM XSD. This syntax includes the use of tags to provide the previously mentioned meta-data. In the following XML example <student>,<first>, and <last> are meta-data tags used to describe the enclosed information.

```
<student>
        <first>Greg</first>
        <last>Harward</last>
</student>
```

The equivalent XSD file contains similar information that outlines what information is expected in the associated XML file. The following XSD example lists what elements are expected along with the data type of each of the expected elements.

```
<xs:element name = “student”>
        <xs:element name = “first” type = “xs:string”/>
        <xs:element name = “last” type = “xs:string”/>
</xs:element>
```

For the purpose of brevity, the SDM specification uses a simplified syntax for element representation. While contained in the ProModel SDM representation in Appendix A, data type information, contained in the previous example listed as “xs:string”, is not contained in the current SDM specification. This is a limitation in the SDM specification that will need to be addressed before the specification can be adopted as an NFF for the simulation industry. In addition to this simplification, end-tag representations were removed for documentation purposes as they provide no additional clarity. When providing element mappings between a ProModel XSD file and SDM XSD file, a simplified syntax is used such as the following example which illustrates the previous example XSD in a simplified fashion.

<student><first />

Or

<student><last />

2.7 NIST SDM Proliferations

The information available for implementation as well as the case studies that have been performed using the SDM is currently limited. In reviewing the available case studies, it is interesting to note that the NIST SDM specification was originally developed in coordination with Kurt J. Lesker Company. In addition, the shipyard case study is not yet complete or published. Therefore, the only completed independent study focusing on the NIST SDM available at the time of this review is the Boeing case study. This limited exposure could indicate the simulation industries impression of the proposed NIST XML

schema as it is presented in its current draft specification. Regardless, the following is a review of all available case studies related to the NIST SDM.

2.7.1 Boeing Case Study

In 2003, a joint project was concluded between NIST and Boeing Commercial Airplanes in order to test the ability of the SDM Schema proposed by NIST to hold shop data information and test real-world application. An XML data model was populated with all of the required information from an aircraft wing manufacturing operation using the schema proposed by NIST in order to represent the process in a simulation software package. The resultant XML data file was then translated into a Batch Control Language (BCL) file, a QUEST command language, which was executed directly by QUEST software from the DELMIA group. Using this process, the simulation model was run using four different scenarios each time modifying the contents of the source XML file in order to perform "what if" analysis on the simulated system. The simulated scenarios contained asynchronous servers, multi-input-output buffers, bi-directional cranes, labors, processes, and machines on different shifts. The purpose of the simulation model was to determine the feasibility of combining wing operations which are currently defined as separate operations into one larger process. The results from this study concluded that the use of XML provided time savings due to the ease with which the XML file could be modified and reused for multiple simulation model iterations. Because of this, the XML specification proposed by NIST was viewed as a viable NFF that could then be used to generate simulation models (Lu, 2003).

2.7.2 Shipyard Case Study

The manufacturing Simulation and Visualization department of NIST conducted research to analyze the schedule impact of new workloads, evaluate production scenarios, and identify resource problems in the context of a ship building facility. The research consisted of four parts, namely a simulator, control logic, user interface, and internal data management component. The integration of these parts was accomplished using a Run-Time Infrastructure designed for enabling distributed simulation by the Department of Defense. The ProModel simulation engine used in this study was populated with data supplied via a remote user interface using the ProModel COM interface (McLean, 2001). An email to Frank Riddick of NIST on March 10, 2005 confirmed that a study is currently underway which implements the information model for this project using the SDM specification. However this study is not yet complete and he was unable to disclose further information concerning the project due to limitations imposed by the project sponsor.

2.7.3 Kurt J. Lesker Company Prototype

In cooperation with the Kurt J. Lesker Company, Carnegie Mellon University developed a prototype external scheduling system to be used at the Kurt J. Lesker Company. This scheduling system, originally developed with the U.S. Air Force to help manage aircraft and aircraft crews, was modified as required in order to fit the needs of a sample manufacturing domain provided by Kurt J. Lesker Company. A Manufacturing Execution System (MES) was used to provide real-time information to the scheduler that then optimized the execution of jobs within the shop. The interface between the MES

and the scheduling system used XML documents as an NFF which were structured based on the SDM provided by NIST (Lee, 2003).

2.8 Introduction to the NIST Machine Shop Data Model

Within NIST there is a department dedicated to addressing the information interface needs of the U.S. manufacturing community. The System Integration of Manufacturing (SIMA) group works to:

1. Develop information exchange and interface protocols to address manufacturing integration problems.
2. Establish test mechanisms for validating protocols and implementations.
3. Transfer information technology solutions to manufacturing enterprises.

The goal of SIMA is to create a collection of manufacturing specifications which are then reviewed by industry until they mature into an authoritative specification. Each of these specifications is referred to as Initial Manufacturing Exchange Specifications (IMES). One such IMES that SIMA has developed is called the Shop Data Model (SDM) and Interface Specification. This specification is presented in two formats which are Unified Modeling Language (UML) and Extensible Markup Language (XML). The goals that the SDM addresses are:

1. The model will be used to support the integration among a manufacturing execution system, a production scheduling system, and a prototype machine shop simulator.
2. The model will be promoted as a standard data interface for manufacturing simulators (Lee, 2003).

2.8.1 The NIST Shop Data Model and Interface Specification

The NIST Shop Data Model and Interface Specification (SDM) describes the NIST XSD information model and consists of fifteen major categories with four major supporting structures. A brief description of each as described in the Shop Data Model and Interface Specification (McLean, 2003) follows:

2.8.2 Fifteen Major Categories

1. Organizations – Used for organizational structure, contacts and addresses information for the manufacturing organization, its customers and suppliers.

2. Calendars – Identifies the shift schedules that are in effect for a period of time, breaks, and holidays.

3. Resources – All of the resources that may be assigned to tasks in the shop consisting of stations, machines, cranes, employees, tools and fixtures.

4. Skill-Definitions – Lists the skills that an employee may possess and the levels of proficiency associated with those skills.

5. Setup-Definitions – A description of the tool or fixture setups on a particular machine.

6. Operation-Definitions – Definitions of the operations that may be performed at a particular station or group of stations in the machine shop.

7. Maintenance-Definitions – Defines preventive or corrective maintenance to be done on machines or other maintained resources within the shop.

8. Layout – Specifies the location of reference points with the shop, area boundaries, paths, resource, and part object. Also defines pointers to external files that use appropriate graphics standards to further describe these elements.

9. Parts – Elements for part specifications, group technology codes, customers, suppliers, as well as links to bill of material, process plans, drawings, part models, and other references.

10. Bills-of-Material – Cross-references the parts and quantities required in a hierarchical bill-of-materials structure.

11. Inventory – Instances and locations for part, material, tool, and fixture inventory.

12. Procurements – External purchases that have been created to satisfy part inventory and manufacturing requirements.

13. Process-Plans – A set of process plans that are associated with production and support activities for a particular part or parts. Contains <process-plan>, <routing-sheets>, <operation-sheets>.

14. Work – A hierarchy of production orders, jobs, and tasks. Work is maintained as a collection of internal orders for maintenance activities, inventory picking, and tool preparation.

15. Schedules – Planned assignments consisting of mappings of work to resources or resources to work.

2.8.3 Four Major Supporting Data Structures

1. Time-Sheets – Provides a list of individual time sheet elements. A time-sheet is used to log the hours that employee works, the time employee takes off from work, and accrual of leave hours.

2. Probability-Distributions – Specifies distributions that are used to vary processing times, breakdown and repair times, availability of resources, etc.

3. References – Describes the information about the references materials that support or further define the data elements contained with the shop data structure.

4. Units-of Measure – Describes the various units of measure used in the file to represent distance, speed, mass, time duration, currency, etc.

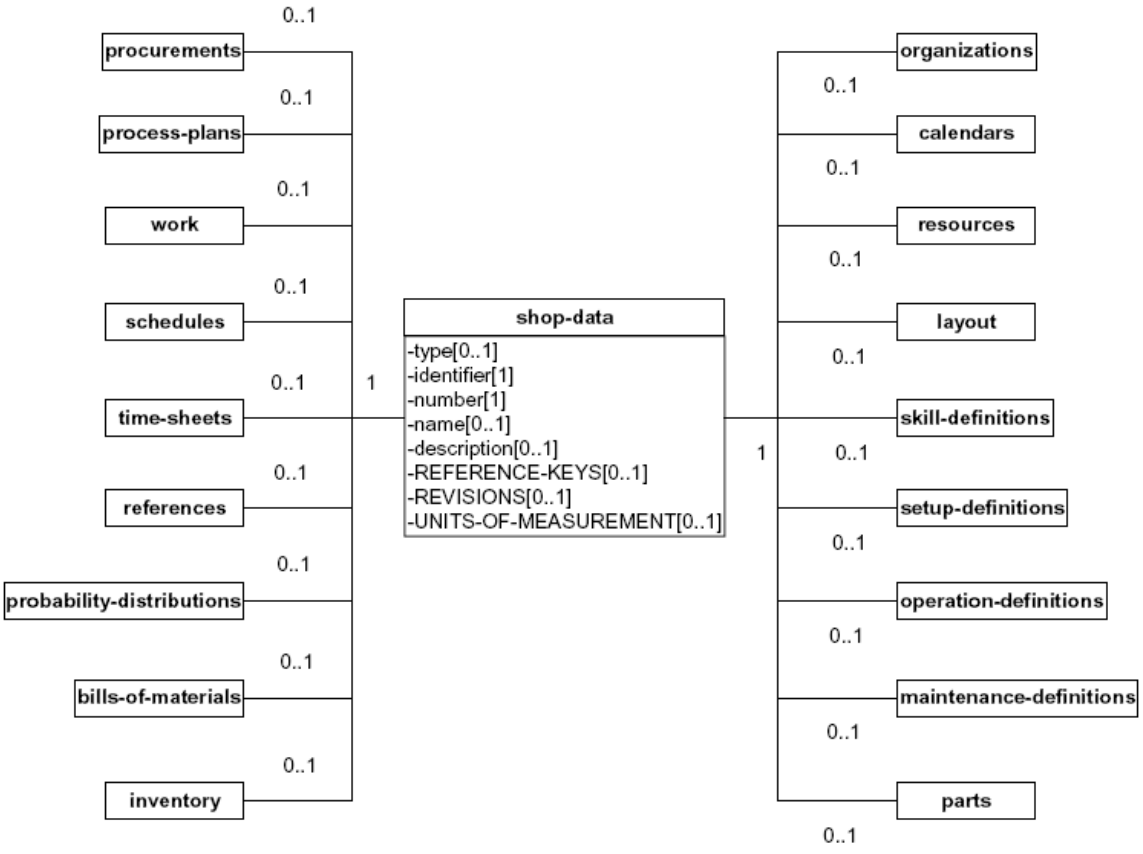

Figure 2. 1 Top Level Diagram of the SDM and Interface Specification (Lee, 2003)

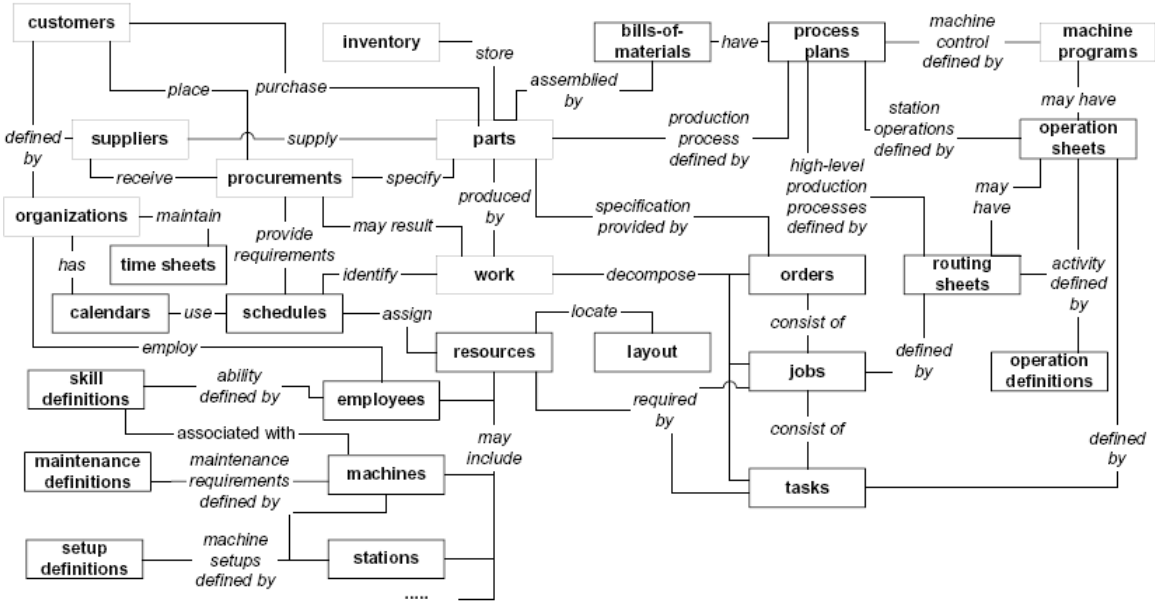

Figure 2. 2 Conceptual Diagram of the SDM and Interface Specification (Lee, 2003)

CHAPTER 3

METHODS AND PROCEDURES

3.1 Introduction

As described previously in chapter 1, the methodology utilized in this thesis consists of a two step process. This two step process matches the efforts that would be required of a simulation vendor attempting to implement native support for the NIST SDM into an existing simulation software product that did not already support an XML file format.

The first step in the two step process is necessitated by the fact that a ProModel XML translator does not currently exist. Therefore, the first step is to manually create an XSD representation of the ProModel information model. Next, the information model is classified into necessary and extraneous elements in order to narrow down the group of available elements to only those that belong in a common information model. This classification is necessary as some ProModel elements are unique and therefore do not belong in an NFF. Next, a mapping comparison between the necessary ProModel XSD elements and the NIST XSD information model elements is performed in order to evaluate compatibility. The results of this comparison (See Appendix C) provide a reference point representing one simulation vendor against which the viability of the NIST SDM as an NFF for the DES industry can be evaluated. This chapter covers each of the two described steps in detail. The results of the ProModel XSD to NIST XSD element comparison are provided in table and chart form in chapter 4.

3.2 Introduction to the ProModel Schema (Step One)

Currently ProModel simulation model information is stored in binary files. While creating a level of security for sensitive model information, the binary file also prohibits direct interoperability with dissimilar file structures. Additionally, ProModel currently does not support a method of exporting the complete contents of the simulation model file in any format other than the binary file.

As the first step in evaluating the information model proposed by NIST, a ProModel XSD file was created so that a direct XSD to XSD comparison could be performed. The ProModel XSD file was created by performing a manual translation of the ProModel information model contained in the C++ programming language source files. The elements that exist in the ProModel information model that are saved to the binary file format in this portion of code were translated into XSD equivalents utilizing Altova's XMLSpy XML editor. The complete contents of the ProModel XSD file can be found in Appendix A.

3.2.1 The ProModel Schema (XSD)

There are currently many software tools available to help with the creation of XML documents. These tools provide the ability to view the XML schema visually in a tree view which displays parent child relationships within expandable branches. The following diagram displays the base parent elements contained in the ProModel XSD. The ProModel XSD contains twenty-two parent categories, each of which contains child

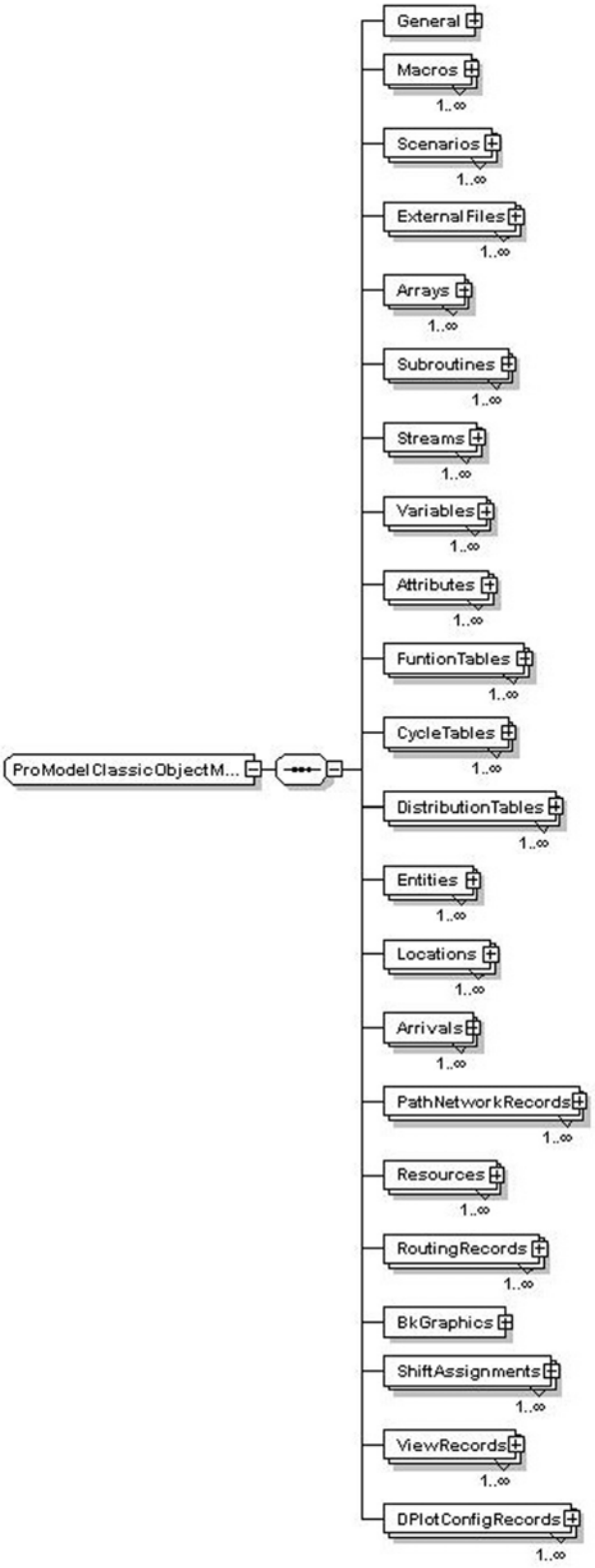

Figure 3. 1 ProModel XSD Parent Element Diagram

sub-categories. A descriptive overview of the contents of each element as described in the ProModel User Guide (ProModel, 2004) follows.

3.2.2 ProModel XSD Element Descriptions

1. General – Contains user defined preferences related to the simulation model which are somewhat static in nature once set for a particular model. Some of these settings are related to User interface (UI) elements such as default sizes, colors, and positions of UI elements.

2. Macros – A placeholder for an often used expression set of statements and functions, or any text that might be used in an expression or logic field.

3. Scenarios – A set of runtime parameters with settings defined by the user. Scenarios allow the end-user to alter model parameters in order to run what-if analysis without changing the actual model. This changing of model parameters is accomplished using Macros which are defined previous.

4. External Files – Files used during the simulation to read data into the simulation model or write data out, as output from the simulation model. Can be used to describe operation times, arrival schedules, shift schedules, and external subroutines.

5. Arrays – A matrix of cells that contain real or integer values used in the model anywhere that a variable can be used.

6. Subroutines – Independent logic blocks that can be called to perform predefined functionality. Subroutines can be called to perform in serial or parallel with the simulation model at runtime. External operations can also be called.

7. Streams – A sequence of independently cycling random numbers. Streams are defined by specifying seed values which set the starting point for values to be sampled from a random number stream.

8. Variables – Global in scope, variables are place holders for holding changing numeric values. They can be integers or real numbers.

9. Attributes – Integer or real number variables associated with locations and entities used to convey additional information.

10. Function Tables (Table Functions) – Provides a method to retrieve a value based on an argument that is passed into a table comprised of dependant and independent values.

11. Cycle Tables (Arrival Cycles) – Tables that supply distributions of arrivals into a simulated system. Cycle tables can be specified as cumulative or non-cumulative.

12. Distribution Tables (User Distributions) – User defined table of values that allows the end user to define their own custom distribution.

13. Entities – Anything that a model processes.

14. Locations – Places in the system where entities are routed for processing, storage, or some other activity or decision making.

15. Arrivals – Set of information describing the entry of entities into the system. This information specifying the number of units, frequency, location, time, and total number of occurrences of each arrival into the system.

16. Path Networks – The pathways that resources and entities are allowed to follow when traveling throughout a system. They are defined as passing or non-passing pathways; movement being specified by speed, distance, or time.

17. Resources – A person, piece of equipment, or some other device used for one or more of the following operations: transporting entities, assisting in performing operations on entities at locations, performing maintenance on locations, or performing maintenance on other resources. Resources are Dynamic or stationary in nature and can have optional associated downtimes.

18. Routings – Define the output for each processing record.

19. Background Graphics – Graphics that are positioned on the layout behind the simulation model. These can be of type BMP, WMF, GIF, or PCX.

20. Shift Assignments (Shifts) – Allows the user to schedule the availability of resources and locations based on shifts and work breaks defined in the shift editor.

21. Views – Allows the user to define positions and zoom factors that can be quickly retrieved to view a portion of a model.

22. Dynamic Plots – Allows the user to select certain metrics for various model elements and observe value changes for those metrics dynamically as the simulation model runs.

3.3 ProModel XSD to NIST SDM Introduction (Step Two)

Each of the ProModel XSD elements created in step one was manually mapped to the equivalent element in the NIST XSD as part of step two. The mapping of each element was performed in a manner that would maintain the concept of each element as described in the 2004 ProModel User Guide (ProModel, 2004) in correlation with the NIST SDM specification (McLean,2003). After the mapping process was completed each of the elements was then organized according to its ability to support the concept and content of the original ProModel simulation model using the criteria next outlined.

Supported – The data element has a corresponding NIST XSD information model element which is able to preserve the original information and meaning of the associated ProModel data element in a logical schema orientation.

Unsupported Necessary – This information is necessary as an input for ProModel in order to run a ProModel simulation as designed and gather statistics to generate meaningful output reports. These items may be supported in concept, but the current NIST SDM specification does not support the element in content.

Unsupported Unnecessary – This information is not required in order to run a ProModel simulation model. These are typically User Interface (UI) elements which deal with how an item appears to the end user at simulation run-time. The element is unnecessary if the exclusion the element does not adversely affect the statistical validity of the model output. Additionally, this category also contains legacy fields which are no longer used which contain the description "Not currently implemented in ProModel". While not necessary, elements with this description are currently found in the ProModel source code and are included as a help for additional future study that may use this thesis as a guide. Knowing this helps to clarify points where confusion may occur.

3.3.1 Comparison Mapping Assumptions

In performing the mapping comparison the following assumptions were made.

The current NIST XSD information model described in the NIST SDM specification provided by NIST does not include data type information. This is a known limitation and is a planned change to include in the next release of the SDM specification according to an email dated February 28, 2005 from Frank Riddick of NIST. Data type information is used to specify the specific programming element type that will be used to represent each

element. Conversely, the ProModel schema was created with data type information that matches the current implementation of ProModel data structures. Because of this inequality, addressing issues related to data type comparisons are omitted in most cases.

In the simulation industry, the creation and execution of simulation models have been enhanced with the use of visual elements that represent various simulation objects. These graphical elements help to define model elements statically at design time and dynamically during simulation run-time. In terms of graphical representations, the NIST SDM specification remains incomplete in some areas where graphics would be applied particularly when dealing with elements of the layout. For this reason not all of the graphical elements supported in ProModel are mapped. For classification, the graphical elements are listed as unnecessary because the use of graphics does not affect the statistical validity of simulation models which makes them not a support requirement for a standard information model or associated XML NFF.

The perspective of the mapping performed in this study is to fit ProModel model file elements into the NIST XSD information model as defined by the NIST SDM specification. The mapping does not force the assumption that the resultant XSD file is a complete and valid file according to the requirements of the SDM specification. Once the SDM specification is complete, a translator would need to be created using the information contained herein to properly confirm the validity of the ProModel XSD and associated XML NFF file.

A pointer in the C programming language is a data element that refers to another data element. The representation of pointers contained in the ProModel source files is represented in the ProModel schema as an element of data type long with the term “ID” appended to the element name. While technically a pointer in the C programming language is a data type long, this is an intentional oversimplification used in this thesis to represent a pointer to an existing structure that is covered in another section of the review. This simplification was made in order to avoid unnecessary complication while maintaining the focus of this thesis. When mapping this type of element, the section that is being referenced is mentioned.

This thesis focuses on evaluating the parent level of the ProModel XSD “tree” and their immediate children. This approach provides evaluation for the majority of the elements and covers all necessary concepts that need to be mapped. Once these base elements are supported it would then be possible to make further evaluations within the complete information element tree based on what support is given to base ProModel XSD elements in the NIST XSD. Until this occurs, it is appropriate to simply report the support provided for ProModel base elements and their immediate children in order to maintain the focus of this thesis.

3.3.2 The ProModel Build Menu

The ProModel software application contains a menu bar item in the application window titled Build. The Build menu provides the user with the essential input options required to create and simulate a model using ProModel software. By design, this menu is

organized in a fashion that guides the user of the software through the simulation model building process by specifying the most vital simulation options first. The Build menu reflects this concept by listing these most vital options at the top of the menu and conversely lists items of lesser importance toward the bottom of the menu. The Build menu therefore serves as a simple visual priority guide to quickly outline which elements in ProModel are necessary in order to build a ProModel simulation model. The classification of ProModel elements is important in order to differentiate between vital and non-vital simulation model elements when considering the common nature of NFF contents.

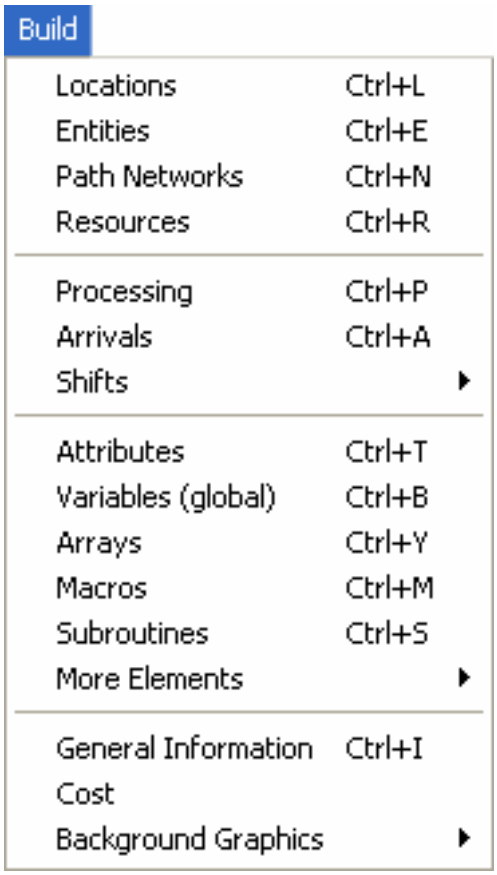

Figure 3. 2 The ProModel Build Menu

3.3.3 The Classification of Simulation Elements

The first two sections of the ProModel Build menu contain four components that are the vital elements required to build a ProModel simulation model. These elements are Locations, Entities, Processing, and Arrivals. Because they are vital in nature, these elements need to be supported in any proposed common information model. For this reason, the mapping of these elements is addressed first.

Second, the mapping of the remaining components in these first two menu sections as well as three other common programming elements will be addressed. These elements include Path Networks, Resources, Shifts, Scenarios, External Files, and Streams. While not absolutely vital for the creation of a ProModel simulation model, these elements are common among simulation applications that compete with ProModel in the same market space. Because they are common in nature, these elements need to be supported in a common information model.

The third section in the Build menu contains elements that are basic software programming elements. This is to say that these elements and their associated functionality are common among the languages that are used to develop software applications such as the C programming language. These elements include Attributes, Variables, Arrays, Macros, and Subroutines and are implemented in order to support functionality that enables ProModel software to function as a general purpose simulation tool. These elements are common among simulation vendors that compete with ProModel so they also need to be supported in a common information model.

Last, the mapping of the remaining ProModel elements is addressed. These elements consist of features that are related to the visual representation of ProModel models. These items are unique to ProModel software or are used to define graphical elements which make the support of the items non-essential to the creation of a simulation model and unnecessary common information model elements.

3.3.4 ProModel XSD to NIST SDM Mapping Syntax

The following section lists each of the necessary ProModel base elements. Following each element is a description of the information that the element holds and in some cases special issues involved in mapping a particular element are discussed. When an element is not supported by the SDM, suggestions for possible implementation are given. Supported mappings and suggested mappings are provided using a simplified XML syntax as described in the XML Syntax section of chapter 2. A comprehensive listing of ProModel elements which includes NIST XSD syntax equivalencies for element comparisons is provided in Appendix B.

3.4 The Four Vital Simulation Elements

3.4.1 Locations

The SDM contains <resources><stations><station><station status><work-assignments/><work-assignment><work-keys/><maintenance-order>< maintenance-order-definition><maintenance-definition-key> which is used to describe preventive or corrective maintenance work to be performed on locations. Within this element a

<periodicity/> element is used to describe how often this operation is to be performed. At the time of this review this element has no definition for units. If this element could be filled or mapped to logic then it may be possible to represent LocationRecordClockDowntimes, LocationRecordUsageDowntimes, and LocationRecordCyclicDowntimes.

Additionally, the SDM contains <probability-distributions> which contain distributions used to vary time related activities such as repair times and availability. This element could help to define the frequency element found within LocationRecordClockDowntimes, LocationRecordUsageDowntimes, and LocationRecordCyclicDowntimes by accepting a distribution type. This element could be accessed as follows <resources><stations><station><probability-distribution-key><probability-distribution>.

3.4.2 Entities

The ProModel schema for each entity record consists of twelve major categories. Of these twelve main categories, seven are represented in the NIST schema under the main parts category as shown in the following mappings.

3.4.3 Routing Records (Processing)

A routing record in ProModel defines the processing for a particular entity at a given location using operation logic statements. After the processing is completed, the entity is then routed to a destination location using a specified rule. Because these operations are

triggered by the arrival of an entity, the simulation can be thought of as “entity driven”. The paradigm outlined in the NIST SDM is not entity driven, but instead is designed to describe process flow though a machine shop. This approach works well for the description of process steps through a system. Because of the difference in paradigms, mapping in this section do not fit well with any section of the SDM.

3.4.4 Arrivals

In ProModel, any time a new item is introduced into the system; this item is called an arrival. The arrivals in ProModel are defined by specifying a schedule that a particular entity arrives at a location. In the SDM arrivals can be mapped using the <procurements><procurement><procurement-definition>. However, the paradigm for defining all the arrival information contained in ProModel does not directly map into one SDM element.

3.5 The Six Common Simulation Elements

3.5.1 Path Networks

The equivalent structure in the Shop Data Model for a ProModel Path Network is found in <layout><paths><path>. However at this time of review the <layout> element was listed as “still under development”. Therefore it is unclear what each of the elements and sub elements inside the <layout> element represent as a formal description remains to be supplied. An email to Frank Riddick of NIST confirmed that this section is still under

development and would be replaced with a new structure aligned more closely with the concepts contained in the SDX specification.

Even though it is not possible to map all of the elements with a great degree of confidence, an attempt at mapping the high level elements has been made given what information has been provided with the understanding that the foundation of this information is subject to change.

3.5.2 Resources

A resource in ProModel is a person, piece of equipment, or other device used to transport entities, assist in performing operations on entities at locations, or perform maintenance on other resources or locations. The SDM lists resources as machines, stations (ProModel locations), cranes, employees, tool-catalog, or fixture catalog. The closest mapping of the ProModel resource is the SDM <machines> element. At the time of this review the SDM doesn't support the concept of a mobile resource, but should be supported in the next release according to an email from Frank Riddick of NIST. This shortcoming limits the ability to map some ProModel elements that are mobile in nature. Additionally some ProModel resource elements interface with the layout. Because the NIST layout specification is still in progress these items do not have an associated mapping.

3.5.3 Shifts (Shift Assignments)

The concept of shift (also called breaks) in ProModel applies to both locations and resources. Shift are defined in ProModel shift files (*.sft) which are plain text files. Once defined these text files are assigned to location and/or resources to provide a means for the location/resource to go on and off shift. Shifts also can contain associated logic that is executed immediately before and after going off shift. The SDM mapping for ShiftAssignments is found in <Calendars><Calendar><Shift-Schedule><Shifts> The relationship between ProModel locations and resources with shift files is obtained in the SDM via schedules which assign resources which may include machines or stations.

3.5.4 Scenarios

Scenarios allow the end-user to alter model parameters through the use of macros in order to simulate multiple what-if combinations aimed at optimizing items of interest. The ability to run multiple scenarios in this manner is a large contributor to the value the end user obtains from this type of simulation. The SDM does not contain any structure for persisting scenario information. This would be a necessary element in any NFF supporting the discrete-event simulation industry.

3.5.5 External Files

External Files are references to remote files used during the simulation to read data into the simulation model or write data out. The concept of accessing external files in the SDM is supported through the existing <reference-key> element which can be used wherever an external file reference is needed.

3.5.6 Streams

A sequence of independently cycling random numbers; streams are used in simulation products in order to allow independent control of statistical variability within the simulation through the use of seeds values. The concept of storing seed or stream information is currently not supported within the SDM. However, it may not make sense to support this information in an NFF because different simulation products could use different streams of differed sizes. The difference is size would create the possibility of storing invalid seed values unless the representation of streams is also standardized across simulation products. These values could be stored in list form or a methodology similar to the one used to store arrays as described in the next section could be used.

3.6 The Five Basic Software Programming Elements

These five elements are used to provide the ProModel model builder with functionality that is common to standard computer programming languages. ProModel provides these elements grouped together by type rather than by application. The SDM does not contain any sections for the description of generic global functionality such as these provide. Given that NIST is planning to implement the ability to define logic for models, it would make sense to address the implementation of these elements at the same time as both features utilize traditional programming paradigms. This type of functionality could be added as a global option, or the functionality could be added locally to each element given the intended functionality.

3.6.1 Attributes

Attributes provide the functionality to assign a real or integer data type value to a ProModel entity or location. The attribute can then be accessed at any time during the simulation to provide decision making ability which has a scope limited to a particular entity or location. This equivalent functionality is analogous to an object property found in the C programming language. In the SDM this type of functionality could be added to the ProModel entities or locations equivalents such as <parts><part><attribute/> or <resources><stations><station><attribute/> respectively.

3.6.2 Variables

Variables contain real or integer values that are accessed by ID. These values are global in nature, so they can be accessed anywhere within model decision making logic and are not tied to any one particular element. The naming and use of variables is equivalent to that found in C programming.

3.6.3 Arrays

Arrays contain a matrix of cells which contain integer or real number values which are accessed by index numbers. A common spreadsheet application is a common example of a two dimensional array which uses row and column specifications to access information. Arrays in ProModel can have multiple dimensions and can be populated with inputs from files which are external to the ProModel application. ProModel arrays are global in nature so they can be accessed globally within the simulation model by ID. The naming and use of arrays found here is equivalent to that found in C programming.

3.6.4 Macros

A macro is a placeholder for a commonly used expression. These placeholders can be used anywhere in the simulation model and are replaced when evaluated with the corresponding macro value. The naming and use of macros found here is equivalent to that found in C programming.

3.6.5 Subroutines

Subroutines contain user defined blocks of code that can be called at anytime and optionally return a value. Subroutines can also accept parameters as inputs. The naming and use of subroutines found here is equivalent to the concept of functions found in the C programming language. The representation of subroutines would be accomplished using the same syntax method that is used to describe decision logic, the caveat being that, by definition, subroutines are portions of decision logic which can be accessed from other elements within the model structure in a global fashion.

3.7 Nonessential Elements

The remaining elements not yet covered include General, FunctionTables, CycleTables, DistributionTables, ViewRecords, DPlotConfigRecords, and BkGraphics. A description of each of these elements is provided in the previous section. These items are unique to ProModel software which makes their support in an NFF unnecessary. They are however included in the ProModel schema for the purpose of providing complete representation.

CHAPTER 4

RESULTS AND ANALYSIS

4.1 Mapping Table

Using the required model elements from the ProModel XSD in Appendix A, values were mapped to the equivalent NIST XSD elements as defined in the NIST SDM. Table 4.2 displays the results of the mapping process with a column that describes the support for each element. When not supported, elements are categorized into one of four incompatibility classifications. The four unsupported element categories are: Basic Data Elements, Decision Logic, Basic Software Programming Elements, and Layout Related Elements. These categories are identified with the support classifications “Basic”, “Logic”, “Program”, and “Layout” respectively. A more comprehensive explanation of each of these categories follows.

An expanded version of Table 4.2 is provided in Appendix C. This expanded table provides the equivalent NIST XSD syntax representation for supported ProModel XSD elements. Additionally, if possible a mapping is provided for unsupported elements to serve as a suggestion for future implementation. The table also contains two support classifications which are not included in Table 4.2 to provide clarity. The support classification “*” refers to those elements that utilize a pointer in the C programming language. The comparison mapping descriptions provided in appendix B of elements marked “Caveat” should be consulted for additional explanations. These classifications

are provided in the event that future work is performed using this information as a foundation.

Table 4. 1 ProModel to SDM XSD Mapping Comparison Summary Table Legend

Support	Description
Y / N	Is element supported? (Yes or No)
Basic	This element is classified as a Basic Data Element.
Logic	This element contains Decision Logic.
Program	This element is a Basic Software Programming Element.
Layout	This element requires support from the NIST Layout Element.

Table 4. 2 ProModel to SDM XSD Mapping Comparison Summary Table

	Support Classification				
Element Type	**Yes**	**No-Basic**	**No-Logic**	**No-Program**	**No-Layout**
Locations	7	7	5		
Entities	7	2			
Routing Records	3	12	4		
Arrivals	4	8	1		
Path Networks	3	2			4
Resources	5	15	2		5
Shifts	3	5	4		
Scenarios			1		
External Files	1				
Streams			1		
Attributes				1	
Variables				1	
Arrays				1	
Macros				1	
Subroutines				1	

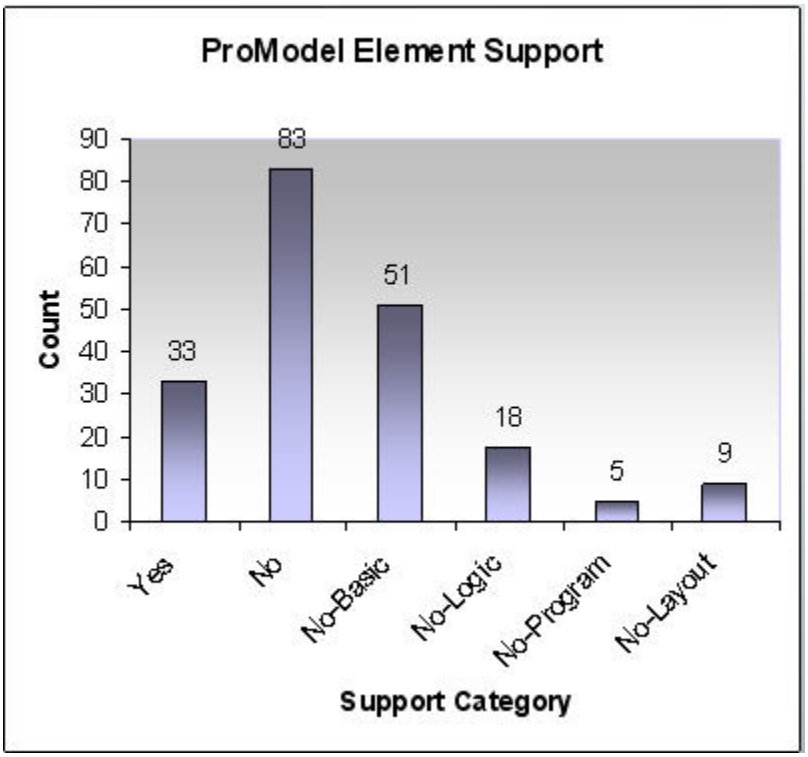

Figure 4. 1 ProModel to SDM XSD Mapping Comparison Summary Chart

4.2 Statistics of Transferability

In attempting to map the ProModel XSD elements which are classified as necessary and meet the criteria as valid common elements, 28.5% of the elements currently have valid representation contained in the current NIST XSD information model. This means that 71.5% of the elements do not currently map to the NIST XSD. Therefore, currently the information model contained in the NIST SDM does not provide adequate support for the necessary ProModel elements in order to serve as an NFF for this test case.

4.3 The Four Unsupported Element Categories

This section categorizes the eighty-three ProModel elements that did not map over to the NIST XSD into four categories. In correspondence with Frank Riddick of NIST, he

stated that NIST is currently aware of the issues associated with three of the categories listed and that representation is expected for these items in the next release of the SDM specification.

4.3.1 Basic Data Elements

Fifty-two of the ProModel unsupported elements fall into the category of Base Data Elements. While the majority of unsupported elements are of this type, the suggested implementation provided shows that support for elements in this category are possible and can be accomplished with the addition of elements which contain basic XML data types. These data types include the optional implementation of additional enumerations to the NIST XSD information model in order to provide clarity.

4.3.2 Decision Logic

Sixteen of the ProModel elements were not supported due to the lack of support for decision logic. According to Frank Riddick, NIST plans to support decision logic in the next release of the SDM specification. The exact format for supporting such logic remains to be determined. The representation for this logic can be achieved utilizing several different approaches which include the use of vbscript, javascript, or other machine readable languages embedded directly into XML tags as described in the referenced article *Reintroducing Web-Based Simulation* (Reichenthal, 2002). Additionally, a simplified approach could be used to supply model logic in the form of strings that would then be parsed out and compiled by the host application after being read. This method is more ambiguous, which is not in keeping with the goal of XML

syntax. However, the string method is more flexible in that it would support any external language desired for use in describing model logic. A third option would be for the XML document to reference external files which contain precompiled code. This option would save the time associated with parsing and recompiling machine readable code while at the same time simplifying the XML syntax.

4.3.3 Basic Software Programming Elements

Five of the unsupported elements are what have previously been referred to as Basic Software Programming Elements. These programming elements are used to provide general functionality which is not related to any particular model paradigm and serve to open up possibilities for additional flexibility for simulation applications. While adding decision logic, it would make sense to also add the five Basic Programming Elements to support items in this category as they are commonly used in cooperation with decision logic. The five Basic Software Programming Elements consist of Variables, Attributes, Arrays, Macros, and Subroutines. Variables and Attributes can be represented in XML utilizing standard XML data types. Arrays and Macros are lookup tables that can be represented using XML syntax as described by Butak (Butak, 2005). Subroutines are isolated groups of decision logic that can be represented in XML syntax utilizing the same methodology as previously described by Reichenthal (Reichenthal, 2002).

4.3.4 Layout Related Elements

Eight of the unsupported elements are related to the unfinished state of the Layout portion of the SDM specification. According to an email from Frank Riddick of NIST, the

Layout portion was under development when the decision was made internally to change the approach and align the work more closely with the paradigm used in the SDX file format. In order to support ProModel, the new specification would need to support the concept of nodes. ProModel nodes serve as interface points to path networks and are also used for positioning elements at various times during simulation runs. In order to support the ProModel methodology, the concept of nodes or some equivalent structure needs to be supported.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

This section begins by addressing the four thesis questions posed in chapter 1 and concludes by presenting other secondary conclusions gained through the course of this thesis.

5.1.1 Thesis Question 1

There were four questions posed in the original thesis statement. The first of these questions conjectures the ability of XML syntax to represent the elements associated with a manufacturing DES model. The process of creating the ProModel XSD (see Appendix A) and also mapping model elements from the ProModel paradigm over to the NIST SDM paradigm shows by example that XML did not pose any syntax related limitation which would invalidate the ability of XML syntax to represent a common information model or associated XML NFF candidate. The elements for which there is currently no representation within the NIST XSD were categorized into four categories. As described in the evaluation of each of the four sections, each of these categories can also be represented using the XML syntax barring the Layout portion which lacks completion.

5.1.2 Thesis Questions 2 and 3

The second and third queries consider the ability of the NIST XSD to support ProModel simulation model elements. The results of performing the process of mapping ProModel model elements to NIST XSD elements are that less than a third of the ProModel elements are currently supported by the NIST SDM specification. Therefore, utilizing ProModel as the test case, the NIST XSD does not support all of the required elements to effectively serve as an NFF.

5.1.3 Thesis Question 4

When drawing conclusions based on the results of this study, it is possible in hindsight to understand the root cause of some of the challenges simulation vendors might encounter when mapping the model elements from the ProModel XSD to the NIST XSD. The origin of some of these challenges can be found in understanding the approach that was originally used when the NIST SDM schemas was initially created.

The NIST XSD information model as described in the SDM was originally designed as an external scheduling system for the U.S. Air Force. As explained in chapter 3, this initial implementation was then modified to meet the needs of a manufacturing facility at the Kurt J. Lesker Company. This modification included integration with a MES system in order to optimize jobs within the manufacturing environment. In addition to the application of the original schema to MES and production scheduling systems, the XSD is now being promoted for use as a data interface for manufacturing simulators (Lee, 2003). Because the XSD was not originally designed to include the data requirements of

simulation models, the XSD is not well suited for representing the data requirements of ProModel simulation models. Currently, the paradigm presented in the SDM does not contain a dedicated simulation component which causes mapping disconnects when mapping to the ProModel paradigm.

This paradigm difference causes some elements to be extraneous in nature as they do not specifically apply to simulation. Organizations, skill-definitions, and bills-of-materials are three parent elements that exist in the SDM specification for which there is not an equivalent element mapping in the ProModel XSD. Additionally, the SDM specification utilizes concepts which are based on the manufacturing engineering principle group technology, contained in the elements <station groups>, <part groups>, and <machine groups>. ProModel does not contain element classifications explicitly based on group technology.

The inverse is also true in that ProModel simulation software contains model elements which are not supported conceptually within the NIST XSD. These items include those that were classified previously as non-essential elements. Logically, these items also include the items for which there are not equivalent NIST XSD mappings.

As explained, because the NIST SDM has roots in addressing the manufacturing organization, the approach of NIST is to describe the manufacturing operation in terms of a group of scheduled sequential steps that are to occur within an organized process described using a “what to do next” approach. This approach is different from the

ProModel approach which describes where an entity is to go after processing at a location. This approach can be described as a “where to go next” approach and conceptually affords for elements within the system to act in a more dynamically autonomous fashion. This paradigm difference necessitated the translation of ProModel XSD element concepts over to equivalent NIST XSD elements in order to overcome the disparity between element meanings.

The best example of this conceptual difference is contained within the mapping section which describes the translation of NIST XSD elements over to ProModel Routing Records. In this conceptual section of the mapping, the NIST SDM defines Jobs and Tasks using Routing Sheets and Operation Sheets respectively. Routing Sheets and Operation Sheets are defined within a parent Process Plan which is generated from a Bill of Materials. The process plan is created by specifying a sequence of operations utilizing a “what to do next” approach. The equivalent elements of an NIST XSD Routing Sheet and an NIST XSD Operation Sheet as described in the ProModel User Guide are contained within a ProModel Routing Record which contains routing logic and a ProModel Process Record which contains Operation Logic. These sections together are used to define Entity Processing which defines “where to go next” from the entities perspective.

5.2 Secondary Conclusions

5.2.1 The Two Step Process

The process of mapping simulation model elements from a proprietary information model over to an XML based NFF in this thesis reveals that a two step process is necessary. The first step in the transformation process is to convert the proprietary ProModel binary file into an XSD file written with XML syntax. This step removes the proprietary nature of the binary source files and removes the file language incompatibility by converting the file contents into human readable XML files. This first step was accomplished and resulted in the creation of the first ProModel XSD presented in chapter 3 (see Appendix A). The next step in the two step process of transformation is to map model elements contained in the resultant ProModel XSD file to the equivalent NIST XSD elements. Syntactically, this is an XML to XML transformation which serves to remove file schema differences. This step was accomplished and the results were presented in the mapping table in chapter 4. While it would be possible to perform both of these steps concurrently, conceptually these two steps would still be required by any simulation vendor who desired to support the NIST SDM.

5.2.2 ProModel Steps to XML NFF

This thesis presents the first ProModel XSD file. The creation of this file accomplishes the initial first step of the two step process for ProModel software. This step is critical in order to arrive at an industry supported XML-based information model as it reveals the ProModel object model in a human readable format which can be evaluated more

definitively rather than conceptually. In addition, step two has been approached by addressing what elements would need to be changed in order to support ProModel elements in the NIST SDM.

5.2.3 NIST SDM Traction

Currently, most of DES software vendors do not support output in the XML file format, but instead utilize binary file types which are inaccessible (Lionheart, 2005). This inaccessibility creates difficulty when attempting to evaluate the specific needs of the DES industry in terms of XSD structure and required elements in an attempt to create a viable information model and resultant XML based NFF. While this file dissimilarity is exactly what NIST is attempting to overcome with the creation of a common information model, it also currently serves to hinder progress on development of the same.

To overcome the problem of dissimilar file types and make progress toward a common information model and XML based NFF, software vendors need to complete the first step in the two step process. Once more vendors completed the first step, it would be possible for vendors to share files and experiment with XML translators which are readily available given the maturity of XML technology. As explained, this step would remove the proprietary nature of the binary file and expose vendor schema and element structure for reuse and evaluation regardless of the direct taken by the SDM. Then it would be possible for NIST to evaluate vendor XML based files to gather common elements for the creation of a viable common XSD. Other than the ProModel XSD proposed in this thesis, Simul8 Corporation is the only example of a discrete-event simulation software

company that has executed this first step. I am of the opinion that more focus should be placed on this first step within the DES industry. The completion of this first step by simulation vendors would then create an environment within which an XML based NFF could be developed based on the needs of the industry. This in turn would generate more industry support for the venture.

Without having the ability to generate an XSD based explicitly upon the consensus of DES vendors, it is assumed that the adoption of an XSD for the simulation industry would also require the adoption of a modeling paradigm. This thesis reveals that the approach taken by ProModel is different than the one currently being explorer by NIST. If the adoption and support of common information model requires the adoption of a new modeling paradigm which does not incorporate elements to support existing vendor paradigms, it is unlikely that the majority of simulation vendors will adopt such, as it would require reengineering software packages to conform to a new modeling paradigm.

5.2.4 DES Vender Value

The results presented in this thesis provide value to DES vendors in two areas.

1. Appendix A provides an XML schema which outlines how ProModel organizes the elements required to support the creation of a DES model using XML syntax. The presentation of this schema affords simulation vendors the opportunity to evaluate the organization of the presented simulation elements and use the XML schema as a starting point for the creation of a similar XML schema for an unrelated DES product.

2. The results presented in chapter 4 and the mappings presented in Appendix B outline the ProModel element support that is currently provided by the NIST XSD. These results along with the description of the steps required to convert an existing information model over to the information model presented by NIST provide a reference point which can be used to make an informed decision concerning the amount of work required to support the proposed NIST NFF specification.

5.3 Suggestions for Future Study

The following recommendations are made for future research that would extend the concepts that were explored in this thesis.

5.3.1 Identify Commonalities

This thesis explores the process of mapping ProModel data elements over to the proposed NIST information model as described in the SDM. In performing this evaluation, ProModel elements were classified into categories based on necessity. A similar study could be performed to evaluate the common concepts and elements that exist within the top simulation vendors in the DES market. This type of study would include the specification of data type information for each of the common elements as well as the modeling paradigms utilized by each vendor. The study would also include an evaluation aimed at specifying the requirements of the simulation industry for supporting logic statements. This type of evaluation is necessary in order to arrive at a common method of storing logic statements that meets the needs of the majority of the DES industry. The

results of this type of study would create a common group of elements and concepts to form a foundation upon which the creation of a common XSD information model and associated XML based NFF could be designed.

5.3.2 Creation of ProModel XML Translator

This thesis is unique in that it explains each of the main elements contained in the ProModel binary file at the binary level. From this evaluation a ProModel XSD was created (see Appendix A). The ProModel XSD file that was created could be used to construct an XML translator. This translator could then be used to automatically export the ProModel information model to equivalent XML files as well as import XML files back into ProModel in order to validate the ProModel XSD presented in this paper.

5.3.3 Translator Performance Evaluation

One of the advantages of the binary file is that the process of importing and exporting files of this type is comparatively fast. While XML has the advantage of being humanly readable, the addition of tags used to create XML files also means that files of this type are exponentially larger than the equivalent binary file. Once the creation of an XML translator as described above is complete, performance evaluation tests could be performed in order to determine the speed differences associated with reading in XML and binary file types.

5.3.4 Validation of SDM Mappings

While the content of the ProModel information model might be mapped over to the NIST XSD in the future using the mappings suggested in Appendix B, the validation of these mappings should be verified. This would include validation that the resultant NIST XSD and XML based files contain all of the elements necessary to form a valid NIST XSD file according to the SDM specification. Also, this file should be validated against a different application which supports an import of XML files created using the NIST XSD. Currently such an application doesn't exist, but this thesis could serve as a basis for such work.

5.3.5 Listing of NIST Recommendations

In summary the following list is provided as recommendations for NIST in support of future efforts.

1. Focus first on initiatives which promote the conversion of binary files into XML files generating support within the industry for XML technology. Then use these XML files to validate the SDM initiative and proceed with an XML based NFF specification based on industry XML based input.
2. From XML based industry files, generate a group of common elements and concept which belong in a common information model. This includes support for existing software vendor paradigms.
3. Develop a section within the SDM explicitly dedicated to simulation model elements and concepts.

4. Complete the sections of the SDM which currently are not finished in order to provide a reference point for further work. Minimally, this includes the completion of the layout section and support for model logic. In addition, the completion of the "Example Usage" section would help to clarify specification details.

5.3.6 NIST SDM Specification Limitations

The conceptual model for the NIST specification as pictured in chapter 3 provides a clear graphical representation of the NIST paradigm. While the tie between objects is clear in the conceptual model, the SDM specification is unclear in the explanation of XML syntax used to create the ties between elements. Currently the specification contains a Section D titled "Example Usage" for each element. It is assumed the intended use of this section is for providing syntax examples for each element. However, currently Section D for each element is blank which serves as a limitation. The completion of the example usage section would provide better understanding of the intended application of the NIST SDM by example, as well as clarify how the ties between objects is to be accomplished syntactically.

REFERENCES

Banks, B., Carson II, J., & Nelson, B. (1996). *Discrete-Event System Simulation*. New Jersey: Prentice-Hall.

Boulton, C (2004). *Group Drafts First Spec for 3D CAD* Retrieved 5/1/05, from Internetnews.com website: http://www.internetnews.com/dev-news/article.php/3362511.

Butak, R (2005). *Web services programming tips and tricks: Array Gotcha—Null Array vs. Empty Array* Retrieved 5/28/05, IBM website: http://www-128.ibm.com/developerworks/xml/library/ws-array.

Chatfield, D., Harrison, T., & Hayya, J. (2004). XML-Based Supply Chain Simulation Modeling. In *Proceedings of the 2004 Winter Simulation Conference*. ed. R.G. Ingallis, M.D.Rossetti, J.S. Smith, and B.A. Peters.

Devguru (2004). *A Beginners Guide to Creating and Displaying Your First XML*. Retrieved September 8, 2004 From Devguru website: http://www.devguru.com/Features/tutorials/XML/beginning_xml.html.

Gustavson P., & Chase, T. (2004). Using XML and BOMS to Rapidly Compose Simulations and Simulation Environments. In *Proceedings of the 2004 Winter Simulation Conference*. ed. R.G. Ingallis, M.D.Rossetti, J.S. Smith, and B.A. Peters

Harrell, C., Bateman, R. E., Gogg, T. J., & Mott, J. R. A. (1997). *System Improvement Using Simulation.* ProModel Corporation. (p.1-20).

Harrell, C., Ghosh, B., & Bowden, R. (2000). *Simulation using PROMODEL*. New York: McGraw-Hill.

Hunter, D. (2000). *Beginning XML*. Wrox Press Ltd.

Kilgore, R. A. (2001). Open Source Simulation Modeling Language (SML). In *Proceedings of the 2001 Winter Simulation Conference*, ed. B.A. Peters, J.S. Smith, D.J. Medeiros, and M.W. Rohrer.

Kim, K. (2003) *The Simulation Data Exchange (SDX) File Format: A Useful Neutral File format (NFF) For Discrete-Event Simulation.* Thesis Brigham Young University, 2003.

Lee, T., McLean, C., & Shao, G. (2003). A Neutral Information Model for Simulating Machine Shop Operations. In *Proceedings of the 2003 Winter Simulation Conference*. ed. S. Chick, P. J. Sanchez, D. Ferrin, and D.J. Morrice.

Linux (n.d.). www.linux.org.

Lionheart Publishing Inc (2005). *Simulation Software Survey.* Retrieved February 8, 2005 From Lionheart Publishing Inc website: http://www.lionhrtpub.com/orms/surveys/Simulation/Simulation6.html.

Lu, R., & Qiao, G. (2003). NIST XML Simulation Interface Specification at Boeing: A Case Study, In *Proceedings of the 2003 Winter Simulation Conference*, ed. S. Chick, P. J. Sanchez, D. Ferrin, and D. J. Morrice.

McLean, C., & Shao, G. (2001). Simulation of Shipbuilding Operations. In *Proceedings of the 2001 Winter Simulation Conference*, ed. B.A. Peters, J.S. Smith, D.J. Medeiros, and M.W. Rohrer.

McLean, C., & Leong, S. (2002). A Framework for Standard Modular Simulation. In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yucesan, C. –H. Chen, J.L. Snowdon, and J. M. Charnes.

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

McLean, C., Leong, S., Harrell, C., Zimmerman, P., & Lu, R. (2003). Simulation Standards: Current Status, Needs, and Future Directions, In *Proceedings of the 2003 Winter Simulation Conference*, ed. S. Chick, P.J. Sanchez, D. Ferrin, and D. J. Morrice.

Microsoft (2004). *Code Generation in the .NET Framework Using XML Schema* Retrieved August 4, 2004 From Microsoft MSDN website: http://msdn.microsoft.com/xml/default.aspx?pull=/library/en-us/dnxmlnet/html/xsdcodegen.asp.

Microsoft (2005). *Microsoft "Office 12" XML File Formats to Give Customers Improved Data Interoperability and Dramatically Smaller File Sizes* Retrieved June 3, 2005 From Microsoft website: http://www.microsoft.com/presspass/press/2005/jun05/06-01OfficeXMLFormatPR.mspx.

NIST (n.d.). http://www.mel.nist.gov.

ProModel Corporation. (2004). *ProModel User Guide* Utah.

Reichenthal, S. (2002). Re-Introducing Web-Based Simulation, In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yucesan, C. –H. Chen, J.L. Snowdon, and J. M. Charnes.

Rowe, J. (2004). *MCAD Industry News Week in Review – Microsoft and Dassault Systemes Forge Strategic Alliance* Retrieved 5/1/05, MCADCafe.com website: http://www.mcadcafe.com/magazine/index.php?run_date=22-Nov-2004.

Simul8 (n.d.). http://www.simul8.com.

SimVentions (n.d.). www.simventions.com.

Sun Microsystems (2005). *StarOffice 7 Office Suite* Retrieved June 3, 2005 from Sun Microsystems website: http://www.sun.com/software/star/staroffice/whitepapers/index.xml.

W3C (2004). *Extensible Markup Language (XML) 1.0 (Third Edition)* Retrieved 9/8/04, W3C website: http://www.w3.org/TR/REC-xml/#sec-intro.

Wang, Y. & Lu, Y. (2002). An XML-based DEVS Modeling Tool to Enhance Simulation Interoperability. *In Proceedings 14th European Simulation Symposium*, ed. A. Verbraek, W. Krug.

Wiedemann, T. (2002). Next Generation Simulation Environments Founded On Open Source Software And XML-Based Standard Interfaces, In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yucesan, C. –H. Chen, J.L. Snowdon, and J. M. Charnes.

APPENDIX A

ProModel XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:complexType name="ProModelClassicObjectModel">
        <xs:sequence>
            <xs:element name="General" type="GeneralRecord"/>
            <xs:element name="Macros" type="MacroWrite" maxOccurs="unbounded"/>
            <xs:element name="Scenarios" type="ScenarioRecord" maxOccurs="unbounded"/>
            <xs:element name="ExternalFiles" type="ExternalFilesRecord" maxOccurs="unbounded"/>
            <xs:element name="Arrays" type="ArrayFileRecord" maxOccurs="unbounded"/>
            <xs:element name="Subroutines" type="SubroutineRecord" maxOccurs="unbounded"/>
            <xs:element name="Streams" type="StreamRecord" maxOccurs="unbounded"/>
            <xs:element name="Variables" type="VariableRecord" maxOccurs="unbounded"/>
            <xs:element name="Attributes" type="AttributeRecord" maxOccurs="unbounded"/>
            <xs:element name="FuntionTables" type="FunctionTableRecord" maxOccurs="unbounded"/>
            <xs:element name="CycleTables" type="CycleTableRecord" maxOccurs="unbounded"/>
            <xs:element name="DistributionTables" type="DistributionTableRecord"
maxOccurs="unbounded"/>
            <xs:element name="Entities" type="EntityRecord" maxOccurs="unbounded"/>
            <xs:element name="Locations" type="LocationRecord" maxOccurs="unbounded"/>
            <xs:element name="Arrivals" type="ArrivalRecord" maxOccurs="unbounded"/>
            <xs:element name="PathNetworkRecords" type="PathNetworkRecord"
maxOccurs="unbounded"/>
            <xs:element name="Resources" type="ResourceRecord" maxOccurs="unbounded"/>
            <xs:element name="RoutingRecords" type="RoutingRecord" maxOccurs="unbounded"/>
            <xs:element name="BkGraphics" type="BkGraphicRecord"/>
            <xs:element name="ShiftAssignments" type="ShiftAssignmentRecord"
maxOccurs="unbounded"/>
            <xs:element name="ViewRecords" type="ViewRecord" maxOccurs="unbounded"/>
            <xs:element name="DPlotConfigRecords" type="DPlotConfigRecord"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="DPlotConfigRecord">
        <xs:sequence>
            <xs:element name="DPlotConfigRecordCharts" type="DPlotConfigRecordChartRecord"
maxOccurs="unbounded"/>
            <xs:element name="DPlotConfigRecordName" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="DPlotConfigRecordChartRecord">
        <xs:sequence>
            <xs:element name="DPlotConfigRecordChartRecordStatList"
type="DPlotConfigRecordChartRecordStatListRecord" maxOccurs="unbounded"/>
            <xs:element name="DPlotConfigRecordChartRecordCaption" type="xs:string"/>
```

```
            <xs:element name="DPlotConfigRecordChartRecordTitle" type="xs:string"/>
            <xs:element name="DPlotConfigRecordChartRecordTitlePositionX" type="xs:string"/>
            <xs:element name="DPlotConfigRecordChartRecordTitlePositionY" type="xs:string"/>
            <xs:element name="DPlotConfigRecordChartRecordDlg" type="Rect"/>
            <xs:element name="DPlotConfigRecordChartRecordActivePage" type="xs:long"/>
            <xs:element name="DPlotConfigRecordChartRecordWindowState" type="xs:long"/>
            <xs:element name="DPlotConfigRecordChartRecordFileSize" type="xs:long"/>
            <xs:element name="DPlotConfigRecordChartRecordFile" type="xs:long"/>
            <xs:element name="DPlotConfigRecordChartRecordBytes" type="xs:short"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="Rect">
        <xs:sequence>
            <xs:element name="Left" type="xs:long"/>
            <xs:element name="Top" type="xs:long"/>
            <xs:element name="Right" type="xs:long"/>
            <xs:element name="Bottom" type="xs:long"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="DPlotConfigRecordChartRecordStatListRecord">
        <xs:annotation>
            <xs:documentation>WriteStatList</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="DPlotConfigRecordChartRecordStatListRecordLocationID"
type="xs:short"/>
            <xs:element name="DPlotConfigRecordChartRecordStatListRecordEntityID" type="xs:short"/>
            <xs:element name="DPlotConfigRecordChartRecordStatListRecordVariableID"
type="xs:short"/>
            <xs:element name="DPlotConfigRecordChartRecordStatListRecordResourceID"
type="xs:short"/>
            <xs:element name="DPlotConfigRecordChartRecordStatListRecordPathNetworkID"
type="xs:short"/>
            <xs:element name="DPlotConfigRecordChartRecordStatListRecordPathNetworkNode"
type="PathNode"/>
            <xs:element name="DPlotConfigRecordChartRecordStatListRecordLogName"
type="xs:string"/>
            <xs:element name="DPlotConfigRecordChartRecordStatListRecordTableIndex"
type="xs:long"/>
            <xs:element name="DPlotConfigRecordChartRecordStatListRecordStatIndex"
type="xs:long"/>
            <xs:element name="DPlotConfigRecordChartRecordStatListRecordExtType" type="xs:short"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ViewRecord">
        <xs:sequence>
            <xs:element name="ViewRecordName" type="xs:string"/>
            <xs:element name="ViewRecordPanX" type="xs:long"/>
            <xs:element name="ViewRecordPanY" type="xs:long"/>
            <xs:element name="ViewRecordPositionX" type="xs:long"/>
            <xs:element name="ViewRecordPositionY" type="xs:long"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ShiftAssignmentRecord">
        <xs:sequence>
```

```
            <xs:element name="ShiftAssignmentRecords" type="ShiftAssignmentRecordItem"
maxOccurs="unbounded"/>
            <xs:element name="ShiftAssignmentEndShiftPriority" type="xs:string"/>
            <xs:element name="ShiftAssignmentOffShiftPriority" type="xs:string"/>
            <xs:element name="ShiftAssignmentStartBreakPriority" type="xs:string"/>
            <xs:element name="ShiftAssignmentBreakPriority" type="xs:string"/>
            <xs:element name="ShiftAssignmentPreOffShiftLogic" type="xs:string"/>
            <xs:element name="ShiftAssignmentOffShiftLogic" type="xs:string"/>
            <xs:element name="ShiftAssignmentPreBreakLogic" type="xs:string"/>
            <xs:element name="ShiftAssignmentBreakLogic" type="xs:string"/>
            <xs:element name="ShiftAssignmentDisable" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ShiftAssignmentRecordItem">
        <xs:sequence>
            <xs:element name="ShiftAssignmentRecordItemLocationList" type="IndirectRefsRecord"
maxOccurs="unbounded"/>
            <xs:element name="ShiftAssignmentRecordItemResRef" type="ShiftResRecordItem"
maxOccurs="unbounded"/>
            <xs:element name="ShiftAssignmentRecordItemExternalFiles" type="ExternalFilesRecord"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ShiftResRecordItem">
        <xs:sequence>
            <xs:element name="ShiftResRecordItemIndex" type="xs:short"/>
            <xs:element name="ShiftResRecordItemUnits" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="BkGraphicRecord">
        <xs:sequence>
            <xs:element name="BkGraphicsVersion" type="xs:long"/>
            <xs:element name="BkGraphicsRecords" type="PrimitiveRecord"/>
            <xs:element name="BkDropGraphicsRecords" type="PrimitiveRecord"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="PrimitiveRecord">
        <xs:sequence>
            <xs:element name="PrimitiveRecordName" type="xs:string"/>
            <xs:element name="PrimitiveRecordWidth" type="xs:float"/>
            <xs:element name="PrimitiveRecordHeight" type="xs:float"/>
            <xs:element name="PrimitiveRecordUnits" type="xs:integer"/>
            <xs:element name="PrimitiveRecordCount" type="xs:short"/>
            <xs:element name="PrimitiveRecordRecords" type="PrimitiveRecordItem"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="PrimitiveRecordItem">
        <xs:sequence>
            <xs:element name="PrimitiveRecordItemHeader" type="xs:short"/>
            <xs:element name="PrimitiveRecordItemNumber" type="xs:long"/>
            <xs:element name="PrimitiveRecordItemIndex" type="xs:long"/>
            <xs:element name="PrimitiveRecordItemVersion" type="xs:long"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="RoutingRecord">
```

```
        <xs:sequence>
            <xs:element name="RoutingRecordReference" type="PartReference"/>
            <xs:element name="RoutingRecordFromLocationID" type="xs:long"/>
            <xs:element name="RoutingRecordOperation" type="xs:string"/>
            <xs:element name="RoutingRecordToRouting" type="RoutingRecordToRoutingRecord"
maxOccurs="unbounded"/>
            <xs:element name="RoutingRecordFromAll" type="xs:boolean"/>
            <xs:element name="RoutingRecordPreempt" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="RoutingRecordToRoutingRecord">
        <xs:sequence>
            <xs:element name="RoutingRecordToRoutingRecordToLocationID" type="xs:short"/>
            <xs:element name="RoutingRecordToRoutingRecordCondition" type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordDestExpr" type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordResourceID" type="xs:short"/>
            <xs:element name="RoutingRecordToRoutingRecordCrane"
type="RoutingRecordToRoutingRecordCraneRecord" maxOccurs="unbounded"/>
            <xs:element name="RoutingRecordToRoutingRecordRobot"
type="RoutingRecordToRoutingRecordRobotRecord" maxOccurs="unbounded"/>
            <xs:element name="RoutingRecordToRoutingRecordPathNetworkID" type="xs:short"/>
            <xs:element name="RoutingRecordToRoutingRecordConveyor" type="Conveyor"
maxOccurs="unbounded"/>
            <xs:element name="RoutingRecordToRoutingRecordTime" type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordLocationPriority" type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordResourcePriority" type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordOutputPartName" type="PartReference"/>
            <xs:element name="RoutingRecordToRoutingRecordPathPoints" type="Point"/>
            <xs:element name="RoutingRecordToRoutingRecordExitLogic" type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordQuantity" type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordProbability" type="xs:double"/>
            <xs:element name="RoutingRecordToRoutingRecordToExit" type="xs:boolean"/>
            <xs:element name="RoutingRecordToRoutingRecordNumOfPoints" type="xs:short"/>
            <xs:element name="RoutingRecordToRoutingRecordConditionType" type="xs:short"/>
            <xs:element name="RoutingRecordToRoutingRecordKeepResource" type="xs:boolean"/>
            <xs:element name="RoutingRecordToRoutingRecordUseDestExpr" type="xs:boolean"/>
            <xs:element name="RoutingRecordToRoutingRecordBeginBlock" type="xs:boolean"/>
            <xs:element name="RoutingRecordToRoutingRecordPointMoved" type="xs:boolean"/>
            <xs:element name="RoutingRecordToRoutingRecordNewEntity" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="Path">
        <xs:sequence>
            <xs:element name="PathPoints" type="Point"/>
            <xs:element name="PathPPTPolygon" type="Point"/>
            <xs:element name="PathPPTRail1" type="Point"/>
            <xs:element name="PathPPTRail2" type="Point"/>
            <xs:element name="PathPPTRollers" type="Point"/>
            <xs:element name="PathSpeed" type="xs:string"/>
            <xs:element name="PathLength" type="xs:string"/>
            <xs:element name="PathLoadSpacing" type="xs:string"/>
            <xs:element name="PathConveyor" type="xs:boolean"/>
            <xs:element name="PathAccumulate" type="xs:boolean"/>
            <xs:element name="PathWidthWise" type="xs:boolean"/>
            <xs:element name="PathNumOfPoints" type="xs:short"/>
            <xs:element name="PathBorderColor" type="xs:long"/>
```

```
            <xs:element name="PathFillColor" type="xs:long"/>
            <xs:element name="PathStyle" type="xs:short"/>
            <xs:element name="PathWidth" type="xs:short"/>
            <xs:element name="PathPolygon" type="xs:short"/>
            <xs:element name="PathRollers" type="xs:short"/>
            <xs:element name="PathInvisible" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="Conveyor">
        <xs:sequence>
            <xs:element name="ConveyorName" type="xs:string"/>
            <xs:element name="ConveyorPath" type="Path"/>
            <xs:element name="ConveyorClockDowntime" type="ClockDowntimeRecord"/>
            <xs:element name="ConveyorUsageDowntime" type="UsageDowntimeRecord"/>
            <xs:element name="ConveyorNote" type="xs:string"/>
            <xs:element name="ConveyorLoadLimit" type="xs:string"/>
            <xs:element name="ConveyorSpeed" type="xs:string"/>
            <xs:element name="ConveyorLoadLength" type="xs:string"/>
            <xs:element name="ConveyorLoadSpacing" type="xs:string"/>
            <xs:element name="ConveyorStatistics" type="xs:short"/>
            <xs:element name="ConveyorCellular" type="xs:boolean"/>
            <xs:element name="ConveyorAccumulate" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="RoutingRecordToRoutingRecordRobotRecord">
        <xs:sequence>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordName" type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordCapacity" type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordQuantity" type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordNote" type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordClockDowntime"
type="ClockDowntimeRecord"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordUsageDowntime"
type="UsageDowntimeRecord"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordHomeNode"
type="PathNode"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordPickupTime"
type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordDepositTime"
type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecord"
type="RoutingRecordToRoutingRecordRobotRecordPoint" maxOccurs="unbounded"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordMinAttribute"
type="AttributeRecord"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordMaxAttribute"
type="AttributeRecord"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordSearchForEntity"
type="xs:short"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordStatistics" type="xs:short"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordPosition" type="Point"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordColor" type="xs:long"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordBaseWidth"
type="xs:short"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordPointColor"
type="xs:long"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordResource" type="xs:short"/>
```

```
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="RobotConnectionRecord">
        <xs:sequence>
            <xs:element name="RobotConnectionRecordPoint" type="IndirectRefsRecord"/>
            <xs:element name="RobotConnectionRecordTraversalTime" type="xs:string"/>
            <xs:element name="RobotConnectionRecordBiDirectional" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="RoutingRecordToRoutingRecordRobotRecordPoint">
        <xs:sequence>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordPointLocationInterface"
type="LocationRecord"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordPointRobotConnection"
type="RobotConnectionRecord" maxOccurs="unbounded"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordPointWorkSearchList"
type="SearchList"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordPointParkSearchList"
type="IndirectRefsRecord"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordPointEntryLogic"
type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordRobotRecordPointExitLogic"
type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="RoutingRecordToRoutingRecordCraneRecord">
        <xs:sequence>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordName" type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordCapacity" type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordQuantity" type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordNote" type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordClockDowntime"
type="ClockDowntimeRecord"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordUsageDowntime"
type="UsageDowntimeRecord"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordHomeNode"
type="PathNode"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordPickupTime"
type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordDepositTime"
type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordTrollyGraphic"
type="LocationRecordCompositeGraphicRecord" maxOccurs="unbounded"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordHoistAccellRate"
type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordBridgeAccellRate"
type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordHoistDecelRate"
type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordBridgeDecelRate"
type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordHoistDefaultSpeed"
type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordBridgeDefaultSpeed"
type="xs:string"/>
```

```
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordMinAttribute"
type="AttributeRecord"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordMaxAttribute"
type="AttributeRecord"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordSearchForEntity"
type="xs:short"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordStatistics" type="xs:short"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordPointColor"
type="xs:long"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordOrientation"
type="xs:short"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordBoundary" type="Rect"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordColor" type="xs:long"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordPosition" type="Point"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordResource" type="xs:short"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="RoutingRecordToRoutingRecordCraneRecordPoint">
        <xs:sequence>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordPointLocationInterface"
type="LocationRecord"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordPointWorkSearchList"
type="SearchList"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordPointParkSearchList"
type="IndirectRefsRecord"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordPointEntryLogic"
type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordPointExitLogic"
type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordPointBridgeDistance"
type="xs:string"/>
            <xs:element name="RoutingRecordToRoutingRecordCraneRecordPointHoistDistance"
type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="SearchList">
        <xs:sequence>
            <xs:element name="SearchListLocationsList" type="IndirectRefsRecord"/>
            <xs:element name="SearchListExclusiveSearch" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="PathNode">
        <xs:annotation>
            <xs:documentation>LPPATH_NODE</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="PathNodeIndex" type="xs:string"/>
            <xs:element name="PathNodeCapacity" type="xs:string"/>
            <xs:element name="PathNodeRailDist" type="xs:short"/>
            <xs:element name="PathNodeBridgeDist" type="xs:short"/>
            <xs:element name="PathNodeScreenPoint" type="Point"/>
            <xs:element name="PathNodeRunPoint" type="xs:short"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ResourceRecord">
        <xs:sequence>
```

```
            <xs:element name="ResourceRecordIndex" type="xs:short"/>
            <xs:element name="ResourceRecordName" type="xs:string"/>
            <xs:element name="ResourceRecordCapacity" type="xs:string"/>
            <xs:element name="ResourceRecordQuantity" type="xs:string"/>
            <xs:element name="ResourceRecordNote" type="xs:string"/>
            <xs:element name="ResourceRecordClockDowntimes" type="ClockDowntimeRecord"
maxOccurs="unbounded"/>
            <xs:element name="ResourceRecordUsageDowntimes" type="UsageDowntimeRecord"
maxOccurs="unbounded"/>
            <xs:element name="ResourceRecordHomeNodeIndex" type="xs:short"/>
            <xs:element name="ResourceRecordShiftNodeIndex" type="xs:short"/>
            <xs:element name="ResourceRecordBreakNodeIndex" type="xs:short"/>
            <xs:element name="ResourceRecordPickupTime" type="xs:string"/>
            <xs:element name="ResourceRecordDepositTime" type="xs:string"/>
            <xs:element name="ResourceRecordLibraryGraphicRecord" type="LibraryGraphicRecord"/>
            <xs:element name="ResourceRecordDiscreteParts" type="DiscretePartRecord"
maxOccurs="unbounded"/>
            <xs:element name="ResourceRecordAccelRate" type="xs:string"/>
            <xs:element name="ResourceRecordDecelRate" type="xs:string"/>
            <xs:element name="ResourceRecordFullSpeed" type="xs:string"/>
            <xs:element name="ResourceRecordEmptySpeed" type="xs:string"/>
            <xs:element name="ResourceRecordParkSearch" type="ResourceRecordParkSearchRecord"
maxOccurs="unbounded"/>
            <xs:element name="ResourceRecordWorkSearch" type="ResourceWorkSearchRecord"
maxOccurs="unbounded"/>
            <xs:element name="ResourceRecordResourcePoints" type="ResourceRecordPointRecord"
maxOccurs="unbounded"/>
            <xs:element name="ResourceRecordMinAttribute" type="AttributeRecord"/>
            <xs:element name="ResourceRecordMaxAttribute" type="AttributeRecord"/>
            <xs:element name="ResourceRecordNodeLogic" type="ResourceRecordNodeLogicRecord"
maxOccurs="unbounded"/>
            <xs:element name="ResourceRecordCostRate" type="xs:string"/>
            <xs:element name="ResourceRecordOTcostRate" type="xs:string"/>
            <xs:element name="ResourceRecordCostPerUse" type="xs:string"/>
            <xs:element name="ResourceRecordCostTimeUnits" type="xs:short"/>
            <xs:element name="ResourceRecordSearchForEntity" type="xs:short"/>
            <xs:element name="ResourceRecordStatistics" type="xs:short"/>
            <xs:element name="ResourceRecordReturnToHome" type="xs:boolean"/>
            <xs:element name="ResourceRecordSearchForResource" type="xs:short"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ResourceRecordNodeLogicRecord">
        <xs:sequence>
            <xs:element name="ResourceRecordNodeLogicRecordIndex" type="xs:short"/>
            <xs:element name="ResourceRecordNodeLogicRecordEntryOperation" type="xs:string"/>
            <xs:element name="ResourceRecordNodeLogicRecordExitOperation" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ResourceRecordPointRecord">
        <xs:sequence>
            <xs:element name="ResourceRecordPointRecordNodeIndex" type="xs:short"/>
            <xs:element name="ResourceRecordPointRecordPoint" type="Point"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ResourceWorkSearchRecord">
        <xs:sequence>
```

```
            <xs:element name="ResourceWorkSearchRecordPathNodeIndex" type="xs:short"/>
            <xs:element name="ResourceWorkSearchRecordList" type="IndirectRefsRecord"
maxOccurs="unbounded"/>
            <xs:element name="ResourceWorkSearchRecordType" type="xs:short"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ResourceRecordParkSearchRecord">
        <xs:sequence>
            <xs:element name="ResourceRecordParkSearchRecordPathNodeIndex" type="xs:short"/>
            <xs:element name="ResourceRecordParkSearchRecordList" type="IndirectRefsRecord"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="PathNetworkRecord">
        <xs:sequence>
            <xs:element name="PathNetworkName" type="xs:string"/>
            <xs:element name="PathNetworkNodes" type="PathNetworkNodeRecord"
maxOccurs="unbounded"/>
            <xs:element name="PathNetworkSegments" type="PathNetworkSegmentRecord"
maxOccurs="unbounded"/>
            <xs:element name="PathNetworkInterfaces" type="PathNetworkInterfaceRecord"
maxOccurs="unbounded"/>
            <xs:element name="PathNetworkMappings" type="PathNetworkMappingRecord"
maxOccurs="unbounded"/>
            <xs:element ref="PathNetworkCranes" maxOccurs="unbounded"/>
            <xs:element name="PathNetworkCalcSpeedByDist" type="xs:boolean"/>
            <xs:element name="PathNetworkType" type="xs:boolean"/>
            <xs:element name="PathNetworkColor" type="xs:long"/>
            <xs:element name="PathNetworkInvisible" type="xs:boolean"/>
            <xs:element name="PathNetworkNetworkIndex" type="xs:short"/>
            <xs:element name="PathNetworkRecompile" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="PathNetworkCranes">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="PathNetworkCraneSpacing" type="xs:string"/>
                <xs:element name="PathNetworkCraneRailWidth" type="xs:long"/>
                <xs:element name="PathNetworkCraneRailBorderColor" type="xs:long"/>
                <xs:element name="PathNetworkCraneRailFillColor" type="xs:long"/>
                <xs:element name="PathNetworkCraneRailVisible" type="xs:boolean"/>
                <xs:element name="PathNetworkCraneBridgeWidth" type="xs:long"/>
                <xs:element name="PathNetworkCraneBridgeBorderColor" type="xs:long"/>
                <xs:element name="PathNetworkCraneBridgeFillColor" type="xs:long"/>
                <xs:element name="PathNetworkCraneBridgeVisible" type="xs:boolean"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="PathNetworkMappingRecord">
        <xs:sequence>
            <xs:element name="PathNetworkMappingRecordFromID" type="xs:short"/>
            <xs:element name="PathNetworkMappingRecordToID" type="xs:short"/>
            <xs:element name="PathNetworkMappingRecordDestinations" type="IndirectRefsRecord"
maxOccurs="unbounded"/>
            <xs:element name="PathNetworkMappingRecordCompilerDestinations"
type="IndirectRefsRecord" maxOccurs="unbounded"/>
```

```
    </xs:sequence>
</xs:complexType>
<xs:complexType name="IndirectRefsRecord">
    <xs:sequence>
        <xs:element name="IndirectRefsRecordRefListRecordID" type="xs:short"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="PathNetworkInterfaceRecord">
    <xs:sequence>
        <xs:element name="PathNetworkInterfaceRecordPathNodeID" type="xs:long"/>
        <xs:element name="PathNetworkInterfaceRecordLocationID" type="xs:long"/>
        <xs:element name="PathNetworkInterfaceRecordGraphicID" type="xs:long"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="PathNetworkSegmentRecord">
    <xs:sequence>
        <xs:element name="PathNetworkSegmentRecordFromNodeID" type="xs:long"/>
        <xs:element name="PathNetworkSegmentRecordToNodeID" type="xs:long"/>
        <xs:element name="PathNetworkSegmentRecordSpeedFactor" type="xs:string"/>
        <xs:element name="PathNetworkSegmentRecordDistance" type="xs:string"/>
        <xs:element name="PathNetworkSegmentRecordTime" type="xs:string"/>
        <xs:element name="PathNetworkSegmentRecordPath" type="xs:short"/>
        <xs:element name="PathNetworkSegmentRecordPathPoint" type="Point"/>
        <xs:element name="PathNetworkSegmentRecordBiDirectional" type="xs:boolean"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="PathNetworkNodeRecord">
    <xs:sequence>
        <xs:element name="PathNetworkNodeID" type="xs:string"/>
        <xs:element name="PathNetworkNodeCapacity" type="xs:string"/>
        <xs:element name="PathNetworkNodeRailDistace" type="xs:float"/>
        <xs:element name="PathNetworkNodeBridgeDistance" type="xs:short"/>
        <xs:element name="PathNetworkNodeScreenPoint" type="Point"/>
        <xs:element name="PathNetworkNodeRunPoint" type="xs:short"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="ArrivalRecord">
    <xs:sequence>
        <xs:element name="ArrivalRecordPartReference" type="PartReference"/>
        <xs:element name="ArrivalRecordLocationID" type="xs:short"/>
        <xs:element name="ArrivalRecordFrequency" type="xs:string"/>
        <xs:element name="ArrivalRecordNumberofArrivals" type="xs:string"/>
        <xs:element name="ArrivalRecordQuantityPerArrival" type="xs:string"/>
        <xs:element name="ArrivalRecordFirstArrival" type="xs:string"/>
        <xs:element name="ArrivalRecordOperation" type="xs:string"/>
        <xs:element name="ArrivalRecordCycleTableID" type="xs:short"/>
        <xs:element name="ArrivalRecordTimeOffset" type="xs:string"/>
        <xs:element name="ArrivalRecordVariation" type="xs:string"/>
        <xs:element name="ArrivalRecordTimeBasis" type="xs:short"/>
        <xs:element name="ArrivalRecordRepeatType" type="xs:short"/>
        <xs:element name="ArrivalRecordDisabled" type="xs:boolean"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="LocationRecord">
    <xs:sequence>
        <xs:element name="LocationRecordName" type="xs:string"/>
```

```
            <xs:element name="LocationSetupDowntimes" type="LocationSetupDowntimeRecord"
maxOccurs="unbounded"/>
            <xs:element name="LocationRecordClockDowntimes" type="ClockDowntimeRecord"
maxOccurs="unbounded"/>
            <xs:element name="LocationRecordUsageDowntimes" type="UsageDowntimeRecord"
maxOccurs="unbounded"/>
            <xs:element name="LocationRecordCyclicDowntimes"
type="LocationRecordCyclicDowntimeRecord" maxOccurs="unbounded"/>
            <xs:element name="LocationRecordCalledDowntimes"
type="LocationRecordCalledDowntimeRecord" maxOccurs="unbounded"/>
            <xs:element name="LocationRecordNote" type="xs:string"/>
            <xs:element name="LocationRecordCompositeGraphic"
type="LocationRecordCompositeGraphicRecord" maxOccurs="unbounded"/>
            <xs:element name="LocationRecordCapacity" type="xs:string"/>
            <xs:element name="LocationRecordInputAttribute" type="AttributeRecord"/>
            <xs:element name="LocationRecordOutputAttribute" type="AttributeRecord"/>
            <xs:element name="LocationRecordCostRate" type="xs:string"/>
            <xs:element name="LocationRecordCostIdleRate" type="xs:string"/>
            <xs:element name="LocationRecordCostPerEntry" type="xs:string"/>
            <xs:element name="LocationRecordCostPerSetup" type="xs:string"/>
            <xs:element name="LocationRecordCostTimeUnits" type="xs:short"/>
            <xs:element name="LocationRecordMultiUnitQuantity" type="xs:short"/>
            <xs:element name="LocationRecordStatistics" type="xs:short"/>
            <xs:element name="LocationRecordLocationType" type="xs:short"/>
            <xs:element name="LocationRecordSelectionRule" type="xs:short"/>
            <xs:element name="LocationRecordQueueRule" type="xs:short"/>
            <xs:element name="LocationRecordAccessRule" type="xs:short"/>
            <xs:element name="LocationRecordResourcePoints" type="xs:short"/>
            <xs:element name="LocationRecordLocationIndex" type="xs:short"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="LocationSetupDowntimeRecord">
        <xs:sequence>
            <xs:element name="LocationSetupDowntimeRecordPartReference" type="PartReference"/>
            <xs:element name="LocationSetupDowntimeRecordPriorPartReference"
type="PartReference"/>
            <xs:element name="LocationSetupDowntimeRecordOperation" type="xs:string"/>
            <xs:element name="LocationSetupDowntimeRecordDisabled" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="PartReference">
        <xs:annotation>
            <xs:documentation>This is a pointer to a Part and could be shown as "ID" to follow
paradigm</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="PartIndex" type="xs:short"/>
            <xs:element name="PartType" type="xs:short"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ClockDowntimeRecord">
        <xs:sequence>
            <xs:element name="ClockDowntimeRecordOperation" type="xs:string"/>
            <xs:element name="ClockDowntimeRecordFirstOccurrence" type="xs:string"/>
            <xs:element name="ClockDowntimeRecordFrequency" type="xs:string"/>
            <xs:element name="ClockDowntimeRecordList" type="xs:string"/>
```

```
            <xs:element name="ClockDowntimeRecordPriority" type="xs:string"/>
            <xs:element name="ClockDowntimeRecordNode" type="PathNode"/>
            <xs:element name="ClockDowntimeRecordDisabled" type="xs:boolean"/>
            <xs:element name="ClockDowntimeRecordSuspendStatistics" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="UsageDowntimeRecord">
        <xs:sequence>
            <xs:element name="UsageDowntimeRecordOperation" type="xs:string"/>
            <xs:element name="UsageDowntimeRecordFirstOccurrence" type="xs:string"/>
            <xs:element name="UsageDowntimeRecordFrequency" type="xs:string"/>
            <xs:element name="UsageDowntimeRecordList" type="xs:string"/>
            <xs:element name="UsageDowntimeRecordPriority" type="xs:string"/>
            <xs:element name="UsageDowntimeRecordNode" type="PathNode"/>
            <xs:element name="UsageDowntimeRecordDisabled" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="LocationRecordCyclicDowntimeRecord">
        <xs:sequence>
            <xs:element name="LocationRecordCyclicDowntimeRecordOperation" type="xs:string"/>
            <xs:element name="LocationRecordCyclicDowntimeRecordFirstOccurrence"
type="xs:string"/>
            <xs:element name="LocationRecordCyclicDowntimeRecordFrequency" type="xs:string"/>
            <xs:element name="LocationRecordCyclicDowntimeRecordDisabled" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="LocationRecordCalledDowntimeRecord">
        <xs:sequence>
            <xs:element name="LocationRecordCalledDowntimeRecordName" type="xs:string"/>
            <xs:element name="LocationRecordCalledDowntimeRecordPriority" type="xs:string"/>
            <xs:element name="LocationRecordCalledDowntimeRecordOperation" type="xs:string"/>
            <xs:element name="LocationRecordCalledDowntimeRecordSuspendStatistics"
type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="LocationRecordCompositeGraphicRecord">
        <xs:sequence>
            <xs:element name="LocationRecordCompositeGraphicLibraryGraphicRecord"
type="LibraryGraphicRecord"/>
            <xs:element name="LocationRecordCompositeGraphicPosition" type="Point"/>
            <xs:element name="LocationRecordCompositeGraphicWidth" type="xs:short"/>
            <xs:element name="LocationRecordCompositeGraphicHeight" type="xs:short"/>
            <xs:element name="LocationRecordCompositeGraphicRotation" type="xs:short"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="EntityRecord">
        <xs:sequence>
            <xs:element name="EntityRecordName" type="xs:string"/>
            <xs:element name="EntityRecordNote" type="xs:string"/>
            <xs:element name="EntityRecordWidth" type="xs:string"/>
            <xs:element name="EntityRecordLength" type="xs:string"/>
            <xs:element name="EntityRecordSpeed" type="xs:string"/>
            <xs:element name="EntityRecordDiscreteParts" type="DiscretePartRecord"
maxOccurs="unbounded"/>
            <xs:element name="EntityRecordInitialCost" type="xs:string"/>
            <xs:element name="EntityRecordOtherCost" type="xs:string"/>
```

```
            <xs:element name="EntityRecordPartType" type="xs:short"/>
            <xs:element name="EntityRecordIndex" type="xs:short"/>
            <xs:element name="EntityRecordRotationType" type="xs:short"/>
            <xs:element name="EntityRecordStatistics" type="xs:short"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="DiscretePartRecord">
        <xs:sequence>
            <xs:element name="DiscretePartRecordLibraryGraphic" type="LibraryGraphicRecord"/>
            <xs:element name="DiscretePartRecordWidth" type="xs:string"/>
            <xs:element name="DiscretePartRecordLength" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="DistributionTableRecord">
        <xs:sequence>
            <xs:element name="DistributionTableRecordName" type="xs:string"/>
            <xs:element name="DistributionTableDCValues" type="DCValueRecord"
maxOccurs="unbounded"/>
            <xs:element name="DistributionTableCumulative" type="xs:boolean"/>
            <xs:element name="DistributionTableType" type="xs:short"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="CycleTableRecord">
        <xs:sequence>
            <xs:element name="CycleTableRecordName" type="xs:string"/>
            <xs:element name="CycleTableDCValues" type="DCValueRecord"
maxOccurs="unbounded"/>
            <xs:element name="CycleTableCumulative" type="xs:boolean"/>
            <xs:element name="CycleTableQuantity" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="DCValueRecord">
        <xs:sequence>
            <xs:element name="DCValueRecordPercent" type="xs:string"/>
            <xs:element name="DCValueRecordTime" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="FunctionTableRecord">
        <xs:sequence>
            <xs:element name="FuntionTableRecordName" type="xs:string"/>
            <xs:element name="FunctionTableValues" type="FuntionTableValueRecord"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="FuntionTableValueRecord">
        <xs:sequence>
            <xs:element name="FuntionTableValueRecordIndependent" type="xs:string"/>
            <xs:element name="FuntionTableValueRecordDependent" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="AttributeRecord">
        <xs:sequence>
            <xs:element name="AttributeIndex" type="xs:string"/>
            <xs:element name="AttributeNote" type="xs:string"/>
            <xs:element name="AttributeOffset" type="xs:short"/>
            <xs:element name="AttributeKind" type="xs:short"/>
```

```
            <xs:element name="AttributeType" type="xs:short"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="VariableRecord">
        <xs:sequence>
            <xs:annotation>
                <xs:documentation>VariableGraphics should be set here.</xs:documentation>
            </xs:annotation>
            <xs:element name="VariableIndex" type="xs:string"/>
            <xs:element name="VariableInitialValue" type="xs:string"/>
            <xs:element name="VariableNote" type="xs:string"/>
            <xs:element name="VariableLibraryGraphics" type="LibraryGraphicRecord"/>
            <xs:element name="VariableType" type="xs:integer"/>
            <xs:element name="VariableStats" type="xs:short"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="LibraryGraphicRecord">
        <xs:annotation>
            <xs:documentation>Code uses case statement which is not supported natively in XML Schema
so only sample (non-inclusive) code provided for GT_LIBRARY.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="GraphicLibraryWidth" type="xs:float"/>
            <xs:element name="GraphicLibraryHeight" type="xs:float"/>
            <xs:element name="GraphicLibraryRotation" type="xs:long"/>
            <xs:element name="GraphicLibraryHotSpot" type="Point"/>
            <xs:element name="GraphicLibraryPosition" type="Point"/>
            <xs:element name="GraphicLibraryIndex" type="xs:short"/>
            <xs:element name="GraphicLibraryColor" type="xs:long"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="Point">
        <xs:sequence>
            <xs:element name="PositionX" type="xs:long"/>
            <xs:element name="PositionY" type="xs:long"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="StreamRecord">
        <xs:sequence>
            <xs:element name="StreamIndex" type="xs:string"/>
            <xs:element name="StreamSeed" type="xs:long"/>
            <xs:element name="Streamreset" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="SubroutineRecord">
        <xs:sequence>
            <xs:element name="SubroutineIndex" type="xs:string"/>
            <xs:element name="SubroutineOperation" type="xs:string"/>
            <xs:element name="SubroutineParameters" type="SubroutineRecordWriteOperation"
maxOccurs="unbounded"/>
            <xs:element name="SubroutineType" type="xs:integer"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="SubroutineRecordWriteOperation">
        <xs:sequence>
            <xs:element name="SubroutineText" type="xs:string"/>
```

```
            <xs:element name="SubroutineType" type="xs:integer"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ArrayFileRecord">
        <xs:sequence>
            <xs:element name="ArrayFileIndex" type="xs:string"/>
            <xs:element name="ArrayFileNumberOfDimensions" type="xs:short"/>
            <xs:element name="ArrayFileDimensions" type="xs:long"/>
            <xs:element name="ArrayFileMultFactor" type="xs:long"/>
            <xs:element name="ArrayFileNote" type="xs:string"/>
            <xs:element name="ArrayFileTotalElements" type="xs:unsignedLong"/>
            <xs:element name="ArrayFileType" type="xs:integer"/>
            <xs:element name="ArrayFileRecordIndex" type="xs:short"/>
            <xs:element name="ArrayFileExportSheetName" type="xs:string"/>
            <xs:element name="ArrayFileExportStartCell" type="xs:string"/>
            <xs:element name="ArrayFileExportEndCell" type="xs:string"/>
            <xs:element name="ArrayFileImportSheetName" type="xs:string"/>
            <xs:element name="ArrayFileImportStartCell" type="xs:string"/>
            <xs:element name="ArrayFileImportEndCell" type="xs:string"/>
            <xs:element name="ArrayFileDBConnect" type="xs:string"/>
            <xs:element name="ArrayFileDBSQLCommand" type="xs:string"/>
            <xs:element name="ArrayFileDBRecordCount" type="xs:string"/>
            <xs:element name="ArrayFileImportType" type="xs:integer"/>
            <xs:element name="ArrayFileExportType" type="xs:integer"/>
            <xs:element name="ArrayFileStoredProc" type="xs:boolean"/>
            <xs:element name="ArrayFileKeepValues" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ExternalFilesRecord">
        <xs:sequence>
            <xs:element name="ExternalFilesCount" type="xs:short"/>
            <xs:element name="ExternalFileRecords" type="ExternalFileRecord"
maxOccurs="unbounded"/>
            <xs:element name="ScenarioName" type="xs:string"/>
            <xs:element name="ScenariosRun" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ExternalFileRecord">
        <xs:sequence>
            <xs:element name="ExternalFileIndex" type="xs:string"/>
            <xs:element name="ExternalFilePath" type="xs:string"/>
            <xs:element name="ExternalFilePrompt" type="xs:string"/>
            <xs:element name="ExternalFileNote" type="xs:string"/>
            <xs:element name="ExternalFileType" type="xs:integer"/>
            <xs:element name="ExternalFileLastWrittenToDate" type="xs:duration"/>
            <xs:element name="ExternalFileLastWrittenToDate" type="xs:duration"/>
            <xs:element name="ExternalFileSize" type="xs:unsignedInt"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ScenarioRecord">
        <xs:sequence>
            <xs:element name="ScenariosCount" type="xs:short"/>
            <xs:element name="ScenarioName" type="xs:string"/>
            <xs:element name="ScenarioRTIVals" type="ScenarioRTIValRecord"
maxOccurs="unbounded"/>
            <xs:element name="ScenariosRun" type="xs:boolean"/>
```

```
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ScenarioRTIValRecord">
        <xs:sequence>
            <xs:element name="ScenarioRTIValRecordValue" type="xs:short"/>
            <xs:element name="ScenarioRTIValRecordText" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="MacroWrite">
        <xs:sequence>
            <xs:element name="MacroSymbol" type="xs:string"/>
            <xs:element name="MacroText" type="xs:string"/>
            <xs:element name="MacroRunTimeInterfaces" type="MacroWriteRunTimeInterface"
maxOccurs="unbounded"/>
            <xs:element name="MacroGroup" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="MacroWriteRunTimeInterface">
        <xs:sequence>
            <xs:element name="MacroEmpty" type="xs:boolean"/>
            <xs:element name="MacroDescription" type="xs:string"/>
            <xs:element name="MacroInstructions" type="xs:string"/>
            <xs:element name="MacroRangeMinValue" type="xs:string"/>
            <xs:element name="MacroRangeMaxValue" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="GeneralRecord">
        <xs:all>
            <xs:element name="ModelVersionNumber" type="xs:positiveInteger"/>
            <xs:element name="ModelTitle" type="xs:string"/>
            <xs:element name="ModelNote" type="xs:anySimpleType"/>
            <xs:element name="ModelRunLength" type="xs:string"/>
            <xs:element name="ModelWarmUpPeriod" type="xs:string"/>
            <xs:element name="ModelUseWarmUp" type="xs:boolean"/>
            <xs:element name="ModelTimeBasis" type="xs:integer"/>
            <xs:element name="ModelShowNotes" type="xs:boolean"/>
            <xs:element name="ModelInitialPause" type="xs:boolean"/>
            <xs:element name="ModelAdjustDaylightSavingsTime" type="xs:boolean"/>
            <xs:element name="ModelTimeSimulationBegin" type="tm"/>
            <xs:element name="ModelTimeSimulationEnd" type="tm"/>
            <xs:element name="ModelTimeSimulationWarmUp" type="tm"/>
            <xs:element name="ModelUseWarmUp" type="xs:boolean"/>
            <xs:element name="ModelStreamSeedsPresent" type="xs:boolean"/>
            <xs:element name="ModelStreamSeedsStreamNumber" type="xs:short"/>
            <xs:element name="ModelStreamSeedsSeedNumber" type="xs:short"/>
            <xs:element name="ModelStreamSeedsReset" type="xs:boolean"/>
            <xs:element name="ModelBackgroundBitmapFile" type="xs:string"/>
            <xs:element name="ModelOutputFile" type="xs:string"/>
            <xs:element name="ModelGraphicsLibraryFile" type="xs:string"/>
            <xs:element name="ModelRelications" type="xs:string"/>
            <xs:element name="ModelNumberofStreams" type="xs:short"/>
            <xs:element name="ModelDefaultTimeUnit" type="xs:integer"/>
            <xs:element name="ModelClockPrecision" type="xs:integer"/>
            <xs:element name="ModelClockPrecisionUnits" type="xs:integer"/>
            <xs:element name="ModelDisableTimeSeries" type="xs:boolean"/>
            <xs:element name="ModelDisableDowntimes" type="xs:boolean"/>
```

```
            <xs:element name="ModelDisableAnimation" type="xs:boolean"/>
            <xs:element name="ModelDisableCosts" type="xs:boolean"/>
            <xs:element name="ModelDisableArrayExport" type="xs:boolean"/>
            <xs:element name="ModelUseCommonRandomNumbers" type="xs:boolean"/>
            <xs:element name="ModelGenerateAnimationScript" type="xs:boolean"/>
            <xs:element name="ModelScreenWidth" type="xs:short"/>
            <xs:element name="ModelScreenHeigth" type="xs:short"/>
            <xs:element name="ModelDefaultSizeUnit" type="xs:integer"/>
            <xs:element name="ModelBackgroundColor" type="xs:unsignedLong"/>
            <xs:element name="ModelStatusLightWidth" type="xs:short"/>
            <xs:element name="ModelPartSpothtWidth" type="xs:short"/>
            <xs:element name="ModelInitializationLogic" type="xs:string"/>
            <xs:element name="ModelTerminationLogic" type="xs:string"/>
            <xs:element name="ModelOperationStatistics" type="xs:integer"/>
            <xs:element name="ModelGridSize" type="xs:integer"/>
            <xs:element name="ModelGridColor1" type="xs:unsignedInt"/>
            <xs:element name="ModelGridColor2" type="xs:unsignedInt"/>
            <xs:element name="ModelGridTimeUnits" type="xs:double"/>
            <xs:element name="ModelGridDistanceUnits" type="xs:float"/>
            <xs:element name="ModelRecalculateSegments" type="xs:boolean"/>
            <xs:element name="ModelRoutingNormalColor" type="xs:unsignedInt"/>
            <xs:element name="ModelRoutingProcessColor" type="xs:unsignedInt"/>
            <xs:element name="ModelRoutingCurrentColor" type="xs:unsignedInt"/>
            <xs:element name="ModelPanXPosition" type="xs:integer"/>
            <xs:element name="ModelPanYPosition" type="xs:integer"/>
            <xs:element name="ModelGridZoom" type="xs:float"/>
            <xs:element name="ModelReportInterval" type="xs:string"/>
            <xs:element name="ModelStatsMethod" type="xs:integer"/>
        </xs:all>
    </xs:complexType>
    <xs:complexType name="tm">
        <xs:annotation>
            <xs:documentation>Implementation of tm structure</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="tmSec" type="xs:nonNegativeInteger"/>
            <xs:element name="tmMin" type="xs:nonNegativeInteger"/>
            <xs:element name="tmHour" type="xs:nonNegativeInteger"/>
            <xs:element name="tmMday" type="xs:positiveInteger"/>
            <xs:element name="tmMon" type="xs:nonNegativeInteger"/>
            <xs:element name="tmYear" type="xs:positiveInteger"/>
            <xs:element name="tmWday" type="xs:nonNegativeInteger"/>
            <xs:element name="tmYday" type="xs:nonNegativeInteger"/>
            <xs:element name="tmIsdst" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

APPENDIX B

ProModel / NIST Comparison Mappings

The Four Vital Simulation Elements

1. Locations
Supported
LocationRecordName:
This is the name given for the location. The element is supported in <resources><stations><station><name/>.

LocationRecordNote:
This element holds optional notes related to the location. The element is supported in <resources><stations><station><description/>.

Location Record Capacity:
This field refers to the number of entities that the location can hold. The field can hold logic which is evaluated at the beginning of the simulation run and is a static value during simulation. Even though logic is not supported in the SDM this element is deemed supported because it supports the more common use of the element in <resources><stations><station><machine keys/><machine-specifications><workpiece-capacity/>.

LocationRecordCostRate:
This element specifies the cost associated with the operation of a location. Even though this concept is supported, ProModel provides a method to specify the unit of time cost multiplier in LocationRecordCostTimeUnits. For this mapped value to be fully supported, <hourly-rate> would need to be changed to something more general such as simply <rate/> then LocationRecordCostTimeUnits would need to be supported in <resources><stations><station><hourly-rate/>.

LocationRecordLocationType:
ProModel uses this element to specify that the location is one of eight types which include: single capacity location, multi-unit location, multi-unit member location, conveyor, conveyor multi-unit, conveyor multi-unit member, tank, or none. The type field supplies an adequate element to specify the mapping of this element; however additional types in the SDM need to be added to support all of the ProModel equivalents using <resources><stations><station><type/>.

LocationRecordIndex:
This element is a number that uniquely identifies the location. The SDM contains an element <identifier/> that is an auto generated number to identify each element of this type using supported in <resources><stations><station><identifier/>.

LocationRecordMultiUnitQuantity:
Multi-unit locations are used in ProModel software when there are multiple processing locations of exactly the same type and yet uniqueness is desired for each location in order to track statistics or alter individual location characteristics. If a multi-unit location can instead be represented using a multi-capacity location this type of differentiation in not necessary and could allow for representation via a single location which is supported. Another approach could also be to model the multi-unit location as a <stations><station-group> which would mean that the quantity is simply a count of the elements contained in this station group using <stations><station-group>.

Unsupported Necessary
LocationSetupDowntime:
A setup downtime is a portion of time used to prepare a machine to perform an operation. The SDM schema supports the concept of a setup downtime in the data element <resources><stations><station><station-status><work-assignments/><work-assignment><setup-definition-keys>. However in order to execute a downtime, LocationSetupDowntimes needs to contain a reference to the amount of time the setup will take. In the ProModel schema the string LocationSetupDowntimeRecordOperation holds logic to perform the setup operation. In order to map this value a <probability-distribution> element could be included to define the setup time such as <resources><stations><station><station-status><work-assignments/><work-assignment><setup-definition-keys><probability-distribution>.

LocationRecordClockDowntimes:
This type of downtime is performed when a predefined time on the simulation clock has expired. This can be used to model a routine maintenance schedule. The ProModel schema holds an element ClockDowntimeRecordOperation that holds a logic string to perform this operation. This type of downtime is available to single and multi capacity locations.

LocationRecordUsageDowntimes:
This type of downtime is performed when the location has been in operation for a certain amount of time. This type of downtime utilizes logic and is only available to single capacity locations.

LocationRecordCyclicDowntimes:
This type of downtime is performed when a location has processed a certain number of items. It contains the data item LocationRecordCyclicDowntimeRecordOperation which contains a logic string to perform this operation. This type of downtime is only available to single capacity locations.

LocationRecordCalledDowntimes:
This type of downtime is performed by directly calling for the downtime to occur from other user programmable sections of code. This allows a downtime to occur based on user customized conditions which can be specified using logic statements. This type of downtime is available to single and multi capacity locations.

LocationRecordInputAttribute:
This data type contains an associated location attribute record. This value is used when the values for LocationRecordAccessRule are either Highest Attribute Value or Lowest Attribute Value. The element could be supported by referencing the element located in another stations <resources><stations><station><attribute/> element.

LocationRecordOutputAttribute:
This data type contains an associated location attribute record. This value is used when the values for LocationRecordQueueRule are either Highest Attribute Value or Lowest Attribute Value. The element could be supported by referencing the element located in another stations <resources><stations><station><attribute/> element.

LocationRecordCostTimeUnits:
This data item allows for the setting of the units of measure that are associated with LocationRecordCostRate. The options are: second, minute, hour, day. This could be implemented using the SDM time-duration-units enumeration specified in an element such as <resources><stations><station><rate-units/>.

LocationRecordStatistics:
This element is an enumeration that determines the type of statistics to keep track of for a location. The options are None, Average, Standard Deviation, and Time-Series. To support this element the SDM would need to implement an enumeration with the necessary options such as <resources><stations><station><statistics-type/>.

LocationRecordSelectionRule:
For ProModel multi-unit locations only, this enumeration selects the logic that is used to decide which available unit in a multi-unit location an entity is to be routed to for processing. The options are First Available, By Turn, Most Available Capacity, Fewest Entries, Random, and Longest Empty. The representation of a multi-unit location in the SDM is found in <stations><station-group>. The specification of a rule by which a particular station within a <station-group> should be selected, might be accomplished with the addition of an element that contained an enumeration of the necessary values such as <process-plans><process-plan><routing-sheets><routing-sheet><selection-rule>.

LocationRecordQueueRule:
An enumeration used when an entity finishes operation at a location, it must decide how to exit if other entities that are ahead of it have not yet departed the location. The options are No Queuing, FIFO, LIFO, By Type, Highest Attribute Value, and Lowest Attribute Value. The concept might be supported in the SDM with the addition of this enumeration to <routing-sheets><routing-sheet><queue-rule>.

LocationRecordAccessRule:
An enumeration used when multiple elements are waiting to enter a location at the same time. When this occurs a decision needs to be made as to which one is admitted first. The options are Oldest by Priority, Random, Least Available Capacity, Last Selected Location, Highest Attribute Value, and Lowest Attribute Value. This concept might be controlled with the addition an enumeration to <routing-sheets><routing-sheet><access-rule>.

Unsupported Unnecessary
LocationRecordCompositeGraphic:
This data type contains a reference to graphics in the ProModel Graphic Library (LibraryGraphicRecord) which are associated with each location. The graphics associated with the data elements LocationRecordCompositeGraphicHasPointer and LocationRecordCompositeGraphicLibraryGraphicRecord of this location can be described using the <resources><stations><station><reference-keys/><reference-key><digital-files>. However LocationRecordCompositeGraphic also contains data items LocationRecordCompositeGraphicPosition, LocationRecordCompositeGraphicWidth, LocationRecordCompositeGraphicHeight, and LocationRecordCompositeGraphicRotation. Together these data items are used to describe the size and position of the graphic as they are displayed in the ProModel UI. These could be represented by the SDM if a <reference-frame-key> element were added such as <resources><stations><station><reference-frame-keys/>.

LocationRecordCostIdleRate:
Not currently implemented in ProModel.

LocationrecordCostPerEntry:
Not currently implemented in ProModel.

LocationRecordCostPerSetup:
Not currently implemented in ProModel.

2. Entities
Supported
EntityRecordName:
The name of the entity supported in <parts type><part><name/>.

EntityRecordNote:
This element holds optional notes related to the entity which is supported in <parts type><part><description/>.

EntityRecordWidth:
This element specifies the logical width of the entity supported in <parts type><part><part-specifications><width/>.

EntityRecordLength:
This element specifies the logical length of the entity. This element is supported in <parts type><part><part-specifications><length/>.

EntityRecordInitialCost:
This element holds the cost that is assumed when an entity enters the system. The element is supported in <parts type><part><part-specification><unit-cost/>.

EntityRecordPartType:
ProModel uses this element to specify that the entity is one of two types which include: discrete and continuous. The type field supplies an adequate element to specify the mapping of this element; however additional types in the SDM would need to be added to support all of the ProModel equivalents in the element <parts type><part><type/>.

EntityRecordIndex:
This element is a number that uniquely identifies the entity. The SDM contains an element <identifier/> that is an auto generated number to identify each element of this type supported in <parts type><part><identifier/>.

Unsupported Necessary
EntityRecordStatistics:
This element is an enumeration that determines the type of statistics to keep track of for an entity. The options are None, Average, Standard Deviation, and Time-Series. To support this element the SDM would need to implement an enumeration with the necessary options such as <parts type><part><statistics-type/>.

EntityRecordSpeed:
This element keeps track of the entities speed through the system. It typically applies to self-moving entities such as humans as they move through a system. This concept could be supported with the addition of an element such as <parts type><part><part-specifications><speed/>

Unsupported Unnecessary
EntityRecordDiscreteParts:
The main ProModel schema category EntityRecordDiscreteParts contains nested structures which describe the entities graphical representation. This representation can be associated with multiple graphics only one of which is assigned to the entity at any one time. This is accomplished with references to graphics in the ProModel Graphic Library (LibraryGraphicRecord) which can then be used at simulation runtime. Each of the graphics associated with this entity can be described using the <parts type><part><reference-keys/><reference-key><digital-files>. However the element also contains elements DiscretePartRecordWidth and DiscretePartRecordLength. The LibraryGraphicRecord contains elements GraphicLibraryWidth, GraphicLibraryHeight, GraphicLibraryRotation, GraphicLibraryHotSpot, GraphicLibraryPosition, and GraphicLibraryColor. These elements allow the user to customize how the binary

graphic file is displayed. These could be represented by the SDM if a <reference-frame-key> element were added such as <parts type><part><reference-frame-keys/>.

EntityrecordRotationType:
This data type contains an enumeration which describes the orientation of the entity as it travels through the system. This information is not necessary for simulation as it is related to the U.I. representation.

EntityRecordOtherCost:
Not currently implemented in ProModel.

3. Routing Records (Processing)
Supported
RoutingRecordEntityID:
This element contains a reference to the entity that is going to perform the process. This item is displayed in the processing table and is supported in <parts type><part>. Refer to the Entities section for more information.

RoutingRecordFromLocationID:
This element defines the location at which the processing of the element will occur and is supported in <machines><number/>. Refer to the Locations section for more information.

RoutingRecordPreempt:
This value specifies that a processing record is a preemption process record. This type of record overrides the default preemption behavior associated with location preemption and allows the user to control what will happen when a location attempts to be preempted. This element can be represented using <work><orders><order><order-definition><priority-rating>. The priority rating associated with an order could then be used to modify the processing of jobs or job tasks.

Unsupported Necessary
RoutingRecordOperation:
This element specifies any logic to be performed on an entity at the location defined in RoutingRecordFromLocationID. This logic could be mapped to the concept of <process-plans><process-plan><operation-sheets> in the SDM given that the ability to define logic is added to the operation-sheets object.

RoutingRecordToRouting:
This parent element contains nested elements that define destination information for the entity after the entity defined in RoutingRecordReference has been processed at the location defined in RoutingRecordFromLocationID. The concept can be mapped in the SDM to <process-plans><process-plan><routing-sheets> element.

RoutingRecordToRouting\RoutingRecordToRoutingRecordToLocation:
This element defines the destination location the entity is to route to after finished processing. The concept can be mapped in the SDM to <process-plans><process-plan><routing-sheets><routing-sheet><reference-key/> element, where the <reference-key/> element refers to a ProModel location.

RoutingRecordToRouting\RoutingRecordToRoutingRecordCondition:
This element allows for the specification of a condition by which the entity will route after finished processing. The concept can be mapped in the SDM to <process-plans><process-plan><routing-sheets><routing-sheet>, however an additional element would need to be added which allowed for logic to be defined that would specify dynamically which <routing-sheet> contained in <routing-sheets> should be used.

RoutingRecordToRouting\RoutingRecordToRoutingRecordDestExp:
This element allows for the use of an expression to specify a destination location the entity will route to after finished processing. The specification is made using a logic statement. Given the description provided previous for RoutingRecordToRouting\RoutingRecordToRoutingRecordToLocation this element could be mapped in the SDM if the <reference-key/> element, which refers to a ProModel location, could be specified using logic.

RoutingRecordToRouting\RoutingRecordToRoutingRecordLocationPriority:
The priority value defined here is assigned to a routing for access of the destination location. Priority is used to provide precedence when there are multiple entities attempting to access the same destination location. This idea could be supported with the addition of an element to <process-plans><process-plan><routing-sheets><routing-sheet><priority/>.

RoutingRecordToRouting\RoutingRecordToRoutingRecordEntityNameID:
This element is used to define the name of the entity as is exits a location after processing. This idea could be supported with the addition of a <new-name/> element to <process-plans><process-plan><routing-sheets><routing-sheet>.

RoutingRecordToRouting\RoutingRecordToRoutingRecordExitLogic:
This element contains logic to be executed before routing, but after finished processing at a location. ProModel refers to this as Move Logic. This idea could be supported with the addition of a <preroute-logic/> element to <process-plans><process-plan><routing-sheets><routing-sheet>.

RoutingRecordToRouting\RoutingRecordToRoutingRecordQuantity:
When entities exit a location they can be split into multiple items. This value specifies the number of items to exit a location per item entry. In the SDM the element <process-plans><process-plan><operation-sheets><operation-sheet><batch-size> refers to the number of parts to work on for a given task. A similar <exit-batch-size/> could be added to specify that the batch size increased as the result of an operation.

RoutingRecordToRouting\RoutingRecordToRoutingRecordProbability:
This element holds a probability to be applied to the routing to define how often it should be used in relation to other defined routings. This concept could be supported with the addition of a reference to an existing <probability-distribution> element to <process-plans><process-plan><routing-sheets><record-probability/> which could be used to determine which routing sheet to use.

RoutingRecordToRouting\RoutingRecordToRoutingRecordToExit:
This element defines if the entity is to be routed to exit the system as a result of using this routing. If the schema required that a default exit routing was always required in <process-plans><process-plan><routing-sheets> then the addition of an element possibly through the use of a <reference-key/> added to <process-plans><process-plan><operation-sheets> could be used to specify the use of the default route to exit <routing-sheets><routing-sheet>.

RoutingRecordToRouting\RoutingRecordToRoutingRecordConditionType:
This element holds the routing type. Available options are: First Available, Most Available, By turn, Random, If Join request, If Load request, If Send, Longest Occupied, Until Full, If Empty, Probability, User Condition, and Continue. If the options Probability or User condition are selected here, they use logic defined in RoutingRecordToRouting\RoutingRecordToRoutingRecordProbability or RoutingRecordToRouting\RoutingRecordToRoutingRecordCondition, respectively. Similar to the description for these items, this element could be represented with the addition of an element to <process-plans><process-plan><routing-sheets> which could be used to determine which routing sheet to use to provide the desired logical decision.

RoutingRecordToRouting\RoutingRecordToRoutingRecordUseDestExpr:
This element is a flag used to enable or disable the application of the expression found in RoutingRecordToRoutingRecordDestExp. The concept can be represented in the SDM with the addition of an element such as <process-plans><process-plan><routing-sheets><routing-sheet><use-destination-expression/>.

RoutingRecordToRouting\RoutingRecordToRoutingRecordBeginBlock:
This element is a flag used to define a new routing block. Routing blocks are logical structures used to group together routing information to be executed on entities as they route out of a location. This concept can be mapped in the SDM by using <process-plans><process-plan><routing-sheets>.

RoutingRecordToRouting\RoutingRecordToRoutingRecordNewEntity:
This element is a flag to designate that an element is to route out of a location as a new element. New elements contain a new set of statistical information. This concept could be represented with the addition of a flag such as <routing-sheets><routing-sheet><new/>.

RoutingRecordFromAll:
This value specifies that the defined processing record is to be applied to all available entities. If not set, the processing record only applies to one entity. This concept could be represented by the addition of an element such as <process-plans><process-plan><operation-sheets><operation-sheet-number>which would specify that the same operation-sheet is always used at a given task.

Unsupported Unnecessary
RoutingRecordToRouting\RoutingRecordToRoutingRecordPathPoints:
This element is used to store an array of path points that are used to define the graphical layout of routings in two dimensional space. Because it defines graphic elements it is not necessary. It is related to ReoutingRecordToRoutingRecordNumOfPoints.

RoutingRecordToRouting\RoutingRecordToRoutingRecordNumOfPoints:
This element keeps track of the number of points used to define the routing arrays defined in RoutingRecordToRoutingRecordPathPoints.

RoutingRecordToRouting\RoutingRecordToRoutingRecordResource:
Not currently implemented in ProModel.

RoutingRecordToRouting\RoutingRecordToRoutingRecordCrane:
Not currently implemented in ProModel.

RoutingRecordToRouting\RoutingRecordToRoutingRecordRobot:
Not currently implemented in ProModel.

RoutingRecordToRouting\RoutingRecordToRoutingRecordPathNetwork:
Not currently implemented in ProModel.

RoutingRecordToRouting\RoutingRecordToRoutingRecordConveyor:
Not currently implemented in ProModel.

RoutingRecordToRouting\RoutingRecordToRoutingRecordTime:
Not currently implemented in ProModel.

RoutingRecordToRouting\RoutingRecordToRoutingRecordResourcePriority:
Not currently implemented in ProModel.

RoutingRecordToRouting\RoutingRecordToRoutingRecordKeepResource:
Not currently implemented in ProModel.

RoutingRecordToRouting\RoutingRecordToRoutingRecordPointMoved:
Not currently implemented in ProModel.

4. Arrivals

Supported

ArrivalRecordEntityID:
This element contains a reference to the entity that is going to perform the arrival. Refer to the Entities section for more information. In the SDM this item can be mapped to the element found in <procurements><procurement><procurement-definition><procurement-items><procurement-item>.

ArrivalRecordLocationID:
This element defines the location at which the arrival of the element will occur. Refer to the Locations section for more information. The SDM mapping for this item could be contained in schedules which define work to be assigned to resources. This would be contained within the <schedules><schedule><resources-section> element.

ArrivalRecordQuantityPerArrival:
This element defines the quantity of arrivals that are contained per arrival occurrence. This element is represented in the SDM under the <schedules><schedule><work-section><order-section><order-schedule><orders><order><order-definition><order-items><order-item><order-item-definition><part-quantities/> element.

ArrivalRecordFirstArrival:
This element defines when the first arrival is to occur in the system for this arrival record. This element could be defined in the SDM using the <schedules><schedule> element which would define the first order arrival via a schedule contained in <work-section><order-section><order-schedule>.

Unsupported Necessary

ArrivalRecordFrequency:
This element defines how often the arrival is to occur. This field can contain a distribution. This element could be represented in the SDM under the <schedules><schedule><work-section><order-section><order-schedule><orders><order><order-definition> element. A new element would need to be added here that would allow for a distribution to be used to define the arrival. This might be done by expanding or modifying the <due-dates/> item using the existing <probability-distributions> SDM element.

ArrivalRecordNumberOfArrivals:
This element defines the number of arrivals that are to occur. This element could be represented in <work-section><order-section><order-schedule> section of the SDM by adding and element to specify the number of times an order is to be processed.

ArrivalRecordOperation:
The element defines logic to be performed when an entity arrives at a location. This logic could be contained within each SDM resource as a child work item to be executed whenever a part arrives. This could be mapped in the SDM to <tasks><task><task-definition><child-work-item>

ArrivalRecordCycleTableID:
This element is a reference into the arrival cycle table which defines patterns of arrivals into a system. It is beyond the score of this evaluation to examine the entire contents of cycle tables, however the concept of cycle tables could be defined in the SDM using the <schedules><schedule> element which would define the when order arrival are to occur via a schedule contained in <work-section><order-section><order-schedule>.

ArrivalRecordTimeOffset:
This element defines a numeric time to be used as a modifier to the value used to describe the time an element is to arrive in the system. In the SDM this concept could be defined in the <schedules><schedule> element to adjust the schedule time for which procurement is to occur.

ArrivalRecordVariation:
This element defines a time distribution to be used as a modifier to the value used to describe the time an element is to arrive in the system. In the SDM a distribution would need to be specified using the existing <probability-distributions> SDM element to define when a schedule <schedules><schedule> element should request a <procurement>.

ArrivalRecordTimeBasis:
This element describes the time units used to describe the arrival activity. The options are Time Only, Weekly Time, and Calendar Date. The SDM could support this concept with the addition of a new data element enumeration used in the <schedules> element. Currently the SDM does not contain an enumeration to support this set of options.

ArrivalRecordRepeatType:
This element describes how often the arrival cycle is to be repeated. Options are Daily or Weekly. This concept would be represented in the <schedules> section to specify how often a <procurement> is to occur. Currently the SDM does not contain an enumeration to support this set of options.

ArrivalRecordDisabled:
This element describes if the arrival record is to be used or not. In the SDM this concept could be contained in <schedules> which would enable or disable <procurements>.

The Six Common Simulation Elements

1. Path Networks
Supported
PathNetworkName:
This element supplies the specified name for a path network element. Supported in <layout><paths><path><name/>.

PathNetworkType:
This element is used to specify a type of path network. ProModel supports three separate types which are non-passing, passing, and crane. The SDM type field supplies an

adequate element to specify the mapping of this element; however additional types in the SDM would need to be added to support all of the ProModel equivalents used in <layout><paths><path><type/>.

PathNetworkNetworkIndex:
This element is a number that uniquely identifies the path network. The SDM contains an element <identifier/> that is an auto generated number to identify each element of this type supported in <layout><paths><path><identifier/>

Unsupported Necessary
PathNetworkNodes:
Nodes are points in the path network where path segments are joined together.
These specifications could be included in a sub element of <layout><paths><path>.

PathNetworkSegments:
Segment defines multiple individual line segments which together are what constitute an entire path network. Also includes elements which define the segment as unidirectional or bidirectional. These specifications could possibly be included in a sub element of <layout><paths><path>.

PathNetworkInterface:
A network interface is a node on a path network at which an object can get off or onto the path from a location. In this way it interfaces the path network with the location.

PathNetworkMappings:
Path network mappings contain elements to hold *from*, *to*, and *destination* values. When an object is moving through a system along a path network, these values are used to define which paths are used as the object travels. The *from* node defines where the object currently is and the *destination* node defines where the object is attempting to travel *to*. The *to* node is the next node the object should take in order to eventually arrive at the *destination* node. These mappings are calculated automatically by the ProModel compiler or they can be specified by the user to perform specific routings. These specifications could possibly be included in <layout><paths><path><path-route>.

PathNetworkCranes:
ProModel methodology implements cranes as special path networks along which a bridge and hoist travels. Within PathNetworkCranes there are three non user interface elements that would need to be mapped for comparable functionality to be supported across the mapping. PathNetworkCraneSpacing, PathNetworkCraneRailWidth, and PathNetworkCraneBridgeWidth define the spacing required between parallel cranes while in operation, the width of the crane path, and the width of the bridge that is riding on the crane rails. Within the SDM XSD specification there is a representation to support cranes found in <resources><cranes><crane>. This specification treats the crane similar to a <resource><stations><station> and contains an element to define crane specifications apply named <resources><cranes><crane><crane-specifications>. This specification contains detailed information such as the <manufacturer/>, <model-

number/>, <serial-number/>, <horsepower/>, <maximum-load-capacity/>, and <speed/> that could be used to define a machine by specification. If this <crane-specifications> element could be modified to include information about the size of the envelope within which the crane operates a mapping would be possible.

PathNetworkCalcSpeedByDist:
This element describes the speed at which objects travel along a path network is automatically calculated by ProModel according the graphic representation of the path in the ProModel user interface. However, the setting for speed can also be manually set according to a fixed amount of time for an object to travel from one point to another. This value keeps track of which of these two methods is used to determine the speed at which objects travel.

Unsupported Unnecessary
PathNetworkColor:
This value keeps track of the color of the path network.

PathNetworkInvisible:
This value keeps track if a path network is visible at simulation run-time.

PathNetworkRecompile:
At simulation runtime a pre-translation operation is performed to validate the integrity of all path networks defined in the ProModel UI. During this operation the exact path network routings that entities will take as they travel through the system are calculated. This information is then saved so that the time required for the pre-compile operation is avoided in subsequent simulation runs when the path network has not been modified. This value keeps track of whether or not to recompile due to a change in the path network record set. I implemented this feature in ProModel software as an optimization.

2. Resources
Supported
ResourceRecordIndex:
This element is a number that uniquely identifies the resource. The SDM contains an element <identifier/> that is an auto generated number to identify each element of this type supported in <machines><machine><identifier/>.

ResourceRecordName:
This element contains the name of the resource and is supported in <machines><machine><name/>.

ResourceRecordNote:
This element holds optional notes related to the resource. The element is supported in <resources><machines><machine><description/>.

ResourceRecordQuantity:
This element holds the number of units represented by this resource and ranges in values from 0-999. This element is represented in the SDM under <resources><machines><machine-group>. For multiple machines each machine could be defined within <machines><machine-group>.

ResourceRecordDiscreteParts:
This data type contains a reference to graphics in the ProModel Graphic Library (LibraryGraphicRecord) which are associated with each resource. The graphics associated with the data element DiscretePartRecordLibraryGraphic of this location can be described using <resources><machines><reference-keys/><reference-key><digital-files>. ResourceRecordDiscreteParts also contains DiscretePartRecordWidth and DiscretePartRecordLength. The LibraryGraphicRecord contains elements GraphicLibraryWidth, GraphicLibraryHeight, GraphicLibraryRotation, GraphicLibraryHotSpot, GraphicLibraryPosition, and GraphicLibraryColor. These data items are used to describe the size of the graphics as they are displayed in the ProModel UI. These could be represented by the SDM if a <reference-frame-key> element were added such as <resources><machines><reference-frame-keys/>.

Unsupported Necessary
ResourceRecordPickupTime:
This element represents the time required for a resource to pickup an entity. The SDM could support this idea with the addition of another element under the specification section such as <resources><machines><machine><machine-specifications><Pickup-time/>. This element could be specified using the enumeration units (time-duration-units).

ResourceRecordDepositTime:
The time required for a resource to deposit an entity. The SDM could support this idea with the addition of another element under the specification section such as <resources><machines><machine><machine-specifications><Deposit-time/>. This element could be specified using the existing SDM enumeration units (time-duration-units).

ResourceRecordAccelRate:
The element holds the acceleration rate of the resource in feet/sec^2. The SDM could support this idea with the addition of another element under the specification section such as <resources><machines><machine><machine-specifications><acceleration-rate/>. Presumably another data element, enumeration units (acceleration-units) would also need to be added to support this element.

ResourceRecordDecelRate:
The element is used to specify the deceleration rate of the resource in feet/sec^2. The SDM could support this idea with the addition of another element under the specification section such as <resources><machines><machine><machine-

specifications><deceleration-rate/>. Presumably, another data enumeration element unit (acceleration-units) would be added to support this element.

ResourceRecordFullSpeed:
The element is used to specify the speed of the resource when full in feet/minute. The SDM could support this idea assuming that resources can be mobile with the addition of another element under the specification section such as <resources><machines><machine><machine-specifications><full-speed/>. To support this specification presumably feet/minute would also be added to the existing data enumeration element units (speed-units).

ResourceRecordEmptySpeed:
The element is used to specify the speed of the resource when empty in feet/minute. The SDM could support this idea assuming that resources can be mobile with the addition of another element under the specification section such as <resources><machines><machine><machine-specifications><empty-speed/>. To support this specification presumably feet/minute would also be added to the existing data enumeration element units (speed-units).

ResourceRecordParkSearch:
If there is more than one resource assigned to a path network, each resource must have an associated list of nodes at which it can go down. This list is searched when the resource is to go down to find available capacity at a node to go down. This element is used for mobile resources such as <employees> or <stations> which are not supported in the current SDM specification.

ResourceRecordWorkSearch:
This element defines locations where the entity searches for work when it finishes its previous task. This concept could be represented by referencing location groups in <resources><machines><machine><machine-group><machine-keys/> which are keys to other machines that are contained within the same group. The reference would then need to be added to <parts type><part><work-search/>. The search behavior could then be performed on the machines in this group and would not blur the intended purpose of the machine group if the machine group matched where the ProModel resource would normally search for work.

ResourceRecordSearchForEntity:
When two or more entities of equal priority request a resource at the same time an operation called an entity search is performed to decide which entity will acquire the resource. The search follows rules defined in this element. Available options are Longest Waiting, Closest Entity, value defined in ResourceRecordMinAttrIndex, or the value defined in ResourceRecordMaxAttrIndex. This type of decision support could be included in the machine element <resources><machines><machine> with the addition of an enumeration to support the necessary options.

ResourceRecordNodeLogic:
This element defines logic to be performed as a resource enters and exits a path network via the specified node. Because the current SDM specification for path network items and mobile resources are both currently under development and subject to change, this item is undefined.

ResourceRecordHomeNodeIndex:
This element describes where the resource should go when idle and is a reference to a node on the layout. Because the current SDM specification for path network items and mobile resources are both currently under development and subject to change, this item is undefined.

ResourceRecordShiftNodeIndex:
This element describes where the resource should go when off shift and is a reference to a node on the layout. Because the current SDM specification for path network items and mobile resources are both currently under development and subject to change, this item is undefined.

ResourceRecordBreakNodeIndex:
This element describes where the resource should go when on a break and is a reference to a node on the layout. Because the current SDM specification for path network items and mobile resources are both currently under development and subject to change, this item is undefined.

ResourceRecordReturnToHome:
The element specifies that a resource return to a specified home node on the layout when it is no longer required. Because the current SDM specification for path network items and mobile resources are both currently under development and subject to change, this item is undefined.

ResourceRecordClockDowntime:
The element holds information that causes a resource to go down at a specific instant in simulation time. This parent element contains a child element ClockDowntimeRecordOperation which is used to define logic to execute while the resource is down. ResourceRecordClockDowntime also contains the child element ClockDowntimeRecordFrequency which contains a constant, distribution, or expression to express how often the downtime should occur.

ResourceRecordUsageDowntimes:
A usage downtime is based on how long a resource has been in operation. The element contains then element UsageDowntimeRecordOperation which contains logic to execute while the resource is down and UsageDowntimeRecordFrequency which can contain a distribution, or expression to express how often the downtime should occur.

ResourceRecordCostRate:
This element defines the cost per units of time defined in ResourceRecordCostTimeUnits. The SDM contains a <resources><machines><machine><hourly-rate/> which if changed to a more generic element name such as <rate/> could support this mapping. In addition another element would then need to be added to capture the units portion of <hourly-rate/>.

ResourceRecordCostPerUse:
This element defines the cost per use of a resource. This could be supported in the SDM with the inclusion of another element such as <resources><machines><machine><cost-per-use/>.

ResourceRecordCostTimeUnits:
The element defines the units associated with the cost. Available options are sec, min, hr, and day. The SDM contains a <resources><machines><machine><hourly-rate/> which if changed to a more generic element name such as <time-units> and supported the time-duration-units SDM enumeration could support this concept.

ResourceRecordStatistics:
This element is an enumeration that determines the type of statistics to keep track of for a resource. The options are None, Average, Standard Deviation, and Time-Series. To support this element the SDM would need to implement an enumeration with the necessary options such as <resources><machines><machine><statistics-type/>.

ResourceRecordSearchForResource:
When an entity that needs a resource must select from multiple available resources, it uses the rule defined by this element to determine which resource to select. Available options are closest resource, least utilized resource, and longest idle resource. To support this element the SDM would need to implement an enumeration with the necessary options such as <resourcesearch-type>. Then search functionality or some equivalent implantation would need to be added to the SDM.

ResourceRecordMinAttrIndex:
If selected, the search operation defined in ResourceRecordSearchForResource: uses the attribute defined in this element to select an entity with a minimum value of the specified attribute.

ResourceRecordMaxAttrIndex:
If selected, the search operation defined in ResourceRecordSearchForResource: uses the attribute defined in this element to select an entity with a maximum value of the specified attribute.

Unsupported Unnecessary
ResourceRecordResourcePoints:
The element defines points on the screen where resources are displayed when they travel to a node.

ResourceRecordOTCostRate:
Not currently implemented in ProModel.

ResourceRecordCapacity:
Not currently implemented in ProModel.

ResourceRecordLibraryGraphicRecord:
Not currently implemented in ProModel.

3. Shifts (Shift Assignments)

Supported

ShiftAssignmentRecords\ShiftAssignmentRecordItemLocationList:
This element contains a list of locations that are assigned to this shift assignment record. The SDM implements the association between shifts and locations (SDM stations) is accomplished using a reference in the <calendar> element which refers to <schedules><schedule><resources-section><stations-section>.

ShiftAssignmentRecords\ShiftAssignmentRecordItemResourceIDs:
This element contains a list of resources that are assigned to this shift assignment record. The SDM implements the association between shifts and resources (SDM machines) is accomplished using a reference key in the <calendar> element which refers to <schedules><schedule><resources-section><machines-section>.

ShiftAssignmentRecords\ShiftAssignmentRecordItemRecords:
This element contains a list of external shift files that are assigned to this shift assignment record. The SDM is designed to represent a shift internally using the <calendars><calendar><shift-schedule><shifts> element which negates the necessity of an external file reference.

Unsupported Necessary

ShiftAssignmentEndShiftPriority:
The element is used to specify the priority level that must be exceeded in order to prevent the location or resource from going off shift. The SDM could hold this value with the station or machines element with the addition of an <off-shift-prevent-priority> element to each of these elements.

ShiftAssignmentOffShiftPriority:
The element is used to specify the priority level that must be exceeded in order to bring a location or resource back online when it is currently offline. The SDM could hold this value with the station or machines element with the addition of an <off-shift-preempt-priority> element to each of these elements.

ShiftAssignmentStartBreakPriority:
The element is used to specify the priority level that must be exceeded in order to prevent the location or resource from going on break. The SDM could hold this value with the

station or machines element with the addition of an <start-break-prevent-priority> element to each of these elements.

ShiftAssignmentBreakPriority:
The element is used to specify the priority level that must be exceeded in order to bring a location or resource back online when it is currently on break. The SDM could hold this value with the station or machines element with the addition of an <off-break-preempt-priority> element to each of these elements.

ShiftAssignmentPreOffShiftLogic:
This element defines logic to be executed whenever a resource or location is scheduled to go off shift. This logic could be contained within the station or machine element with the addition of an <shift-pre-offshift-logic> element which would hold logic.

ShiftAssignmentOffShiftLogic:
This element defines logic to be executed whenever a resource or location is going off shift. This logic could be contained within the station or machine element with the addition of an <shift-offshift-logic> element which would hold logic.

ShiftAssignmentPreBreakLogic:
This element defines logic to be executed whenever a resource or location is scheduled to go off break. This logic could be contained within the station or machine element with the addition of an <shift-pre-break-logic> element which would hold logic.

ShiftAssignmentBreakLogic:
This element defines logic to be executed whenever a resource or location is going off break. This logic could be contained within the station or machine element with the addition of an <shift-break-logic> element which would hold logic.

ShiftAssignmentDisable:
This element specifies if the shift assignment is enabled or disabled. The SDM could specify this concept on a per location/station basis with the addition of a <shift-enabled> or <break-enabled> element to the shift referenced in <calendars><calendar><shift-schedule><shifts><shift>.

4. Scenarios

5. External Files

6. Streams

APPENDIX C

ProModel / NIST Mapping Table

Table C.1 ProModel to SDM XSD Mapping Comparison Table Legend

Support	Description
Y / N	Is element supported? (Yes or No)
Basic	This element is classified as a Basic Data Element.
Logic	This element contains Decision Logic.
Program	This element is a Basic Software Programming Element.
Layout	This element requires support from the NIST Layout Element.
Caveat	See description for a caveat that exists for this mapping.
*	A pointer in C programming language is used with this element.

Table C.2 ProModel to SDM XSD Mapping Comparison Table

ProModel XSD / NIST SDM XSD Element Names	Support
LOCATIONS	
LocationRecordName	Y
<resources><stations><station><name/>	
LocationRecordNote	Y
<resources><stations><station><description/>	
LocationRecordCapacity	Y-Caveat
<resources><stations><station><machine keys/><machine-specifications><workpiece-capacity/>	
LocationRecordCostRate	Y-Caveat
<resources><stations><station><hourly-rate/>	
LocationRecordLocationType	Y-Caveat
<resources><stations><station><type/>	
LocationRecordIndex	Y
<resources><stations><station><identifier/>	
LocationRecordMultiUnitQuantity	Y
<stations><station-group>	
LocationSetupDowntime	N-Logic
<resources><stations><station><station-status><work-assignments/><work-assignment><setup-definition-keys><probability-distribution>	

LocationRecordClockDowntimes	N-Logic
LocationRecordUsageDowntimes	N-Logic
LocationRecordCyclicDowntimes	N-Logic
LocationRecordCalledDowntimes	N-Logic
LocationRecordInputAttribute	N-Basic
<resources><stations><station><attribute/>	
LocationRecordOutputAttribute	N-Basic
<resources><stations><station><attribute/>	
LocationRecordCostTimeUnits	N-Basic
<resources><stations><station><rate-units/>	
LocationRecordStatistics	N-Basic
<resources><stations><station><statistics-type/>	
LocationRecordSelectionRule	N-Basic
<process-plans><process-plan><routing-sheets><routing-sheet><selection-rule>	
LocationRecordQueueRule	N-Basic
<routing-sheets><routing-sheet><queue-rule>	
LocationRecordAccessRule	N-Basic
<routing-sheets><routing-sheet><access-rule>	
ENTITIES	
EntityRecordName	Y
<parts type><part><name/>	
EntityRecordNote	Y
<parts type><part><description/>	
EntityRecordWidth	Y
<parts type><part><part-specifications><width/>	
EntityRecordLength	Y
<parts type><part><part-specifications><length/>	
EntityRecordInitialCost	Y
<parts type><part><part-specification><unit-cost/>	
EntityRecordPartType	Y
<parts type><part><type/>	
EntityRecordIndex	Y
<parts type><part><identifier/>	
EntityRecordStatistics	N-Basic

<parts type><part><statistics-type/>	
EntityRecordSpeed	N-Basic
<parts type><part><part-specifications><speed/>	
ROUTING RECORDS	
RoutingRecordEntityID	Y*
<parts type><part>	
RoutingRecordFromLocationID	Y*
<machines><number/>	
RoutingRecordPreempt	Y
<work><orders><order><order-definition><priority-rating>	
RoutingRecordOperation	N-Logic
<process-plans><process-plan><operation-sheets>	
RoutingRecordToRouting	N-Basic
<process-plans><process-plan><routing-sheets>	
RoutingRecordToRouting\RoutingRecordToRoutingRecordToLocation	N-Basic*
<process-plans><process-plan><routing-sheets><routing-sheet><reference-key/>	
RoutingRecordToRouting\RoutingRecordToRoutingRecordCondition	N-Logic
<process-plans><process-plan><routing-sheets><routing-sheet>	
RoutingRecordToRouting\RoutingRecordToRoutingRecordDestExp	N-Logic
RoutingRecordToRouting\RoutingRecordToRoutingRecordLocationPriority	N-Basic
<process-plans><process-plan><routing-sheets><routing-sheet><priority/>	
RoutingRecordToRouting\RoutingRecordToRoutingRecordEntityNameID	N-Basic*
<process-plans><process-plan><routing-sheets><routing-sheet><new-name/>	
RoutingRecordToRouting\RoutingRecordToRoutingRecordExitLogic	N-Logic
<process-plans><process-plan><routing-sheets><routing-sheet><preroute-logic/>	
RoutingRecordToRouting\RoutingRecordToRoutingRecordQuantity:	N-Basic
<process-plans><process-plan><operation-sheets><operation-sheet><exit-batch-size>	
RoutingRecordToRouting\RoutingRecordToRoutingRecordProbability	N-Basic
<process-plans><process-plan><routing-sheets><record-probability/>	
RoutingRecordToRouting\RoutingRecordToRoutingRecordToExit	N-Basic
<process-plans><process-plan><operation-sheets>	

RoutingRecordToRouting\RoutingRecordToRoutingRecordConditionType	N-Basic
<process-plans><process-plan><routing-sheets>	
RoutingRecordToRouting\RoutingRecordToRoutingRecordUseDestExpr	N-Basic
<process-plans><process-plan><routing-sheets><routing-sheet><use-destination-expression/>	
RoutingRecordToRouting\RoutingRecordToRoutingRecordBeginBlock	N-Basic
<process-plans><process-plan><routing-sheets>	
RoutingRecordToRouting\RoutingRecordToRoutingRecordNewEntity	N-Basic
<routing-sheets><routing-sheet><new/>	
RoutingRecordFromAll	N-Basic
<process-plans><process-plan><operation-sheets><operation-sheet-number>	
ARRIVALS	
ArrivalRecordEntityID	Y*
<procurements><procurement><procurement-definition><procurement-items><procurement-item>	
ArrivalRecordLocationID	Y*
<schedules><schedule><resources-section>	
ArrivalRecordQuantityPerArrival	Y
<schedules><schedule><work-section><order-section><order-schedule><orders><order><order-definition><order-items><order-item><order-item-definition><part-quantities/>	
ArrivalRecordFirstArrival	Y
<work-section><order-section><order-schedule>	
ArrivalRecordFrequency	N-Basic
<schedules><schedule><work-section><order-section><order-schedule><orders><order><order-definition>	
ArrivalRecordNumberOfArrivals	N-Basic
<work-section><order-section><order-schedule>	
ArrivalRecordOperation	N-Logic
<tasks><task><task-definition><child-work-item>	
ArrivalRecordCycleTableID	N-Basic*
<work-section><order-section><order-schedule>	
ArrivalRecordTimeOffset	N-Basic
<schedules><schedule>	
ArrivalRecordVariation	N-Basic

<schedules><schedule>	
ArrivalRecordTimeBasis	N-Basic
<schedules>	
ArrivalRecordRepeatType	N-Basic
<schedules>	
ArrivalRecordDisabled	N-Basic
<schedules>	
PATH NETWORKS	
PathNetworkName	Y
<layout><paths><path><name/>	
PathNetworkType	Y-Caveat
<layout><paths><path><type/>	
PathNetworkNetworkIndex	Y
<layout><paths><path><identifier/>	
PathNetworkNodes	N-Layout
<layout><paths><path>	
PathNetworkSegments	N-Layout
<layout><paths><path>	
PathNetworkInterface	N-Layout
PathNetworkMappings	N-Layout
<layout><paths><path><path-route>	
PathNetworkCranes	N-Basic
<resources><cranes><crane>	
PathNetworkCalcSpeedByDist	N-Basic
<layout><paths><path>	
RESOURCES	
ResourceRecordIndex	Y
<machines><machine><identifier/>	
ResourceRecordName	Y
<machines><machine><name/>	
ResourceRecordNote	Y
<resources><machines><machine><description/>	
ResourceRecordQuantity	Y
<resources><machines><machine-group>	
ResourceRecordDiscreteParts	Y-Caveat
<resources><machines><reference-keys/><reference-key><digital-files>	
ResourceRecordPickupTime	N-Basic
<resources><machines><machine><machine-specifications><Pickup-	

time/>	
ResourceRecordDepositTime	N-Basic
<resources><machines><machine><machine-specifications><Deposit-time/>	
ResourceRecordAccelRate	N-Basic
<resources><machines><machine><machine-specifications><acceleration-rate/>	
ResourceRecordDecelRate	N-Basic
<resources><machines><machine><machine-specifications><deceleration-rate/>	
ResourceRecordFullSpeed	N-Basic
<resources><machines><machine><machine-specifications><full-speed/>	
ResourceRecordEmptySpeed	N-Basic
<resources><machines><machine><machine-specifications><empty-speed/>	
ResourceRecordParkSearch	N-Layout
ResourceRecordWorkSearch	N-Basic
<parts type><part><work-search/>	
ResourceRecordSearchForEntity	N-Basic
<resources><machines><machine>	
ResourceRecordSearchForResource	N-Basic
<parts type><part><resourcesearch-type>	
ResourceRecordHomeNodeIndex	N-Layout
ResourceRecordShiftNodeIndex	N-Layout
ResourceRecordBreakNodeIndex	N-Layout
ResourceRecordReturnToHome	N-Layout
ResourceRecordClockDowntime	N-Logic
ResourceRecordUsageDowntimes	N-Logic
ResourceRecordCostRate	N-Basic
<resources><machines><machine><rate/>	
ResourceRecordCostPerUse	N-Basic
<resources><machines><machine><cost-per-use/>	

ResourceRecordCostTimeUnits	N-Basic
<resources><machines><machine><time-units/>	
ResourceRecordStatistics	N-Basic
<resources><machines><machine><statistics-type/>	
ResourceRecordMinAttrIndex	N-Basic
ResourceRecordMaxAttrIndex	N-Basic
SHIFTS	
ShiftAssignmentRecords\ShiftAssignmentRecordItemLocationList	Y
<schedules><schedule><resources-section><stations-section>	
ShiftAssignmentRecords\ShiftAssignmentRecordItemResourceIDs	Y
<schedules><schedule><resources-section><machines-section>	
ShiftAssignmentRecords\ShiftAssignmentRecordItemRecords	Y
<calendars><calendar><shift-schedule><shifts>	
ShiftAssignmentEndShiftPriority	N-Basic
<off-shift-prevent-priority>	
ShiftAssignmentOffShiftPriority	N-Basic
<off-shift-preempt-priority>	
ShiftAssignmentStartBreakPriority	N-Basic
<start-break-prevent-priority>	
ShiftAssignmentBreakPriority	N-Basic
<off-break-prempt-priority>	
ShiftAssignmentPreOffShiftLogic	N-Logic
<shift-pre-offshift-logic>	
ShiftAssignmentOffShiftLogic	N-Logic
<shift-offshift-logic>	
ShiftAssignmentPreBreakLogic	N-Logic
<shift-pre-break-logic>	
ShiftAssignmentBreakLogic	N-Logic
<shift-break-logic>	
ShiftAssignmentDisable	N-Basic
<calendars><calendar><shift-schedule><shifts><shift>	
SCENARIOS	
	N-Logic
EXTERNAL FILES	
<reference-key>	Y
STREAMS	
	N-Logic

ATTRIBUTES	
<parts><part><attribute/>	N-Program
VARIABLES	
	N-Program
ARRAYS	
	N-Program
MACROS	
	N-Program
SUBROUTINES	
	N- Program