NISTIR 7256

A Simulation and Gaming Architecture for Manufacturing Research, Testing, and Training

Charles R. McLean
Sanjay Jain
Y. Tina Lee
Frank Riddick

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

NISTIR 7256

A Simulation and Gaming Architecture for Manufacturing Research, Testing, and Training

Charles R. Mclean
Manufacturing Systems Integration Division
National Institute of Standards and Technology
Gaithersburg, MD 20899-8260

Sanjay Jain
Department of Management Science, School of Business
The George Washington University
Washington, DC 20052

Y. Tina Lee
Frank Riddick
Manufacturing Systems Integration Division
National Institute of Standards and Technology
Gaithersburg, MD 20899-8260

September 2005

U.S. DEPARTMENT OF COMMERCE
Carlos M. Gutierrez, Secretary
TECHNOLOGY ADMINISTRATION
Michelle O'Neill, Acting Under Secretary of Commerce for Technology
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
William Jeffrey, Director

Table of Contents

A Simulation and Gaming Architecture
For Manufacturing Research, Testing, and Training

Abstract:
Manufacturing systems are often costly to develop and operate. Simulation technology has been demonstrated to be an effective tool for improving the efficiency of manufacturing system design, operation, and maintenance. But manufacturing simulations are usually developed to address a narrow set of industrial issues, e.g., the purchase of new equipment or the modification of a manufacturing process. Once the analysis is complete a particular simulation model may not be used again. If simulations could be made more modular and easily integrated, they could have tremendous value as tools for manufacturing research, testing, and training. With the incorporation of video game technology, simulations could be created that would enable researchers and students to get near real-world experiences with manufacturing. In order to create a realistic virtual manufacturing environment, a number of different types of systems will need to be modeled and integrated. This paper presents a modular reference architecture to facilitate the integration of manufacturing simulation and gaming systems. Opportunities for research, testing, and training are also discussed that will be enabled by the implementation of the architecture.

Keywords: virtual manufacturing, simulation, video games, serious games, software architectures, information models, interface standards

1. Introduction

Manufacturing systems tend to be large, complex, and expensive to construct and operate. Due to hardware-acquisition, maintenance, and space costs, academic and research institutions cannot afford to duplicate real manufacturing systems in their laboratories. Student and researcher hands-on experiences with manufacturing systems are often limited to individual or small groups of machine tools in laboratory shops, prototype work cells, or tabletop manufacturing systems. Engineering schools often arrange cooperative extension or intern programs with manufacturing facilities so that students may get hands-on experience with real production systems while working towards their academic degrees. Although the student or researcher typically benefits personally, these experiences are often narrow in scope. Furthermore, the educational institution often gets little direct benefit or useful knowledge from the student’s work experiences.

Manufacturing training, experimentation, and testing could be significantly enhanced if manufacturing systems could somehow be brought into the laboratories of academic and research institutions. Computer simulation and video game technology now allow us to construct large, realistic virtual worlds in software. The military and the entertainment industry have made extensive use of this technology for a number of years. The corporate and academic world is just beginning to recognize the potential of this technology as a training tool. Virtual manufacturing enterprises could be used by a variety of organizations involved in manufacturing for training, experimentation, and testing. This paper focuses on what this technology is and how it might be put to effective use by manufacturing industry and supporting institutions.

Although simulations have been used as instructional tools in the past, this paper suggests that these tools can be significantly enhanced through the addition of and integration with video-game technology. The paper presents an architecture for manufacturing research, testing, and training based upon simulation, virtual reality, and video game technology, as well as actual commercial manufacturing software and hardware systems. The architecture is being implemented at the U.S. National Institute of Standards and Technology (NIST) as part of the Virtual Manufacturing Enterprise Project within the Manufacturing Interoperability Program. The systems under development at NIST will be used primarily as research tools for testing and evaluating interface specifications and standards.

The paper provides background on video game technology (Section 2), summarizes opportunities for using a simulation and game-based manufacturing environment for industrial research, testing, and training (Section 3), and provides an overview of the proposed simulation and gaming system architecture (Section 4).

2. Why Use Video Game Technology?

Video game technology could be a powerful tool for training the manufacturing workforce. Game technology promises to enhance the educational process by providing a more engaging learning experience than traditional classroom or computer-aided instruction methods alone. Video game technology could be used to create virtual environments for the student. These environments would contain realistic three-dimensional graphics and sound that could significantly enhance the learning experience. Training applications might address theory, development, operation, and maintenance of manufacturing systems.

Video game engines provide integrated environments for creating virtual worlds. Engines typically have capabilities for creating three-dimensional graphics, sound, animated characters, intelligent character behaviors, and various physical phenomena. They may also provide support for the creation of user-level game modifications, Web-based software distribution, and multi-player game interactions over the Internet.

Given market demands, it is not surprising that computer games have evolved in a rather haphazard fashion over the past 25 years. Due to language barriers and regional interests, individual game titles have been targeted to specific sectors in the international marketplace. Furthermore, games had to be tailored to run on a wide variety of platforms including personal computers, game consoles, handheld game players, palm computers, and, more recently, cell phones. Each of these computing devices had different performance characteristics, display screen resolutions, and user input mechanisms. For a graphical history of the evolution of video games, see [Demaria 2004]. For some predictions about the future of the video game industry, see [Rollings 2004]

Although games were largely produced for entertainment in the past, there has been more and more interest in using game technology for serious purposes. Recently, a number of organizations and conferences have sprung up that are focused on educational and training applications of video games technology. Some examples include the Serious Games Summit [Vargas 2004] and [Serious 2005], the MIT Education Arcade [MIT 2005] at the Electronics Entertainment Expo (E3), and the U.S. Navy's Naval Education and Training Command (NETC) Learning Strategies Consortium [Learning 2004].

Given the diversity of game titles, computing platforms, and tremendous deadline pressures that are often placed on game developers, it is not surprising that their software development process has been somewhat chaotic. Within the video game industry, there has always been a rush to get the next game to market before the customer's interests or technology changes. Experienced game developers acknowledge that good software engineering practices were seldom used in the past [Rollings 2004]. Little thought was typically given to code re-use. Game programmers spend considerable time and effort re-developing code that they or others may have written before. Standards that are purely game-related are non-existent; although the game development community makes considerable use of graphics and sound standards developed by related industries. For these reasons, games, supporting software, and hardware (peripheral devices, consoles, etc.) are often incompatible with each other. Only recently has there been a trend towards separating the code that is specific to a particular game title, the common elements of the game engine, the hardware computer platform, and its peripheral devices.

Video games are usually very complex software systems. Game software has elements of graphics visualization systems, sound processors, computer operating systems, compilers, text editors, networked client-server environments, artificial intelligence, physics modeling systems, as well as other functionality. Since games are often intended to create realistic visual environments for the user, the motions of game characters and other animated objects must appear to be smooth. Much of the data in the game world needs to be updated at video frame rates (i.e., television screen refresh rates of 30 times per second or faster). As such, a considerable amount of complex information processing must be carefully orchestrated during each second of game execution. Due to these demands, game developers have been more concerned with performance and appearance of the game than following good software engineering practices.

There are problems that make it difficult to adapt game software quickly for new purposes. In the past, games were monolithic software systems. Each game typically involved considerable custom software development. A single publisher or small development team was responsible for creating all of the software modules that comprised a game title. The term "game engine" was often used to refer to the entire body of software associated with a single title. If the industry is to become more efficient and achieve its growth potential in the future, title-specific content should be separated from the generic processing functions needed to implement a variety of games. The code required to support the generic processing functions should be consolidated into a general-purpose game engine that can be re-used. Title-specific code should be the only custom software development required and it should not require low-level coding.

This division of functionality and responsibility will help the market to grow by allowing content developers to work independently. Whereas today, a development company may need to have expertise in both low-level game engine programming and content development, in the future that need not be so. Similar product specialization, standard interfaces, and division of labor have enabled the personal computer industry to undergo tremendously rapid growth. Specialization with the game and learning-content industry has similar potential. Today, a number of companies are now marketing or developing game engines for the PC. Some companies are actually selling components of a game engine, e.g., physics and graphics modules. The licensing cost per seat (i.e., development user station) of these engines or component modules varies from several hundred dollars to hundreds of thousands of dollars. Section 4.2 of this paper discusses a modular architecture for a core game engine that will allow different software vendors to develop plug-compatible component modules independently.

The next section discusses how the manufacturing community could benefit from the use of simulation and gaming technology.

3. Opportunities for Industrial Research, Testing, and Training

Simulation technology enables the construction of technically correct, dynamic models of organizations, systems, and processes. The models, once validated, can be used for supporting decisions for design and operation of the systems to achieve desired performance. Gaming technology provides user interfaces, interaction mechanisms, and human modeling capabilities that support the creation of immersive experiences in virtual worlds. It thus provides the means to train operator and decision makers allowing them to get familiarized with the machine or manufacturing system and understand the potential consequences of their decisions and actions without suffering actual costs, lost time, injury, equipment damage, etc. The subsections that follow discuss how these technologies could support industrial research, testing, and training.

3.1 Research

Research applications include those that may be used by academia or the manufacturing research and development community. Applications include the development of models of new manufacturing systems, policies, procedures, processes, and algorithms. Simulation can be a powerful tool for conducting research experiments in manufacturing. In The Handbook of Simulation, Jerry Banks defines simulation as:

> *"…the imitation of the operation of a real-world process or system over time. Simulation involves the generation of an artificial history of the system and the observation of that artificial history to draw inferences concerning the operational characteristics of the real system that is represented. Simulation is an indispensable problem-solving methodology for the solution of many real-world problems. Simulation is used to describe and analyze the behavior of a system, ask what-if questions about the real system, and aid in the design of real*

> *systems. Both existing and conceptual systems can be modeled with simulation." [Banks, 1998]*

Manufacturing simulation focuses on modeling the behavior of manufacturing organizations, processes, and systems. Organizations, processes and systems include supply chains, as well as people, machines, tools, and information systems. Simulation models can be used to study parts of the manufacturing organization, sub-systems or individual processes as needed to evaluate their performance. The individual models can be integrated to create a virtual factory that can be used to support decisions through manufacturing system life cycle [Jain 2001]. The research applications of manufacturing simulation include system design and operation, as described below:

Manufacturing system design

- Utilize hybrid optimization-simulation to support selection of design parameters such as locations of distribution centers and manufacturing plants, number of machines, grouping of machines, and products
- Select new production systems, equipment, and processes
- Evaluate the impact of new systems on overall business performance
- Analyze layouts and flow of materials within production areas, lines, and workstations
- Perform capacity planning analyses
- Determine production and material handling resource requirements
- Visualize the form, fit, and function of products, as well as the processes and systems used to manufacture those products
- Perform ergonomic analysis of manual tasks and work area layout.

Manufacturing system operation

- Evaluate new operating policies by modeling "as-is" and "to-be" manufacturing and support operations from the supply chain level down to the shop floor
- Evaluate improvements in business processes supporting manufacturing by modeling "as-is" and "to-be" for order processing, shipping, invoicing, etc.
- Optimize process parameters for existing and new products using models of physics of the processes such as machining, injection molding, sheet metal forming, and semiconductor fabrication
- Evaluate resource allocation, scheduling and dispatching algorithms
- Evaluate use of simulation to support generation and evaluation of plans and schedules.

Simulation models may also be built to support decisions regarding investment in new technology, expansion of production capabilities, modeling of supplier relationships, materials management, human resources, etc.

The research applications require the simulation models to be technically accurate commensurate with the level of detail. The system-level applications should focus on use of stochastic factors to ensure evaluation of the artifacts in realistic situations. The models should be validated carefully before their use for supporting research application such as those listed above. The simulation application may also be integrated with other input and output analysis tools to ensure statistical validity. In addition, integration with applications such as design of experiments would help guide the development of the artifact that is the objective of the research.

Availability of a test bed that is based on the proposed architecture would help researchers focus on their key objectives. They would not have to spend substantial amounts of time building and integrating the needed simulation models. Based on the application, the researchers can select the relevant set of simulation and gaming modules that should be brought together to test the development. For example, for evaluating a scheduling algorithm, the researchers may pull together the manufacturing system model and associated supply chain modules. For evaluating a new process, the relevant machine models may be integrated with models of the relevant parts of the manufacturing system to understand the feasibility of individual steps and the impact on material flow. Documented procedures for validation and experimentation would further help the users of the test bed.

3.2 Testing

Testing applications include the use of simulation primarily by operations personnel or operations support engineers to test new methods, processes, and equipment before integrating them into operations. Testing applications are also applicable at design and prototype stages. Some of the testing applications may appear to duplicate the research applications, but they are typically carried out when the product or manufacturing system design is nearly finalized. Sometimes the testing is carried out by using the same applications as used for research but by personnel other than those directly involved in the development. Use of video game technology integrated with simulation may allow engineering students and faculty members to experience, analyze, and study the testing procedures.

Integrated simulation and video game technology can be used to support the following testing applications in the manufacturing domain.

- Perform interoperability testing with models of systems being integrated. For example a model of a robot controller may be integrated with a model of the robot for testing purposes to ensure interoperability.
- Perform interoperability testing with emulated physical equipment. For example, a physical programmable logic controller may be tested with an emulated conveyor system before the physical conveyor system is installed or even delivered.
- Evaluate the capability of the delivered process, system or design to meet interface specifications.
- Perform conformance and acceptance tests using simulations to create the specified range of inputs for a delivered system or process.
- Evaluate whether new systems, processes or designs meet performance specifications, for example, test program for robots and other machinery using simulations.
- Develop metrics to allow the comparison of predicted performance against "best in class" benchmarks to support continuous improvement of manufacturing operations.

- Evaluate the manufacturability of new product designs.
- Test the usability of a system using its representation through immersive video gaming technology.

Testing applications also require gaming and simulation representations to be technically correct. Again the models need to be carefully validated, however, the procedures used may be more focused on functional and deterministic validation rather than statistical validation used for system-level research applications that use stochastic factors. The validation procedures should be defined to ensure common practices. Supporting applications that exercise the models through the range of parameters defined in the specifications should be provided to facilitate the process.

Associated development of test cases and procedures would help by allowing a common scale on which alternate artifacts can be tested. Vendors of artifacts can use the results from standard test cases to highlight their products. Customers can use the results from the standard test cases for initial screening of vendors and then proceed with testing using company specific data. The test bed with associated test procedures and test cases will benefit both researchers and industrial personnel due to large reduction in effort for testing of new artifacts. Researchers from academic world will gain a better understanding of industrial strength systems and testing. Researchers and developers from manufacturing and vendor organizations would gain by unbiased testing of the developed and delivered artifacts. Finally, operations personnel would gain the ability to perform objective testing and savings of time involved.

3.3 Training

Training applications include the use of simulation and video game technology to increase familiarity of the human operators and decision makers with the artifact and provide accelerated experience through exposure to a range of potential behaviors of the artifact that may be generated based on their interactions with it. These applications are used typically just before the trainees are expected to interact with the new artifacts in their areas of responsibility. For example, a scheduler may be trained on setting parameters for scheduling software based on the daily inputs of status, orders, resource availabilities, and performance reports that are generated using simulations based on the inputs. A machine operator may be trained on setting the feed and rotation rates of the machine through visualizing the impact on the feed stock through a game application and perhaps feeling the vibrations of the machine through force feedback devices.

The training applications should be useful to industrial personnel in preparing them for delivering improved performance on the job. The applications will help them understand the options and the impact of decisions for a wide variety of scenarios. The training applications should be equally, if not more, useful to researchers and students for exposing them to industrial environments to ensure that their research addresses real problems and that students are well prepared for the industry. It will greatly increase the experience level and readiness of students to tackle a wide range of problems. These applications have the potential to improve industry performance by reducing the learning curves of the workforce and management on one hand and by helping the researchers provide better solutions on the other.

Training applications that would gain from use of integrated simulation and gaming technology include:

- Improve process and system visualization for technical and management personnel responsible for them.
- Train operators on controlling machinery and equipment.
- Train production and support staff on systems and processes.
- Provide manufacturing management staff an environment to test strategies for issues such as product designs, manufacturing process and system designs, supply chain partnerships, make-buy decisions, and inventory management.
- Provide realistic substitute for in-depth business and technical experience with real systems and the issues associated with their design, operation, and maintenance to researchers and students.
- Provide opportunities for students and researchers to deal with product life cycle process, systems engineering, management issues, supply chain formation, operation and maintenance.

The training applications should be supported by well-designed training scenarios that target a range of users and delivery options. The options may include training a single user learning on his own interacting with a computer, multiple users working independently in a computer lab setting with a facilitator, and multiple users working as a team. The scenarios should support different user roles ranging from a supply chain manager to an operator on the shop floor and the range of skills associated with each role.

Similar to research and testing applications, the training applications require simulations that are technically correct and valid. It has to be noted though that training scenarios would seldom include controlled multiple replications and associated statistical rigor for simulation output analysis. The focus in training is to ensure generation of correct technical response for the actions and decisions taken by trainees. For complex, system level decisions and other artifacts involving random factors, the training scenarios would need to communicate the importance of understanding that the generated behavior is one among several potential outcomes. In such scenarios, the importance of multiple training sessions with similar settings should be stressed. Such issues should be addressed using defined training procedures associated with each training scenario.

This section described the opportunities for integrated gaming and simulation for research, testing and training in manufacturing domain. Unfortunately, traditional manufacturing simulators seldom provide much opportunity for the user to become engaged in real-time interaction or immersion in the simulated environment. Each simulation vendor has had to develop its own graphical user interface or display system. The level of functionality of these display systems has varied considerably from vendor to vendor. Few manufacturing system simulators come close to achieving the level of graphics of most current video games. Most game system vendors have excelled in the creation of sophisticated graphical interfaces. Gaming technology could be used to provide a consistent, high quality, sharable, and re-usable front-end to manufacturing simulators.

The next section presents an overview of the proposed reference architecture.

4. An Integrated Reference Architecture

The purpose of a reference architecture is to identify the major modules, module functions, and interfaces for a software system. The proposed manufacturing simulation and gaming architecture defines a distributed system that may have three major groups of subsystems, i.e., gaming, simulation, and real manufacturing subsystems. Gaming provides users with a high quality, multimedia, role-playing environment. Simulation provides technically correct models of human and manufacturing organization behavior, systems, and processes. Real manufacturing systems provide a capability for connecting and using real manufacturing systems alongside the virtual gaming and simulation subsystems. The architecture allows the integration of real manufacturing systems with gaming and simulation software, but a discussion of those integration issues is beyond the scope of this paper. Figure 1 illustrates the next level of decomposition of the architecture into clusters of gaming and simulation systems.

Techniques for integrating gaming and simulation subsystems have evolved independently over the years. Within the simulation world, the High Level Architecture (HLA) has been used to integrate distributed simulation systems. Within the gaming world, massively multiplayer online gaming (MMOG) or massively multiplayer online role-playing game (MMORPG) architectures have been used to integrate distributed gaming systems. While HLA has become a standard [IEEE 2000], the gaming world has not standardized its architecture. Each integration approach has its strengths and weaknesses that will be addressed later in this paper. Our approach has been to accept the architectures of both communities and define a bridge between them. The bridge handles time synchronization and data transfer between the two groups of subsystems.

4.1 Manufacturing Simulation Subsystem

The manufacturing simulation subsystem is composed of a distributed collection of simulation applications and integration mechanisms that allow the applications to work together. Why is the architecture based upon a distributed set of manufacturing simulations rather than a single monolithic one? A distributed approach increases the functionality of simulation enabling users to do things that they could not do with a monolithic system. For example, a distributed approach would allow users and vendors to:

- utilize simulation tools developed by different software developers that are specialized to model specific aspects of manufacturing. Individual simulation-vendor's products do not provide the capabilities to model all areas of interest
- allow a vendor to hide the internal workings of a simulation system through the creation of run-time simulators with limited functionality
- provide simultaneous access to executing simulation models for users in different locations (collaborative work environments)

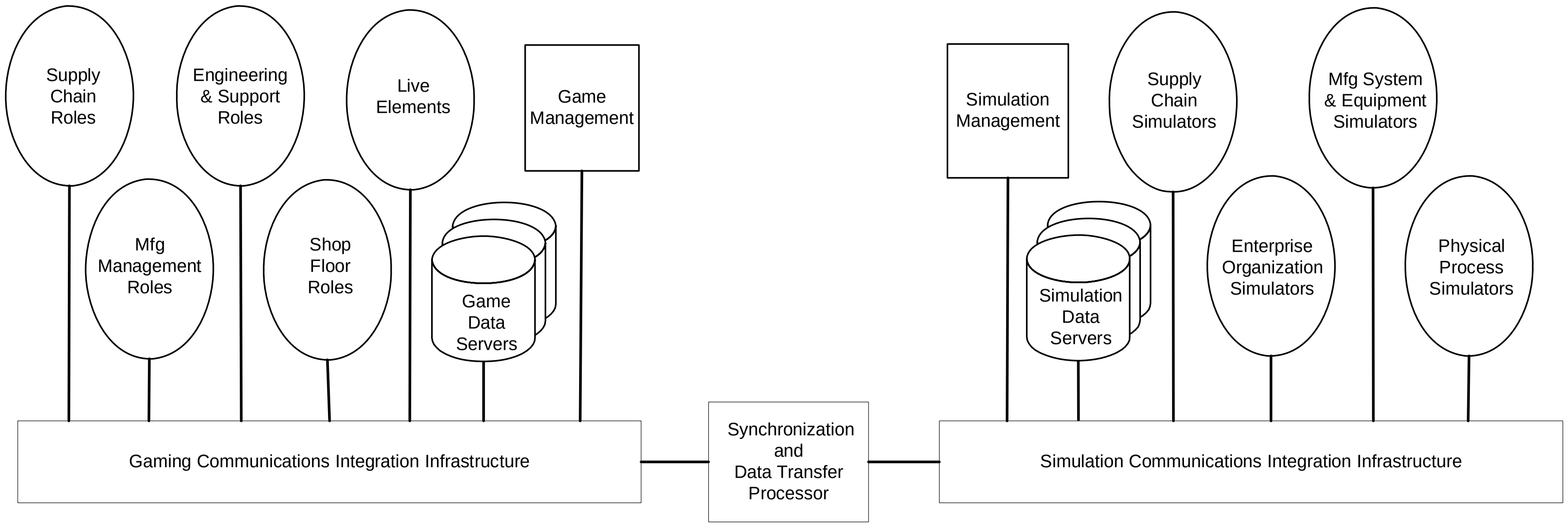
Supply Chain Roles
Mfg Management Roles
Engineering & Support Roles
Shop Floor Roles
Live Elements
Game Data Servers
Game Management
Gaming Communications Integration Infrastructure
Synchronization and Data Transfer Processor
Simulation Communications Integration Infrastructure
Simulation Management
Simulation Data Servers
Supply Chain Simulators
Enterprise Organization Simulators
Mfg System & Equipment Simulators
Physical Process Simulators

Figure 1. Decomposition of the gaming and simulation elements of the architecture.

- take advantage of additional computing power, specific operating systems, or peripheral devices (e.g., virtual reality interfaces) afforded by distributing across multiple computer processors
- offer different types and numbers of software licenses for different functions supporting simulation activities (model building, visualization, execution, analysis).
- create an array of low-cost, run-time, simulation models that can be integrated into larger models
- model supply chains across multiple businesses where some of the information about the inner workings of each organization may be hidden from other supply chain members
- simulate multiple levels of manufacturing systems at different degrees of resolution such that lower level simulations generate information that feeds into higher levels.

Figure 2 shows the manufacturing simulation subsystem of the reference architecture. The elements of the architecture include clusters of simulation applications, data servers, an integrated infrastructure, and a simulation management module.

4.1.1 Simulation Applications

Simulation applications will be used to model the behavior of real manufacturing systems. Several clusters of manufacturing simulators are envisioned. Each cluster and possible simulation applications are briefly introduced below.

- *Supply chain simulators* can be used to model the organization and management of supply chains. Organizations that may be simulated include supply chain headquarters, manufacturing primes, suppliers, transportation networks, warehouses, distribution centers, retailers, and customers. Some of the issues that may be addressed include lead times, inventory levels, production capacity, operations under surge conditions, and information flows.
- *Enterprise organization simulators* can be used to model the internal business processes of various departments within the manufacturing organization, such as customer order servicing, design, engineering, production, and inventory management. Business process modeling techniques may be used to analyze order flow and processing times in order to streamline operations and minimize non value-added functions.
- *Manufacturing system and equipment simulators* can be used to model the normal operations, failure modes, and maintenance of various manufacturing equipment, such as fabrication, assembly, material handling, quality, and packaging systems. Examples of some of the equipment making up these systems includes machine tools, coordinate measuring machines, robots, storage and retrieval systems, and conveyors. Discrete event simulation techniques may be used to analyze operation times, capacity, queue lengths, bottlenecks, buffer storage requirements, inventory levels, etc.
- *Physical process simulators* can be used to create accurate models of the physical transformations that products and tooling undergo in various manufacturing industries. Industries that will have unique process simulations include metalworking, electronics, food, textiles, plastics, and chemicals/refining. For example, a physical

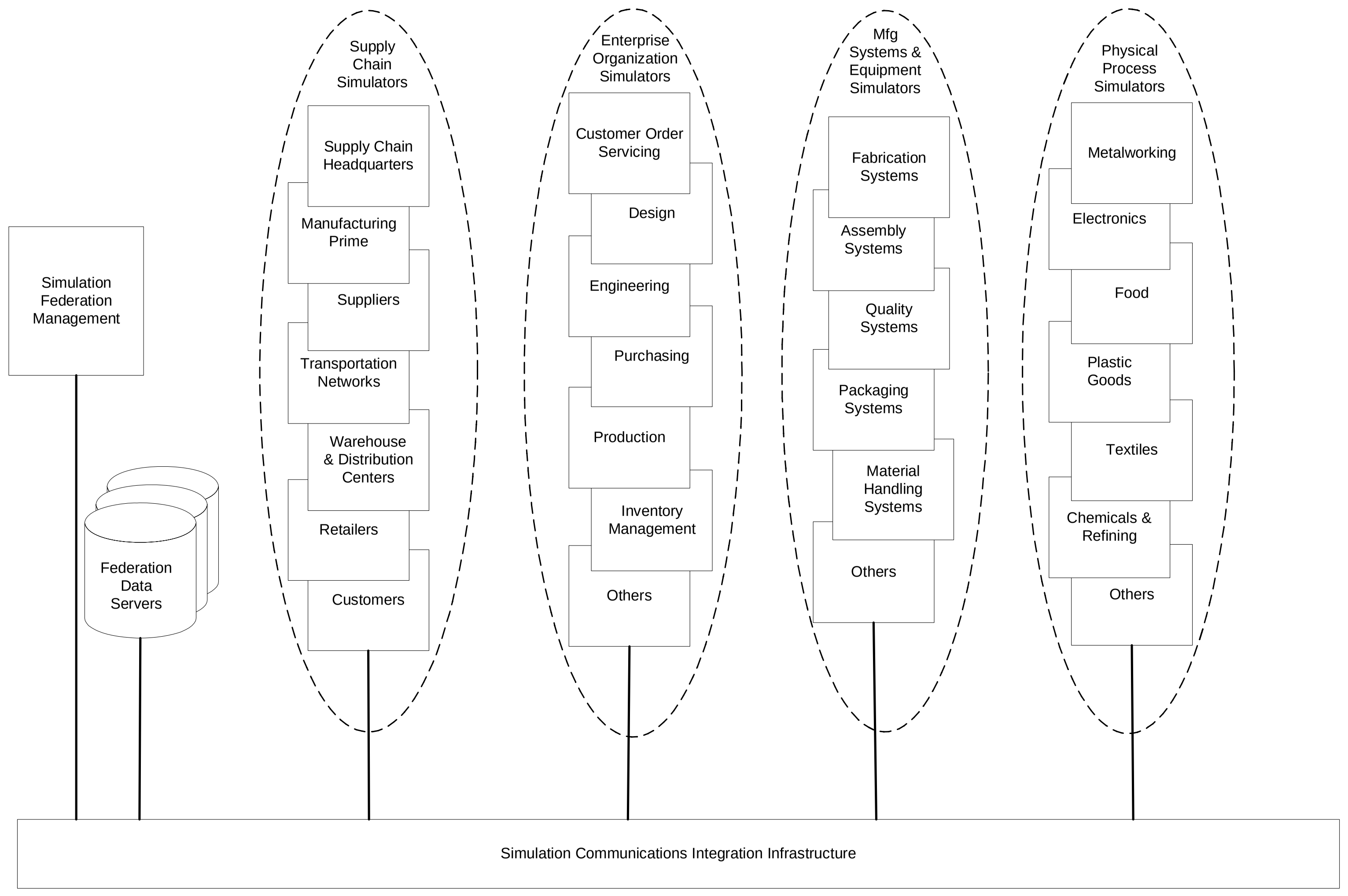

Figure 2. Manufacturing simulation subsystem: simulation clusters, simulation federation management, data servers, and communications infrastructure.

process simulator for metalworking may model processes associated with a machine tool's operation. Information obtained from the simulation may include changes to work piece geometry, chip formation, tool wear, chatter, thermal and mechanical variations to the machine.

Although a simulation may be developed as a custom software application, there are many commercial simulation engines available that support the types of simulation identified above. The next section discusses the topic of simulation engines.

4.1.2 Simulation Engines

The construction of a simulation usually involves some sort of software development. Since software development is often a costly, time-consuming, and error-prone process, minimization of programming and re-use of validated code is highly desirable. A certain amount of code re-use can be achieved through the utilization of simulation engines. Many different simulation engines are available commercially. These engines, or simulators, are computer programs that typically provide functions to:

- Develop and manage simulation models
- Implement control logic and perform calculations
- Assist in model debugging
- Incorporate programming language extensions
- Input and output data
- Initiate and terminate simulation runs
- Generate statistical variations between runs
- Create and display 2D and/or 3D visualizations
- Analyze results and generate reports

By using a simulation engine, much low-level coding could be avoided. Unfortunately, most simulation engines currently run as stand-alone systems on personal computers.

The representation of time is a key aspect of simulation engines. They typically implement models as discrete-change, continuous-change, or combined-change, see [Banks 1998]. In discrete-change simulation engines, data variables only change values at distinct points in simulated time, i.e., discrete events. In engines that implement continuous models, data variables may change values continuously as functions of time. With combined models, variables may change discretely, continuously, or both. Some of the common elements found in simulation engines include a clock that keeps simulated time and system state variables that indicate the current state of the simulation execution. Other elements include representations for entities (the objects whose behavior is simulated), events (points in time where things happen), event queues (a sorted list of events), and random number generators (that are used to create variations in the simulation executions). Some of the most common techniques for implementing and extending models using simulation engines include the use of conventional procedural code, object-oriented programming, rule-based systems, and/or finite state machines.

In the past simulation analysts have expended considerable time and effort extracting data from other software applications to drive simulators. The next two sections describe the servers and the simulation data model that can be used to improve the management of this data.

4.1.3 Simulation Management and Data Servers

Management functions may be needed for the proper operation of a distributed simulation that are logically outside of any single simulation process. In the distributed manufacturing simulation environment, the Manufacturing Simulation Federation Manager is the architectural element that provides these functions. It may implement functionality to execute initialization scripts that launch federates, to provide initialization data to federates, to assist in federation time management, and to provide a user interface so that users can monitor and manipulate the federation and invoke federation services.

Simulation applications will typically need to access manufacturing data that was created using other software applications, e.g., a computer-aided design system. The other applications will store this information on data servers that will manage this information and deliver it to the simulations when requested. The data server typically acts as a central repository for storing data. As such, simulation applications are clients in a "client-server" architecture. On the server-side, at least six types of servers and data repositories may be used to store manufacturing and simulation data. See Figure 3 for an illustration of the simulation data servers. Each server type is briefly described below:

- *Relational Data Base Management System Server* – Perhaps the most common type of database system in existence today is the relational database management system or RDBMS. Some of the most popular commercial database systems are based upon relational technology. Information is organized into relations that are essentially tables. This type of server is potentially useful for storing any kind of information that can be represented in tables. A standard interface has been defined to access and update information in relational databases, the Structured Query Language (SQL), see [ISO 2003] for more information.
- *Object-oriented Data Base Management System Server* – Another, perhaps less commonly used, database server is based on object-oriented technology. With this database, all data elements are defined as objects that have attributes and methods that define the objects behaviors. Methods also define how the objects are created, modified, and destroyed. Each vendor typically provides a custom application programmer interface for interacting with the object-oriented database.
- *Web Data Server* – These servers enable users to access files or data over the Internet using HTTP protocol. Web server applications have been used to provide online access to documentation and models. For more on web data access mechanisms, see [W3C 1999].
- *Product Data Management System Server* – Product data management (PDM) systems are specialized database systems designed to support manufacturing organizations. PDM systems are designed to maintain product-life-cycle data including product specifications, part designs and models, process plans, bills of materials, and engineering change orders. PDM systems are typically implemented

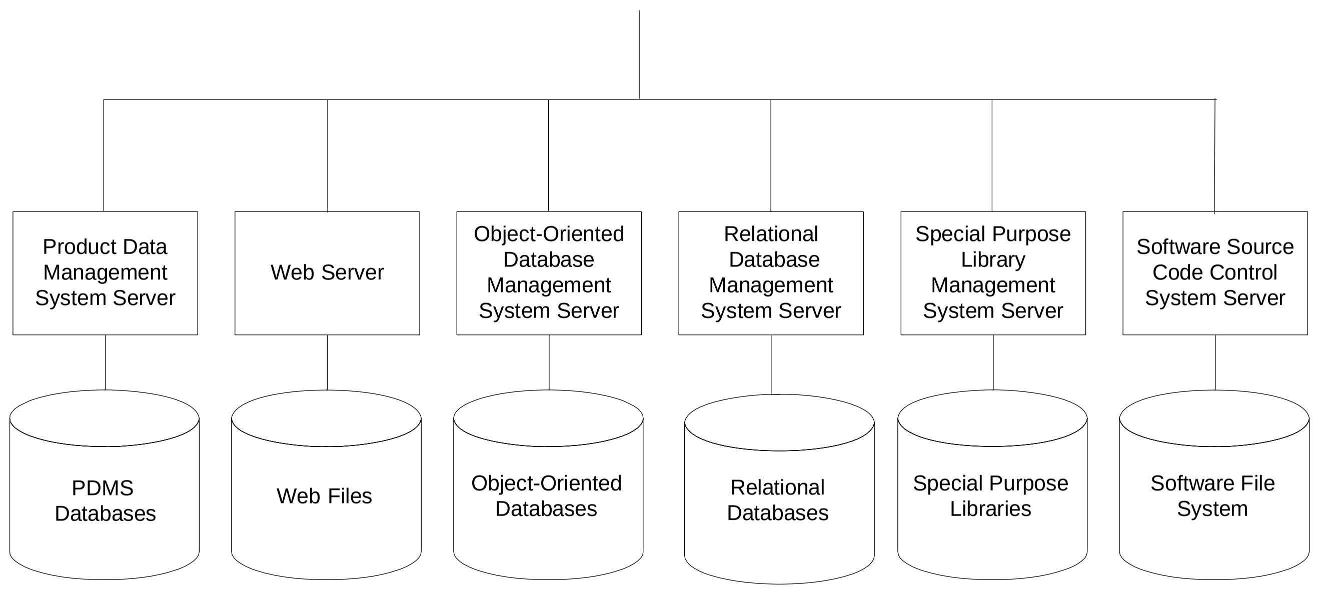
Product Data Management System Server
Web Server
Object-Oriented Database Management System Server
Relational Database Management System Server
Special Purpose Library Management System Server
Software Source Code Control System Server
PDMS Databases
Web Files
Object-Oriented Databases
Relational Databases
Special Purpose Libraries
Software File System

Figure 3. Types of simulation data servers.

using either relational or object-oriented database management systems. Standards have been defined to provide uniform mechanisms for accessing and updating product life cycle data, see [OMG 2002] and [STEP 2001].

- *Special Purpose Library Management System Server* – Often software vendors will incorporate their own custom database or library management system into their software. For example, simulation models may be stored and accessed using a vendor's proprietary interfaces. There is no standardization for these special purpose libraries. Unless the vendor provides an application programmer's interface, there will be no way to access the data on the server other than through the vendor's simulation package.
- *Source Code Control System Server* – Source Code Control Systems (SCCS) are special purpose data management systems that are designed to support the software development and maintenance process. SCCS servers store source code, object modules, header files, documentation, program build files, etc. One of the strengths of these systems is the ability to maintain version control data and build programs using the appropriate component modules. Access mechanisms are typically proprietary to a particular SCCS implementation.

The next section introduces a common, manufacturing-simulation data model that has been developed to enable sharing of information between a wide variety of manufacturing software applications.

4.1.4 Manufacturing Simulation Data Model

If a number of software applications including simulators are going to share data, they should have a common understanding of its meaning and structure. In this section, the concept of a common, shop information model is introduced. The primary objective of this model is to develop a structure for exchanging shop data between various manufacturing software applications, including simulation. The idea is to use the same data structures for managing actual production operations and simulating the machine shop. The rationale is that if one structure can serve both purposes, the need for translation and abstraction of the real data would be minimized when simulations are constructed. The mapping of real world data into simulation abstractions is not, for the most part, addressed in the current data model. Figure 4 illustrates some of the major elements of the conceptual data model and their relationships to each other. For a more detailed discussion of the data model, see [Lee 2003] or [McLean 2005a].

Maintaining data integrity and minimizing the duplication of data is an important requirement. For this reason, each unique piece of information appears in only one place in the model. Cross-reference links are used to avoid the creation of redundant copies of data.

The machine shop data model contains twenty major elements. Each of the major data elements are italicized in the discussion that follows. The data elements are called: *Organizations, Calendars, Resources, Skill-definitions, Setup-definitions, Operation-definitions, Maintenance-definitions, Layout, Parts, Bills-of-materials, Inventory, Procurements, Process-Plans, Work, Schedules, Revisions, Time-Sheets, Probability-distributions, References,* and *Units-of-measurement*. Due to space limitations, the entire

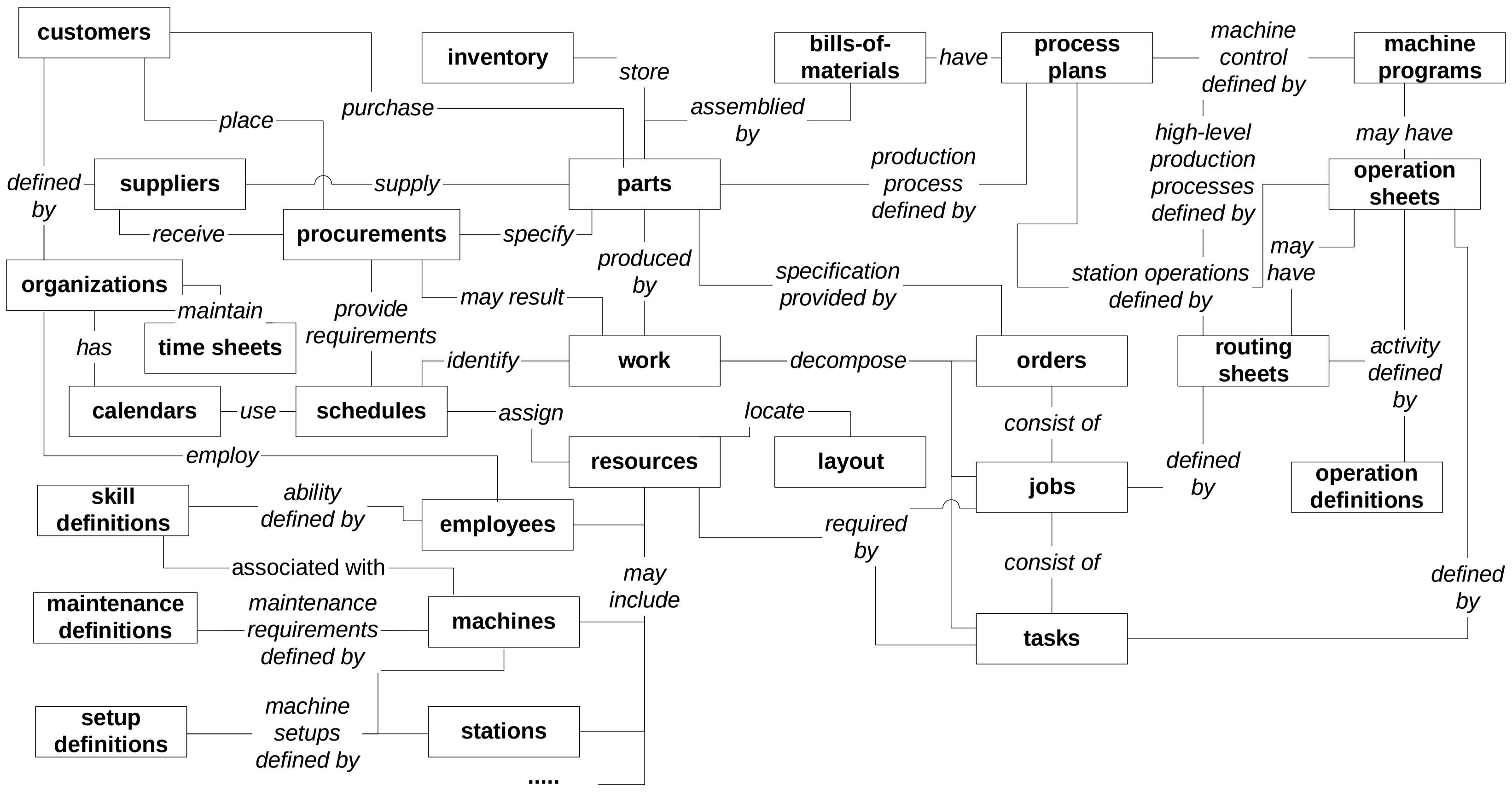

Figure 4. Major elements of the manufacturing conceptual data model and their relationships to each other

model is not shown or discussed in detail. The remainder of this section discusses the data elements and their significance.

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

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

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

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

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

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

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

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

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

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

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

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

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

Schedules and *Calendars* data elements are used to deal with time. *Schedules* provides two views of the planned assignment of work and resources. Work items (orders, jobs, and tasks) are mapped to resources, and conversely, resources are mapped to work items. The planned time events associated with those mappings are also identified, e.g., scheduled start times and end times. *Calendars* identifies scheduled work days for the shop, the shift schedules that are in effect for periods of time, planned breaks, and holiday periods.

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

This information model and associated data formats are undergoing standardization under the Simulation Interoperability Standards Organization [SISO 2005].

4.1.5 Simulation Integration Infrastructure

Although simulations are often implemented as individually executable computer processes, sometimes there is a need to divide simulations into multiple processes. These processes may need to run as a distributed simulation system on a single computer or over a network of computers. A distributed simulation system may be used to:

- Divide a large simulation into smaller functional modules that can be used by multiple training packages
- Provide a simulation service to other client applications
- Enable coordinated simulations over a local Intranet or the Internet.

The High Level Architecture (HLA) is a standard, originally initiated by the DoD, for implementing distributed simulation. It was developed by the Defense Modeling and Simulation Office (DMSO) to provide a consistent approach for integrating distributed, defense simulations. In HLA terms, the individual simulations are called federates and the distributed simulation is referred to as a federation. The HLA defines a framework by which individually executing federates can be combined into a distributed simulation federation.

The HLA framework has three major parts. The first part is a set of rules that federates and federations must adhere to ensure that a federation operates properly. The second part is the integration infrastructure called the Run Time Infrastructure (RTI). The RTI defines an interface that provides a number of services that federates can use to communicate (i.e., exchange simulation data), and coordinate their execution (i.e., synchronize simulation clocks) with other federates in a federation. The third part of the HLA is called the Object Model Template (OMT). The OMT provides a means for describing the format of the data that will be exchanged between federates. For more information on distributed simulation using HLA, see Kuhl et al (1999).

Several implementations of the HLA RTI software are currently available from different sources. There is, however, no interoperability across different vendor's RTI implementations. A distributed simulation running on different computer systems across a network must use the same RTI software as an integration infrastructure

An HLA-based distributed manufacturing simulation may include simulators, visualization system, real production system, and output analysis system as federates. Figure 5 illustrates the relationship between the various component elements of the distributed manufacturing simulation federate.

One common data definition is created for domain data that is shared across the entire federation. It is called the federation object model (FOM). Each federate has a simulation object model that defines the elements of the FOM that it implements.

Developing the adaptation code to integrate with the RTI can be a significant undertaking. This effort must be repeated for each legacy simulation that is to be integrated. Figure 5 also shows the architecture of a legacy simulation that has been integrated into a distributed simulation using the DMS Adapter. Instead of having legacy simulations integrated directly with the HLA/RTI, those simulations will interact with the interface of the adapter. The goal of the adapter is to provide a simplified method for integrating legacy simulations into distributed simulations while also providing as much of the capabilities of the HLA/RTI as possible. Some adaptation code must still be

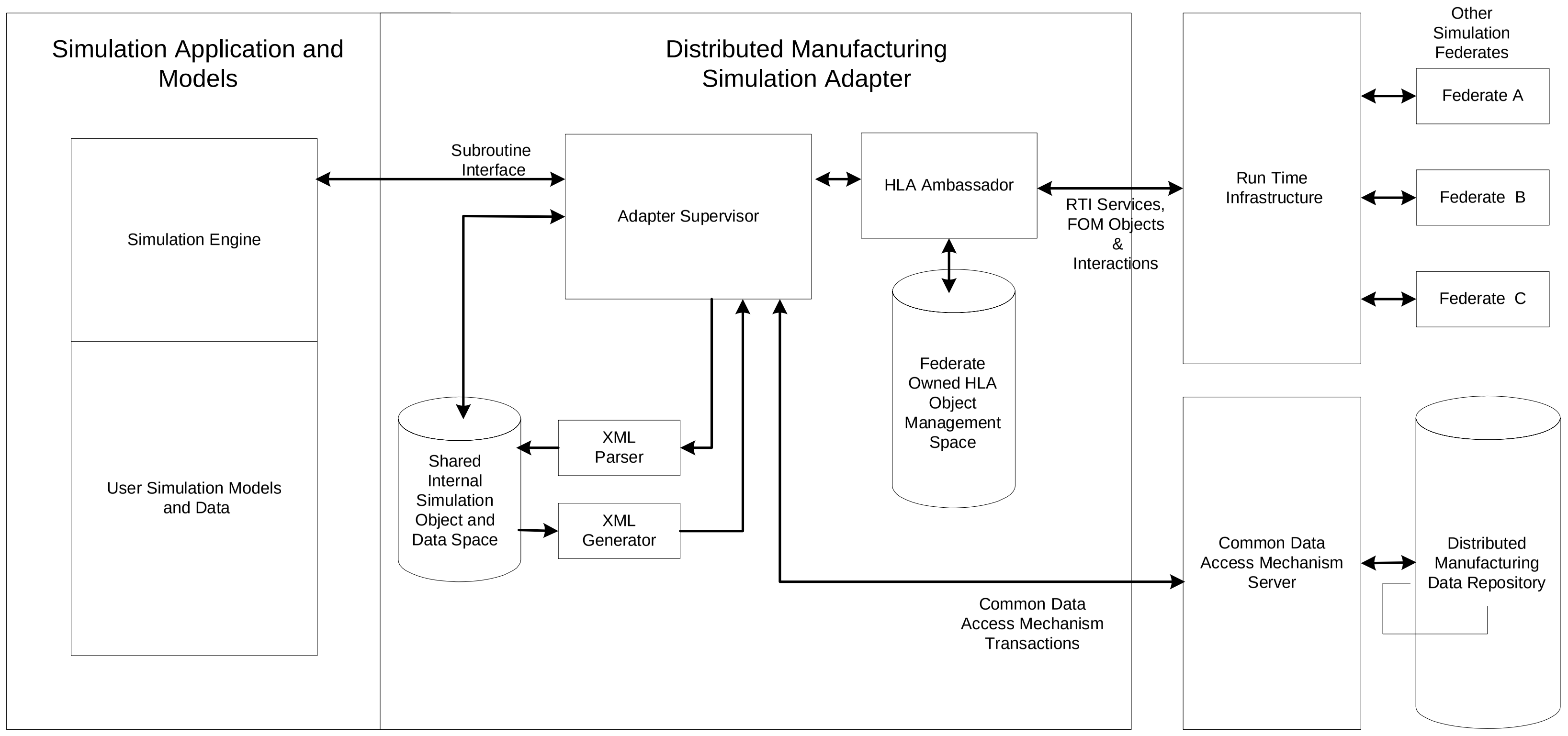
Simulation Application and Models
Simulation Engine
User Simulation Models and Data
Distributed Manufacturing Simulation Adapter
Subroutine Interface
Adapter Supervisor
HLA Ambassador
RTI Services, FOM Objects & Interactions
Federate Owned HLA Object Management Space
Shared Internal Simulation Object and Data Space
XML Parser
XML Generator
Common Data Access Mechanism Transactions
Run Time Infrastructure
Other Simulation Federates
Federate A
Federate B
Federate C
Common Data Access Mechanism Server
Distributed Manufacturing Data Repository

Figure 5. Structure of a distributed manufacturing simulation federate

developed to integrate a legacy simulation system with the DMS Adapter. However, by reducing the complexity of the interface to which the legacy simulation is being integrated, the level of effort for performing the integration should be greatly reduced.

DMS Adapter Module is incorporated into each DMS federate. The DMS Adapter handles the transmission, receipt, and internal updates to all FOM objects used by a federate. The DMS Adapter Module will contain a subroutine interface and data definition file that will facilitate its use as an integration mechanism by software developers. The DMS adapter eases the development of distributed manufacturing simulations by reusing implementations for some of the necessary housekeeping and administrative work. The DMS adapter provides a simplified time management interface, automatic storage for local object instances, management of lists of remote object instances of interest, management and logging for interactions of interest, and simplified object and interaction filtering. For a more detailed discussion of the NIST distributed manufacturing simulation architecture and the adapter module, see [McLean 2005B].

4.2 Manufacturing Gaming Subsystem

The purpose of the gaming subsystem is to provide an immersive virtual manufacturing environment in which users, i.e., players can interact with each other, with manufacturing systems, and organizations. The video game development community has become a leading innovator in the use of graphics, audio, and force feedback to create virtual worlds. Multiplayer game technology allows many players to interact in the same virtual world across the Internet. As such, it is only natural to look to the technology leaders for solutions to creating the front-end interface to a virtual manufacturing environment.

What functions does the gaming subsystem need to provide? Some of the key functions that need to be supported:

- Allow the creation of different game genres such as strategy, role-playing, and puzzle solving based on individual training needs
- Provide user interfaces that allow a variety of user input devices
- Animate characters and other objects
- Render graphics scenes, generate audio, and provide force feedback
- Sequence all processes in a timely fashion
- Implement intelligent behaviors for both player and non-player characters
- Coordinate multi-player game play across the Internet
- Enable “modding” of the game environment through high level scripts
- Compile high level scripts into more efficient low level code
- Manage user sessions and security
- Provide a central repository for game assets or resources

Figure 6 shows the major elements of the manufacturing gaming subsystem, i.e., clusters of game applications, game management, data servers, and communications infrastructure. The next sections discuss these elements in more detail.

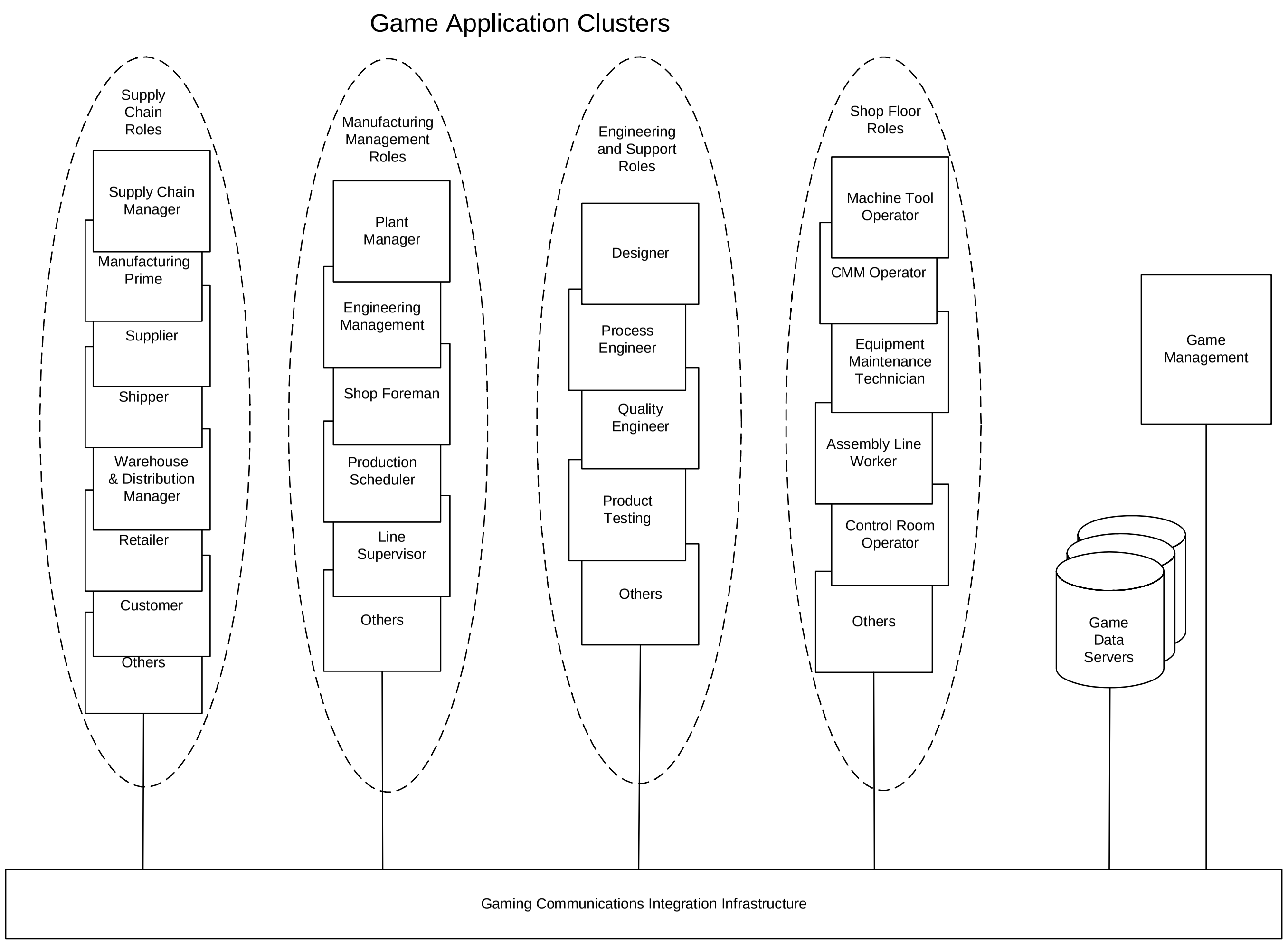

Figure 6. Manufacturing gaming subsystem: game application clusters, game management, data servers, and communications infrastructure

4.2.1 Game Applications

Manufacturing gaming applications are created to support the roles that users may need to play in manufacturing organization for research, testing or training purposes. Four groups of roles and game interfaces have been initially identified: supply chain representatives, manufacturing management, shop floor personnel, engineering and support staff. Other roles and interfaces might also be defined. Each group is briefly described below:

- *Supply Chain Representatives* – Allows players to assume various management roles in a manufacturing supply chain and interact with players representing other supply chain organizations. Players deal with supply chain organizational and management issues including selection of suppliers, negotiations between chain participants, determining lead times, management of inventory levels, etc. Some possible roles include supply chain manager, prime manufacturer, supplier, shipper, warehouse or distribution manager, retailers, and customers.

- *Manufacturing Management* – Players take on various roles in the internal management of a manufacturing organization. Some of the tasks include planning, scheduling, financial management, and reporting. Some of the possible roles include plant manager, engineering manager, shop foreman, production scheduler, and production line supervisor.

- *Engineering and Support Staff* – Players assume technical roles that allow them to perform various engineering and support functions that generate the technical data that is used to drive manufacturing operations. Possible roles include designer, process engineer, tool design, quality engineer, product testing, and others.

- *Shop Floor Personnel* – These roles and interfaces enable players to get hands-on experience with production operations and equipment using virtual machines and virtual products. Some possible roles include machine tool operator, coordinate measuring machine operator, equipment maintenance technician, assembly line worker, and control room operator.

- *Live elements* – The roles and interfaces enable the incorporation of live game play (outside of the virtual world), video feed, external communications channels, etc.

The above list of player roles and interfaces is only intended to be a sampling of what is possible. A game style interface could be devised for almost any job within a manufacturing organization. Management and engineering textbooks define many case study scenarios that could form the basis for engaging game interactions in each of these areas.

4.2.2 Core Game Engine

The Core Game Engine (CGE) provides the basic functionality required to play video games in stand-alone mode and in distributed mode using remote servers, and networks. The CGE executes the game content, outputs graphics to display devices, sound to speakers, and receives user input from different types of peripherals. The architecture of the CGE defines its major modules and the relationships of those modules to each other.

Figure 7 illustrates the major modules of the CGE and their general relationships to each other (Client Interface Module is not shown). The component modules of the CGE are briefly described below:

- *Game Content Module (GCM)* - consists of the scripts and objects of a particular game-based training application that are created by the content developer.
- *Supervisory Controller Module (SCM)* - provides an operating system that sequences processes and executes game content scripts and other internal subsystem functions.
- *Script Interpreter-Compiler Module (SICM)* - translates game content scripts into an internal computer program that invokes internal system services and functions of other modules within the game engine
- *User Input Module (UIM)* - processes and redirects user inputs from various peripheral devices
- *Data Management Module (DMM)* - provides support functions for maintaining and accessing shared data
- *Data Import/Export Module (DIEM)* - moves and translates data between external files and internal data stores
- *Artificial Intelligence* Module *(AIM)* - performs various decision-making or problem-solving processes that are normally associated with human or animal intelligence
- *Physics Module (PM)* - models behaviors of objects associated with various physical phenomena, e.g., collisions, gravity, buoyancy, aerodynamics
- *Animation Module (AM)* - manipulates various characteristics of objects over time to effect graphic, audio, and force renditions
- *Graphics (GM), Sound (SM), and Haptics Modules (HM)* - output information to displays, speakers, and force feedback devices
- *Client Interface Module (CIM)* – provides mechanisms for synchronizing timing and data interactions with other players via communications networks and remote data servers.

The development of the game engine architecture was developed in collaboration with the U.S. Naval Education and Training Command to support the Navy’s future simulation-based training needs. For a more detailed discussion of the game architecture, see [McLean 2004].

4.2.3 Game Integration Infrastructure

Unlike distributed simulation, there is no standard for integrating distributed multi-player games. In the simulation world, a number of implementations of the High Level Architecture (HLA) Run-Time Infrastructure (RTI) have been developed. In order to integrate a set of distributed simulations, one vendor’s RTI implementation must be selected and software must be adapted to work with that RTI.

In the game world, there is no software corresponding to the HLA RTI that can be used to integrate multi-player games created by one developer. For performance reasons, the

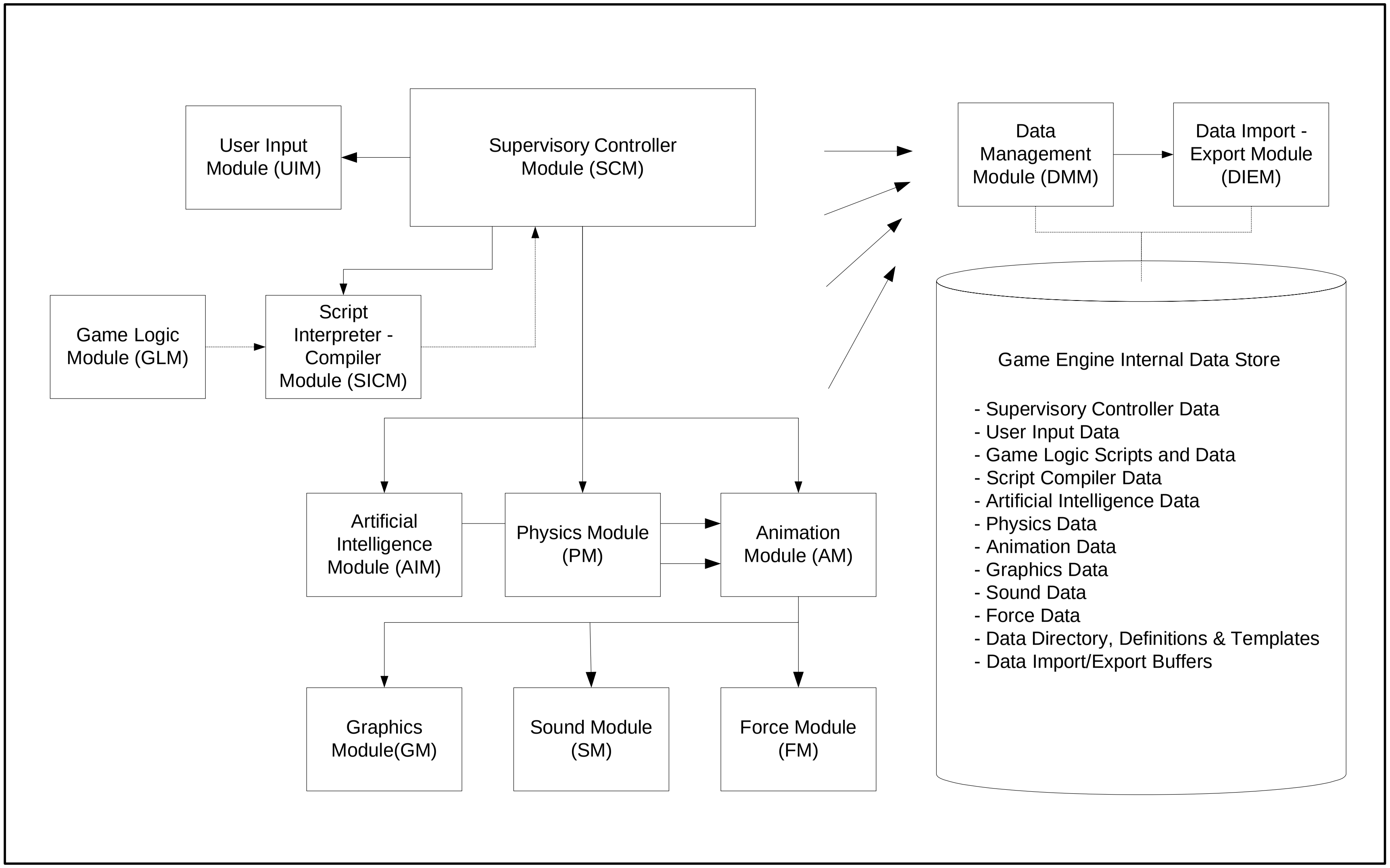
User Input Module (UIM)
Supervisory Controller Module (SCM)
Game Logic Module (GLM)
Script Interpreter - Compiler Module (SICM)
Data Management Module (DMM)
Data Import - Export Module (DIEM)
Game Engine Internal Data Store
- Supervisory Controller Data
- User Input Data
- Game Logic Scripts and Data
- Script Compiler Data
- Artificial Intelligence Data
- Physics Data
- Animation Data
- Graphics Data
- Sound Data
- Force Data
- Data Directory, Definitions & Templates
- Data Import/Export Buffers
Artificial Intelligence Module (AIM)
Physics Module (PM)
Animation Module (AM)
Graphics Module(GM)
Sound Module (SM)
Force Module (FM)

Figure 7. Major modules of the core game engine and their general relationships to each other.

HLA RTIs have been considered unsuitable by game developers. Perhaps the primary reason that the game world has been less than enthusiastic about HLA is their need for real-time performance. HLA can help guarantee that simulations behave in a technically correct manner, e.g., messages arrive at recipient in proper order, but in doing so HLA implementations may sacrifice efficiency and performance to achieve technically correct behavior. Also in the HLA world, there is no requirement for centralized control or persistent data storage. Multi-player games typically need to centralize control and maintain game state data for long periods of time.

Although a common gaming integration infrastructure does not exist, there are some features that are typically used to integrate multi-player games. Features that will be briefly discussed include the client-server interaction model, proxies, arbitration, and socket-based communications.

- *Client-server interaction model* – The primary mechanism used for integrating multi-player games today is the client-server model. Peer-to-peer games have been implemented in the past, but it is unlikely that they will be used in the future. Players do not communicate with each other, but rather with the server. The client-server model is capable of supporting a large number of players, where the peer-to-peer model does not. In client-server game implementations, the server acts as a centralized control point for the game. Game action takes place on the server, but is reflected in the player's display on the local platform. The server is responsible for determining the advancement of time.

- *Proxies and arbitration* – The characters and action that the player views on the local platform is just a proxy for the real characters and actions that exist on the server. When the player moves a proxy character on the local platform, the server must verify that the move is acceptable. The server can later adjust the movement and location of the character, if it determines that the move is not right. An action taken by a player on the local platform must be confirmed by the action on the server. The server as an arbitrator of game interaction and is the ultimate authority on determining game state. For example, if two players are involved in a battle, the server ultimately determines who wins and who loses.

- *Socket-based communications* – The most common mechanism used for communications between clients and servers in multi-player games are sockets. A socket is an endpoint of a two-way communication link between two programs running on a communications network. A socket is bound to a particular port number on a networked computer. The client and server programs write message packets to the socket for delivery to each other. Sockets guarantee the delivery of game data packets across the Internet.

It appears unlikely that the integration infrastructure requirements of the gaming and simulation worlds will be reconciled any time soon. For the purpose of the proposed reference architecture, a separate local integration infrastructure has been identified for each world. A bridge has been defined to join the two worlds, i.e., simulation and gaming. The bridge does not guarantee that the two worlds can be successfully integrated in all circumstances. Time synchronization between the two worlds is perhaps the major problem. If the simulation world can keep up with real-time or generate and store data in advance of game play, integration may be achievable. If a game is turn-based, its performance requirements are not severe, or its execution can be slowed for synchronization purposes, integration may also be achievable. Otherwise, a mutually acceptable integration solution must be found between the two worlds.

4.2.4 Game Management and Data Servers

Because of the extensive use of the client-server-computing model in multi-player gaming, data servers perhaps play a more important role in gaming than they do in simulation. Servers are needed to establish connections for players joining a game session, manage and distribute content, and maintain game state and arbitrate interactions between game players.

These servers may be implemented using the relational, object-oriented, or web-based technologies described in the earlier discussion of simulation data servers. In addition to the purely game-based servers, two other types of servers may be used to support educational games: learning content management system (LCMS), and learning management systems (LMS) servers. Each of the four major types of gaming and educational servers is described below:

- *Game Connection Server* – Large multi-player games may require multiple game world servers to run concurrently in order to support the large number of players involved. Rather than connect directly to the game world, the player first contacts a connection server that acts as operator. The connection server validates the player's identity and routes the player to an appropriate game world server. As such, the connection server may perform load balancing to ensure an appropriate number of players are being supported by each game world server.

- *Game World Server* – The game world server is typically a powerful computer that hosts the game for many players. It is responsible for maintaining game state data for all players. It maintains persistent data between game play sessions. It arbitrates interactions between players. It provides security mechanisms to ensure that players cannot hack or unfairly manipulate game play. Game world servers may divide the regions of game play into cells. As players move around the world, they may move into different cells and be routed to new servers. As programmers update game worlds over time, the preferred approach is to patch the code on the servers rather than on client computers. This simplifies software updates since code does not need to be distributed to clients, rather only game world data is transferred.

- *Learning Content Management System Server* – The LCMS is a development environment where multiple learning content authors can create, store, reuse, manage, and deliver digital learning content. The LCMS provides a central repository for the storage and retrieval of learning content objects. The LCMS may include Learning Management System (LMS) and Course Authoring System (CAS) functions. The

LCMS may also interact with external LMS and CAS systems developed by other software vendors.

- *Learning Management System Server* - The LMS server is a system for managing learners, keeping track of their progress and performance across all types of training activities. Other functions of the LMS may include publishing a catalog of course offerings, providing communications connectivity and interact with the LCMS to obtain course content, providing access control to courses including mechanisms for course enrollment and checking of prerequisites, managing personalize learning plans, launching and tracking progress on learning applications. It also provides instructor interfaces for grading, retaking courses, setting up courses, maintaining transcripts for the student population, taking required tests and assessments online, track student progress, scores, completion, etc. The LMS may also enable collaboration and communication between students and instructors, and enable virtual classrooms. It also may provide an Application Programmer's Interface to enable interactions between courseware and the LMS.

4.2.5 Game Information Model

A common information model will help facilitate integration of gaming systems with each other and with manufacturing simulators. Currently no such model exists. If a common model did exist, developers working on different aspects of a game development could reasonably expect to be able to share information with modules created by other developers. NIST staff have begun to identify the major data types that would be required in a common gaming information model, as well as the data required to support game-based training in two other application areas, i.e., homeland security and Navy training applications. At the highest level, the major types of data required for gaming within the core game engine and/or on the server include:

- *Supervisory controller data* – information about internal game processes and scripts, priority, status, etc.
- *User input data* – information about the configuration of user input devices (e.g., game controller, keyboard, mouse, camera), input data stream, status etc.
- *Game scripts and data* – structures to store program and data information such as game levels, objects, game world and character state in languages used for game scripting, including Python, Lua, Ruby, Perl, and C++
- *Script compiler data* – data structures used in the translation of high level scripts into low level executable code
- *Artificial intelligence data* – data structures used to model intelligent behavior and decision making processes based on various techniques such as finite state machines, genetic algorithms, fuzzy logic, and general problem solvers
- *Physics data* – various physical parameters of game objects and the environment including geometry, mass, location, and velocity
- *Animation data* – animation plan, key frames definitions, interpolation parameters, motion and morphing algorithms and data
- *Graphics data* – scene structures, object geometry, textures, lighting, special effects such as smoke and fog

- *Sound data* – audio clips, playing parameters, audio source location, environmental parameters, special effects
- *Force data* – force feedback, vibration, etc.
- *Data import/export buffers* – directories and files, file types, translators, mapping of file data to internal data structures

The above list of data types is not exhaustive, but is indicative of the various types of data that must be incorporated into a common information model that supports gaming.

5. Conclusions

This paper presented an architecture for integrating gaming and simulation in the manufacturing domain. This integration will bring together the interactive environment provided by gaming for taking actions and making decisions for a situation with the capability to use simulation to produce technically correct impact of the actions taken and decisions made. The integration hence allows the use of gaming for serious applications for manufacturing industry. Such serious gaming applications are expected to be quite effective for the coming generations of workforce that have grown up playing video games.

The integration of gaming and simulation will also allow training of personnel and testing of applications that simultaneously address different levels of manufacturing hierarchy. Such simultaneous involvement of hierarchically diverse personnel and applications will provide improved opportunities for team training. Manufacturing management personnel can visualize how personnel further down the chain implement their decisions. For example, management personnel may make some aggressive scheduling decisions that then lead to the operators using gaming applications to carry out the production on virtual machines. Such an environment will allow testing of practices or technologies for communicating information across the hierarchy as well as the decisions at each level.

It is proposed that the architecture be implemented as a common infrastructure that can be used to integrate independently developed simulation and gaming modules. The availability of such an infrastructure will strongly encourage development of gaming and simulation modules covering the breadth and depth of the manufacturing domain. Manufacturing personnel can select the modules applicable to their environment to create a capability to serve their research, testing and training needs.

An implementation of the architecture will provide a test bed for the Manufacturing Interoperability Program at NIST's Manufacturing Engineering Laboratory and other standards organizations. It can be used to test the interoperability of manufacturing applications including enterprise resource planning, scheduling, manufacturing execution systems, machine and material handling equipment control programs, and machine and robot programs. It can also be used to test the interfaces for such applications.

The proposed test bed will be highly effective if supported with repositories for templates and test case data. Academic and commercial researchers can use the templates and test case data to quickly test out new developments. The test case data can also serve as a benchmark for comparison of alternate approaches for similar applications and thus further spur development and help manufacturing personnel by providing a common scale to rank vendor offerings.

Implementation of the architecture as a common infrastructure will require development of standards at several fronts including the data models, interfaces, distribution and synchronization mechanisms and user interaction devices. NIST researchers have prepared draft standards for shop floor data and are working with the Simulation Interoperability Standards Organization for their formal acceptance. Current work in progress on integration of gaming and simulation is expected to lead to more such activity in the future.

6. References

[Banks 1998]	Banks, J. (Ed.), Handbook of Simulation: principles, methodology, advances, applications, and practice, John Wiley and Sons, New York, NY, 1998.
[Demaria 2004]	DeMaria, Russel and Johnny L. Wilson, High Score! The Illustrated History of Electronic Games, McGraw-Hill/Osborne, Emeryville, CA, 2004.
[IEEE 2000]	IEEE Standards Association, 1516-2000 IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules. (Accessed at http://standards.ieee.org/)
[ISO 2003]	International Organization for Standardization, Information technology -- Database languages -- SQL -- Part 1: Framework (SQL/Framework), ISO/IEC 9075-1:2003, (accessed at http://www.iso.org/ on 08-18-05)
[Jain 2001]	Jain,S., N.F. Choong, K.M. Aye and M. Luo, "Virtual Factory: An Integrated Approach to Manufacturing Systems Modeling", International Journal of Operations and Production Management, Volume 21, no. 5/6, 2001, pp. 594-608.
[Kuhl 1999]	Kuhl, F., R. Weatherly and J. Dahmann, Creating Computer Simulations: An Introduction to the High Level Architecture, Prentice Hall, Upper Saddle River, NJ, 1999.
[Lee 2003]	Lee, Y. Tina, Charles McLean and Guodong Shao, “A Neutral Information Model for Simulating Machine Shop Operation.” Proceedings of the 2003 Winter Simulation Conference, eds: S. Chick, P.J. Sanchez, D. Ferrin and D.J. Morrice, Institute of Electrical and Electronics Engineers, Piscataway, NJ, 2003, pp.1296-1304.

[LSC 2004]	Learning Strategies Consortium Conference, http://www.lscconsortium.com, July 26-27, 2004.
[McLean 2004]	McLean, Charles and Frank Riddick, "PC Modeling and Simulation Guidelines: Volume 1 – Overview," NISTIR 7131, National Institute of Standards and Technology, Gaithersburg, MD, August 2004.
[McLean 2005a]	McLean, Charles, Y. Tina Lee, Guodong Shao, Frank Riddick, "Shop Data Model Interface Specification," NISTIR 7198, National Institute of Standards and Technology, Gaithersburg, MD, January 2005.
[McLean 2005b]	McLean, Charles, Frank Riddick, Y. Tina Lee, "An Architecture and Interfaces for Distributed Manufacturing Simulation," Simulation: Transactions of the Society for Modeling and Simulation International, Volume 81, No. 1, Sage Publications, San Diego, CA, January 2005, pp. 15-32.
[MIT 2005]	http://www.educationarcade.org/
[OMG 2002]	Object Management Group, Inc., Distributed Simulation Systems Specification, version 2.0 (accessed at http://www.omg.org/technology/documents/formal/distributed.htm on 8-18-05)
[Rollings 2004]	Rollings, Andrew and Morris, Dave, Game Architecture and Design: A New Edition, New Riders Publishing, Indianapolis, IN, 2004, p. 409.
[Serious 2005]	http://www.seriousgames.org/
[SISO 2005]	Simulation Interoperability Standards Organization, Core Manufacturing Simulation Data Product Development Group, (accessed at http://www.sisostds.org/ on 8-18-05)
[STEP 2001]	PDM Implementor Forum, Usage Guide for the STEP PDM Schema, Release 4.3 (accessed http://www.pdm-if.org/pdm_schema/pdmug_release4_3.zip on 8-18-05)
[Vargas 2004]	Vargas, Jose, "Problems You Can Shake a Joystick At: War Room to Sickroom, Video Games Are Red-Hot," Washington Post, Washington, DC, October 18, 2004, page A01 and http://www.washingtonpost.com/wp-dyn/articles/A40639-2004Oct17.html?sub=AR.

[W3C 1999]	Network Working Group, Hypertext Transfer Protocol -- HTTP/1.1, (accessed at http://www.w3.org/Protocols/rfc2616/rfc2616.html on 08-18-05)

7. Acknowledgement

Work described in this report was sponsored by the National Institute of Standards and Technology (NIST) Manufacturing Interoperability Program, Gaithersburg, Maryland and in part by the Naval Education Training Command, Pensacola, Florida. The Manufacturing Interoperability Program supports NIST projects applying information technologies and standards-based approaches to manufacturing software integration problems. The work described was funded by the United States Government and is not subject to copyright.

8. Disclaimer

Software architecture, models and languages are identified in context in this paper. This does not imply a recommendation or endorsement of the associated commercial software products by the authors or NIST, nor does it imply that such software products are necessarily the best available for the purpose.