

The system integration architecture: A framework for extended enterprise information systems

J. J. Mills and M. Brand

The Agile Aerospace Manufacturing Research Center,

The Automation & Robotics Research Institute,

The University of Texas at Arlington

Fort Worth, TX, USA, 817/272-5927 FAX: 817/272-5952

jmills@arri.uta.edu mbrand@arri.uta.edu

Abstract

Manufacturing and enterprise formation is changing dramatically with speed and response time key parameters in competitive success. Companies are forming extended or virtual enterprises to tap core competencies and speed up their response times. Unfortunately, current information systems are a major hindrance to the rapid formation of such organizations and often to the rapid response of a single company wanting to change the way it does business. The *Systems Integration Architecture (SIA)* project is a research project to identify and resolve issues in the rapid integration of dynamic heterogeneous hardware, software and typical in this agile world. SIA is an integration framework based on a new basis model and definition of integration which are briefly described. The background to SIA, some of the issues involved and the approach taken to address them within SIA are presented. Specifically, some of the critical issues in the design of an information infrastructures such as SIA are discussed

Keywords

Systems Integration Architecture, integration, infrastructure, manufacturing, information systems, extended enterprises, virtual enterprises, heterogeneous systems integration

1 INTRODUCTION AND BACKGROUND

Manufacturing is undergoing dramatic and radical changes. Quality has advanced to the point where it is no longer a major competitive advantage and time to market is a key driver. Outsourcing has become a way of life - suppliers are being asked to participate in the design phase of a project instead of simply being handed a set of prints and asked for a low cost bid. Strategic partnerships, with integrated suppliers, qualified on performance, quality and delivery, are now recognized as critical for bringing a low cost, high quality product to market faster than anyone else. Agility, the term coined to describe these changes and paradigm shifts has entered the vocabulary of all manufacturers.

Agility has been defined as the ability to rapidly respond to unanticipated changes in market and customer demands and to thrive and prosper in such an environment (Goldman). In pursuit of "agility", organizations are reconfiguring their business processes, incorporating "core functions" from suppliers and customers to form "virtual or extended enterprises" in a very short period of time. One goal of these activities is to achieve extremely short response times for product realization and order fulfillment. The environment which these activities create and in which modern information systems must operate includes rapid change and reconfiguration, heterogeneous computer hardware, data bases and applications, and different business rules and control paradigms. Traditional, monolithic systems incorporating a central, shared data base are not designed to accommodate such requirements and new approaches are needed. Indeed, current information systems form a significant barrier to the formation of an extended or virtual enterprise. Integration must take on a new meaning and meet new requirements in this agile, virtual world. Further, agile information systems must *facilitate* the reconfiguration and the formation of a new business in a very short time, not the 1-2 years required for a conventional approach.

The technology of "integration" itself is complex and vast in scope, from the integration of shop floor motion control devices (Senehi et al., 1991) to the integration of whole enterprises including computers, machines, management organization and human resources (CIM-OSA Reference Architecture Specification, 1990). Integration is a much abused and much misunderstood word that has many meanings and as many implications. For the purposes of this paper, we define the scope of "integration" to encompass the unification of: (a) heterogeneous, distributed computers, (b) heterogeneous data and information, (c) functions provided by software applications (d) diverse control paradigms and business rules, and (e) diverse communications systems. We do not consider the broader issues of how the computers integrate with humans, organizations and other resources. However, under heterogeneous computers we do include any machine that is controlled by a computer.

The Systems Integration Architecture project at the Automation & Robotics Research Institute is focused on deepening our understanding of integration, including reference frameworks, architectures, and information infrastructures for these agile environments, and to explore integration issues within such an environment.

The purpose of this paper is to present our current thinking on the major issues in integrating heterogeneous information systems, discussing various issues and our approaches to addressing them. Specifically, a new basis model for integration is suggested which leads to a new, more appropriate definition of integration. We describe the services required to support this new definition and model of integration and show how the System Integration Architecture (SIA), which is being developed by the Agile Aerospace Manufacturing Research Center, can provide these services. We recommend a standard set of attributes of data interfaces required for plug compatibility and present progress on the design and implementation of SIA.

2 BASIS FOR THE INTEGRATION ARCHITECTURE DESIGN

In previous work, we have identified several issues and presented an approach which addresses them (Mills I, 1995). We describe them briefly below to provide some background

for the discussion of our information infrastructure. The first and major issue concerns the definition of integration in the environment (scoped out above). Elsewhere, we have examined previous definitions, models and identified problems with them (Mills II, 1995). Several features appear common: the concept of a central data base or repository, the need for data integration to include the ability to manage relationships among data sets (Jarke, 1992, Wasserman, 1990), the idea of "Levels of Integration " (CIM-OSA Reference Architecture Specification, 1990), and the need for integration of "processes" or "functions" (Jarke, 1992, Wasserman, 1990). and control (Nof and Papstravrou, 1992). None of these features address the problems of trying to integrate dynamic, heterogeneous systems with legacy data and applications.

2.1 Traditional Basis model

A conclusion we reached during this research was that the traditional basis model (Mills II, 1995) of the logically unified, but physically distributed yet shared data base in a neutral format is not adequate in an agile, virtual world. It has several flaws, among them: Lack of time dependence, unification, no concern for the relationships among data sets and the inability to consider task, function or activity integration.

This traditional basis model has been and still is very useful, and provided a very powerful concept for developing integrating systems. The importance of a common basis model can be illustrated by observing what is happening in the object oriented domain with the Object Management Group using one object model, Microsoft using another, and IBM using yet another (Betz, 1994). This means that the integration of two systems, based on these two different object models, is very difficult. A new, more appropriate basis model is needed for integration in the dynamic agile world of manufacturing.

2.2 Proposed Basis Model of Integration

From the basis model perspective, therefore, the main issue outlined above might be better stated; "What is the new basis model for integration in an environment of heterogeneous, physically distributed hardware, operating systems, software and data?"

Elsewhere we have proposed a new basis model based on research in the design theory and methodology domain (Finger and Dixon, 1989, Nguyen and Rieu, 1987, Takeda et al., 1990, Suh, 1990, Ulrich and Seering, 1987, Sthanusubramonian et al., 1992, Talukdar and Fenves, 1989). Briefly stated, this model, called the TAR model (Mills II, 1995), suggests that all information processing simply "transforms" data sets (Figure 1).

The model is intrinsically modular consisting of software and an input and output data set. The output data set of one becomes the input data set of another (see Section 2.5). Some

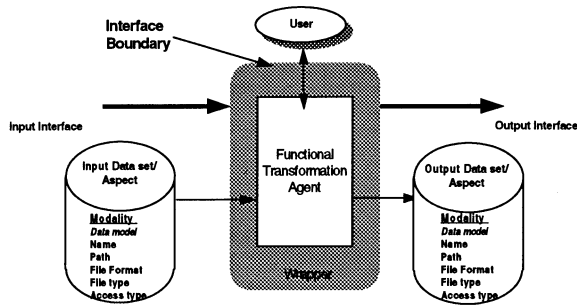


Figure 1. Proposed Basis Model for Integration

transformations may require more than one data set as input. Others may output more than one data set. Transformation modules can also share data sets. The entity which performs the transformations can be as large as a whole business information system (which transforms orders into shipped products and invoices) or modules which transform two numbers into their sum. Modules can be assembled into processes which are simply larger transformation modules (Figure 2). For convenience we call the data sets “Aspects” since they are an aspect of the product model.

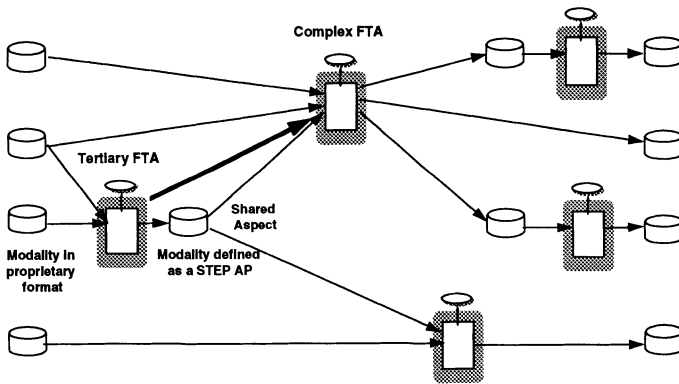


Figure 2. Linking of atomic FTA's into Complex FTA's

Transformations can be manual (i.e. by humans using pen and paper), partially automated (i.e. by humans using some form of computer application to capture the results of his/her transformation), or fully automated (the computer application transforms the Aspect on its own with no human intervention). For brevity, we define the entities which effect transformations as "Functional Transformation Agents" or FTA's. Three kinds of FTA's are recognized: Primary, Secondary and Tertiary. Primary FTA's transform Aspects along the primary product realization process. In the 3-D space formalism of Taylor and Henderson, Primary Transformations transform information sets along the life cycle application axis (Taylor and Henderson, 1994).

Secondary FTA's move any particular Aspect off this main path into a secondary Aspect necessary to perform some test, verification, evaluation or analysis or to assist a human in some activity necessary to understand a particular product representation. They transform the Aspect into another form within a product realization phase. In Taylor and Henderson formalism, they transform information sets with the Generalization-Specialization levels of detail planes (Taylor and Henderson, 1994). Tertiary FTA's translate format, see section 2.5.

2.3 New definition of Integration

Integration within this basis model then is the act of defining FTA's with matching Aspects, and providing mechanisms to support (a) their composition into higher level FTA's or processes, and (b) their enactment, monitoring and control. If this is done in a permanent manner using a shared data base then we have a traditional information systems solution such as has been implemented in many companies. If we provide an infrastructure in which FTA's with matching Aspects and modalities can be configured and reconfigured rapidly, then agile manufacturing can be accommodated. Note that this definition is very close to that suggested by Seheni et al. (1991).

2.4 Module Definition

If this basis model and definition are accepted, a concomitant issue is: What should the modules (i.e. FTA's) be - applications or functions? The traditional, vendor supported view is that since applications are provided by the vendors, the modules should be applications. However, tools or applications are complex combinations of functions which generally (in the case of a suite of applications from a single vendor) cannot be easily separated and can only be readily composed into higher level functions or processes when all the applications to be composed are from a single vendor. From a systems engineering viewpoint, it makes more sense to be able to compose well defined functions with clearly defined interfaces into processes. Examples of "function": might be "create model," "edit model," "assemble product," "plan production," etc. Hence, it is our position that the modules (FTA's) should be based on functions not applications.

2.5 Module Interfaces

The next issue of importance is: "What should the interfaces of the modules look like?" In other words, how should we structure the Aspects so that an Aspect created as the output of one FTA can be the input of another.

In theory, all "data", including commands to start, stop, pause and resume applications can be regarded as data and transformed. Consequently, one could envisage an interface definition based only on data sets. However, in some domains, a function may require another function (or application) to launch and control a second function. This is important in the domains of shop floor control and virtual manufacturing and engineering. As Seheni et al (1991) have pointed out in their work on manufacturing systems integration, commands need to be separated from data, and they, in fact, identify five different components to an interface. To simplify matters, only two major interface components are recognized: (a) administrative

commands such as start stop, etc., and (b) Aspects, the data sets in the TAR basis model. Figure 2 illustrates the concept of one FTA commanding another by means of the thicker arrow.

The purpose of an Aspect is to be able to match up the output Aspect(s) from one or more FTA's to the input Aspect(s) of other FTA's so that they can be formed into processes. In the System Integration Architecture project at the Automation & Robotics Research Institute, research has indicated at least six attributes are required to define the structure of the Aspect so that this matching can be achieved reliably: the data model or type of data, the data format (e.g. DXF, IGES, STEP for CAD, RTF, ASCII for text, TIFF, HGL, PCX for graphics, etc.) the existence, the location, the physical storage structure (e.g. file or data base) and the access type (e.g. read/write, data base query, STEP SDAI, RPC, API or http call) of the actual data set. These attributes are not unrelated. Clearly if the data are accessed through an API, then the physical storage attribute is not relevant. This structure incorporates the idea of the data model and the Standard Data Access Interchange (SDAI) of STEP, but broadens the concepts. It is proposed that this structure be called the Modality of the Aspect since different values of the attributes describe different modes of the Aspect.

A two step method of use for a Modality has been identified. During process composition (i.e. build time) only the data model attribute need be matched to form a process. At run time, however, the data format, existence, location, physical storage structure and access type become important for an executable process. Figure 2 illustrates the concept of linking FTA's through Aspects, into processes. It also illustrates the concept of Tertiary FTA's which simply transform the modality of an Aspect. These are typically the translators provided by CAD vendors.

3 DESIGN ISSUES IN THE SYSTEMS INTEGRATION ARCHITECTURE

The next issue is: "How do we support rapid reconfiguration of FTA's?" That is: What kind of infrastructure is required to support the creation, enactment and control of FTA's? What services should this product provide for legacy systems, heterogeneous computers, functions and data? The System Integration Architecture (SIA) will provide the services necessary to address this and other, related issues. SIA is based on an object oriented, client server paradigm. We are currently using Iona Technologies' Orbix implementation of the Common Object Request Broker Architecture (CORBA) to facilitate distributed objects for our client-server environment.

3.1 Access to Remote Functionality

The infrastructure must communicate with an FTA whose application resides on one remote computer while its Aspect(s) are located on another computer. The infrastructure enacts, monitors, and controls the FTA execution process. The term "execution process" indicates that an application has been enacted and remains within a computer's execution space. FTA's must also have access to Aspects which reside on other computers. Finally the

system must allow FTA's to be enacted in a predefined order. These requirements for the infrastructure are discussed below with some of the design and implementation details of SIA. While this paper is focused on manufacturing, the issues addressed in this project are much more general and can be applied to a wide variety of other domains.

The Launching Server: The infrastructure must communicate with each node containing an Aspect or an application which is a part of an FTA's definition. A server is installed on each computer within the SIA network to provide launching capabilities. The launching server receives incoming communications and provides local functionality for controlling FTA executing processes. The launching server must enact new execution processes, provide monitoring functionality, and allows an end user or another FTA to control the execution process by sending commands to kill, pause, and resume the process.

When the launching server receives a message to enact a new FTA it provides services for notifying the end user if the execution process sends standard output or error messages or if the process terminates normally or otherwise.

FTA's must be spawned as a new execution process so that the launching server's can respond to new incoming messages. Multiple execution processes can therefore be monitored by the same launching server. We are investigating possible solutions to this problem. One possibility is for the dispatcher to be enacted as a forked (multi-threaded) process. A second execution process, or thread, is created which in turn enacts the FTA's execution process. The forked thread can monitor the output streams and will terminate when the FTA process terminates. The forked thread must communicate with the infrastructure to provide notification of errors or termination.

We have encountered a significant degree of complexity while implementing the launching server. Maintaining communications between the parent and children forked processes has proven difficult using the normal Orbix implementation. The multi-threaded version of Orbix would add a considerable degree of complexity and cost to our implementation. Because launching servers represent the most important limit on the scalability of SIA we are investigating other possibilities.

Distributed Communications: The infrastructure must provide communications between the network computers. The communications module must provide distributed communications between the various launching servers. Several methods of distributed communications are available as third-party products. We use third party products wherever possible within SIA to maximize re-use and efficiency. The Communications module must provide distributed communications protocols which are efficient, integrate easily, accessibility, and cost.

We initially implemented distributed communications for SIA using TCP/IP and UDP communications protocols. We converted to Orbix as a result of the emergence of Object Management Group's CORBA as a standard for distributed object communications ("The Common Object Request Broker Architecture and Specification", 1993). CORBA provides distributed communications between objects regardless of which execution space or computer they reside. However, CORBA must be installed on each machine that provides a launching

server or end user interaction. Installation of the vendor product requires both a time and economic investment for each computer and so reduces scalability.

RPC and HTTP are communications protocols which are commonly installed on networked computers. RPC is readily available and has existed for long enough to be considered a stable protocol. RPC is not, however, as directly applicable to the object paradigm. HTTP also appears to have much promise. The popularity of the World Wide Web, which uses the HTTP protocol, suggests that in the near future most, if not all computers, will have an HTTP installed. The World Wide Web's dependence on HTTP servers ensure that they will be freely available and easily accessible for all platforms. HTTP also provides the ability to transfer data and execute processes on remote network nodes. HTTP was not, however, designed for inter-process distributed communications and so is not as robust as CORBA or RPC.

It is not clear which, if any, protocol will emerge as the dominant standard. Accordingly, we are implementing SIA so that distributed communications protocols themselves are rapidly reconfigurable. We can then compare the different methods of distributed communications and choose the most applicable. The design will also allow for *dynamic* reconfigurability of the communications protocol. Each node may have some different form of communications previously installed. Dynamically reconfigurable communications protocols allow new computers to be integrated rapidly using their native communications protocols.

Distributed Aspects(Data Sets): FTA's generally require input and output Aspects. Aspects are composed of data stored somewhere on the network. The FTA's application must have access to an input Aspect and must provide output Aspects to the infrastructure. The infrastructure must, therefore, transfer or otherwise make available data which exists on one network computer to an application which executes on another.

The infrastructure must provide methods of transferring data securely through the network on demand. The infrastructure must also know the Modality (including the location) of the input Aspects before an FTA can be enacted. It may be necessary to extract data from within a database for use as input to the FTA. A file may be transferred from one computer to another, but the copied file is considered to be transitory and will be destroyed upon completion of the FTA's execution process. Output Aspects must be dealt with in a similar manner. If an output Aspect is defined to reside on a network computer other than that of the FTA's application, the Aspect file must be transferred to the specified computer. Only one copy of a particular file should exist on the network in the location specified by the FTA design unless a transitory Aspect is in use.

Definition of Complex FTA's: We define the composition of a number of FTA's and their Aspects into a process, a *Complex FTA*. The Complex FTA provides a higher level transformation. The implementation of Complex FTA's is another important issue for the infrastructure. A *Simple FTA* is defined as an FTA which enacts a single execution process. A Complex FTA links multiple simple FTA's together so they can be executed in a predefined order. Complex FTA paths are defined by a directed graph data structure and must allow parallel paths and iterations. Multiple parallel processes can execute using a target-dependent structure to maintain a proper order of execution. Fig 2 shows an example of a complex FTA graph.

A Complex FTA is similar to a Simple FTA in that it transforms a set of input aspects to a set of output aspects. A node within a complex FTA directed graph can be either a Simple or a Complex FTA. An infinite degree of scalability is therefore available within FTA design. The designer of Complex FTA need not know or understand the internal structure of another complex FTA to use that FTA within the new design.

There are two directed graphs which represent any Complex FTA. The first is the highest level graph which shows a directed graph in which each node is either a Complex or Simple FTA. Figure 2 shows the high level Complex FTA graph. The second graph is an exploded view of the high level Complex FTA graph in which each node in the graph represents a Simple FTA. Figure 3 shows the exploded Complex FTA graph version of Figure 2 with the exploded complex FTA highlighted in grey. In this figure we have dropped the data base representation of the Aspect in the Complex FTA for clarity.

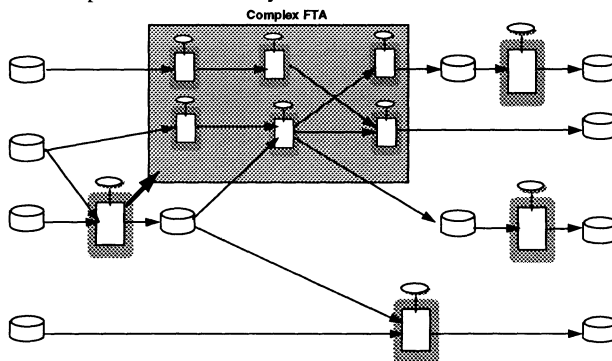


Figure 3. Exploded directed graph of complex FTA showing all simple FTA's

3.2 Entity Design

SIA is an information infrastructure which locates and communicates with network services. The infrastructure must also address the issue of identification and storage of FTAs and other information. The information required to identify, launch, and control FTA's is stored within a second key SIA module, the Librarian. The Librarian is the central point for the infrastructure and usually resides on a single computer. The Librarian provides a database which stores centralized information. A special type of object within SIA's object oriented architecture is called an *entity* or *reference*. The objects provide information about applications, aspects, and FTA's. The entities also allow information to be accessed and for messages to be sent from the entity to other network nodes. Entities can also store administrative information such as User information.

Identification of FTA's, Aspects, Applications, Functions, and Jobs: FTA's, Aspects, and applications are all network services for which specific entity classes must be defined. An FTA entity identifies the application, as well as input and output Aspects associated with the FTA. Aspect entities record the Modality of the Aspect. Application entities record the location of the executable file.

Many applications can perform multiple tasks. The specific task performed is often determined by command line parameters or input data. A function is defined within SIA as the

execution of a given application in a specific mode to perform a specific task. FTA's can be defined for each function rather than just each application. A critical distinction between applications and FTA's is that the FTA entity also includes information about the Aspect(s) required for that function.

Information about execution processes resulting from launching an FTA must also be maintained. The Job entity is introduced to provide this service. A "Job" is created when the end user requests that an FTA be launched. At this point the end user may be called upon to identify any Aspects whose physical names and locations were not predefined within the FTA's definition. After the Job is created and the Aspect locations are identified, SIA sends a launching request message to the appropriate launching server.

Administrating Access to FTA's: As networks increase in size the number of network services (i.e. FTA's) available to an end user will increase geometrically. Scalability is therefore an important issue for an agile information infrastructure. The number of available FTA's must be filtered to insure that the end user can easily find FTA's to address a particular problem. SIA introduces administrative entities to address these issues. Like other entities, administrative entities provide storage of relevant data and the ability for other entities to communicate among themselves and network services

We have identified several administrative entities which can help reduce the complexity of an infinitely scalable network. These include User, Project, and Group. User entities identify users so that the set of available information is reduced to only that required by a particular User. Projects allow FTA's, Aspects, and Users necessary to address a particular problem to be grouped together. Group is similar to but more general than the Project Entity.

Security will also be an issue for a large distributed system. Only a small number of Users should be allowed access to a given project. Administrative entities allow access to Projects, Groups, Aspects, and FTA's to be limited to the subset of Users.

3.3 Access Control

The Graphical User Interface: SIA has been implemented in modular fashion to maximize the efficiency of network communications. The Librarian is a central server existing on one network computer and providing persistent storage of SIA's entities. An end user accesses this database by enacting SIA's graphical user interface (GUI) on a local machine. Once logged in, the user is presented with a list of projects which the end user may address. A list of jobs which the user has launched is also available, allowing them to monitor and control the remote execution processes associated with a launched FTA. The end user can select a project, at which time a list of appropriate FTA's will appear. The end user can choose to launch an FTA from this point.

The end user can exit SIA's GUI at any time without affecting launched processes. If an execution process sends standard output or error messages and the user launching the job is running a GUI, these messages are sent immediately to that user's monitor. If there is no GUI open for the user, the Librarian will simply store them. When the user next enacts a SIA GUI, the messages will be immediately sent to that monitor. This allows a user to terminate the SIA

GUI during long executions and return later without losing any information regarding the execution or the results.

3.4 Design Flexibility

SIA Modules and Layers: Each “module” contains internal layers which provide specific services. Layers generally communicate with only layers directly above or directly below them in the architecture. The layers are also easily replaceable by alternate layers having different implementations. Figure 4 presents an overview of the current design.

The Librarian Module: The Librarian module the central module of SIA. All inter-modular communications pass through the Librarian to determine the network location of the message recipient. A database is located within the Librarian module to persistently record the message. The Librarian has administrative and functional components which provide access for those services to the database. The database forms the final layer within the Librarian, providing passive data, but no active services.

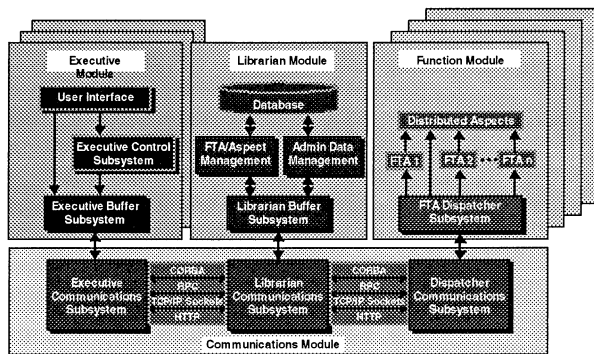


Figure 4. Overview of the Systems Integration Architecture

The Executive Module: The Executive module provides the end user access to SIA through a graphical user interface layer. The Librarian is a server for the Executive providing it with requested entities. The Librarian can also be a client of the Executive, notifying the Executive of any change in the state of launched jobs. The Executive module includes an entity “buffer” layer which provides local access to the remote Librarian entities upon request, and a control subsystem to control the execution of complex FTA’s. There can be many executive modules distributed across the network.

The Communications Module: The Executive and Librarian modules need not reside on the same machine. The Communications module provides access between the Executive and Librarian and among the other modules. Executive, Librarian, and Dispatcher Communications layers are defined within the Communications module to provide distributed communications between the Executive and Librarian modules and the Librarian and Function modules.

The Function Module: The Function module includes the set of all FTA's, launching servers, dispatchers, remote processes, and remote data available to SIA. Launching servers and dispatcher are being developed for UNIX, PC, and Macintosh operating systems. There are many Function modules spread across the network.

4 SUMMARY AND CONCLUSIONS

Various issues in the integration of information systems in an agile environment have been discussed and the problems with the traditional basis model or abstraction presented. A new basis model and definition of integration are proposed and the services required for an integrating system suggested. A brief overview of the System Integration Architecture project shows how the Agile Aerospace Manufacturing Research Center is attempting to address these issues by implementing a new architecture based on the proposed basis model.

The Systems Integration Architecture has been implemented as a framework which is easily extensible for any project that requires an information infrastructure framework. We have also designed SIA to have the flexibility to experiment with alternate technologies whose introduction may contribute to the efficiency of some portion of the project. SIA is being implemented, not as a product which is meant for market, but as a research framework which can be modified and tuned for various needs. The modularity of the design including the reconfigurability of the communications and database layers provides many advantages for research into integration issues in the dynamically changing domain of agile manufacturing.

5 ACKNOWLEDGMENTS

The author acknowledges the financial support of the National Science Foundation from cooperative agreement DDM-9320949. This work is being performed as part of the Agile Aerospace Manufacturing Research Center. The assistance of Dr. Ramez Elmasri in the data base area is acknowledged. The contributions of Dr. Marc Turcotte in the design and implementation of SIA is gratefully recognized.

6 REFERENCES

- Betz, M., "Interoperable Objects," Dr. Dobb's Journal, #220, a Miller Freeman Publication, p.p. 18-39, October, 1994.
- CIM-OSA Reference Architecture Specification,, ESPRIT Project Nr. 688, pub. by CIM-OSA/AMICE, Brussels, Belgium, 1990.
- Finger, S. and Dixon, J.R., "A Review of Research in Mechanical Engineering Design. Part II: Representations, Analysis and Design for the Life Cycle," Research in Engineering Design, Vol. 1, p.p. 121-137, 1989.
- Fulton, J.A., PDES/STEP Model Unification: SUMM," Product Data International, May, pp. 6-7, 1992.
- Goldman, S.L. "Agile Manufacturing: A New Paradigm for Society," White paper published by the Agile Manufacturing Enterprise Forum, the Iaccoca Institute, Lehigh University, PA, undated.
- Jarke, M., "Strategies for Integrating CASE Environments," IEEE Software, March, p.p. 54-61, 1992.

- I. Mills, J.J., "An Agile Information Infrastructure: Integration Issues and Approaches," Proceedings of the Fifth National Agility Conference, Boston, March 5-7, The Agility Forum, 1995.
- II. Mills, J.J., "A Basis Model for Agile Computer integrated Manufacturing," Proceedings of the Third International Conference on CIM, Singapore, July 9-14, Vol. 3, p.p. 53-63, 1995.
- Mills, J.J., Elmasri, R., Khan, K., Miriyala, S., and Subramanian, K., "The Systems Integration Architecture: An Agile Information Infrastructure," in *Information Systems Development for Decentralized Organizations*, Eds. A. Solveberg, J. Krogstie, and A.H. Seltveit, Chapman Hall, 1995.
- Nguyen, G.T. and Rieu, D., "Expert Database Concepts for Engineering Design," AI EDAM Vol. 1(2), p.p. 88-101, 1987.
- Nof, S. Y. and Papstravrou, J. D., "Decision Integration Fundamentals in Distributed Manufacturing Topologies," IIE Trans., Vol. 24(3), p.p. 27-42, 1992.
- Senehi, M.K., Wallace, S, Barkemeyer, M.E., Ray, S.R. and Wallace, E.K., "Manufacturing Systems Integration Initial Architecture Document, NIST Interagency Report 91-4682, September, 1991.
- Sthanusbramonian, T. Finger, S, and Rinderle, J, "A Transformational Approach to Configuration Design," in The 1992 NSF Conference on Design, p.p. 419-424, 1992.
- Suh, N., *The Principles of Design*, Oxford University Press, 1990.
- Takeda, H, Veerkamp, P, Tomiyama, T. and Yoshikawa, H. "Modeling Design Processes," AI Magazine, Winter, p.p. 37-48, 1990.
- Talukdar, S.N. and Fenves, S. J., "Towards a Framework for Concurrent Design", in *Computer-Aided Cooperative Product Development, MIT-JSME Workshop*, D. Sriram et al, Lecture Notes in Computer Science No. 492, Springer-Verlag, pp. 140-151, November 1989.
- Taylor, L., and Henderson, M., "The Role of Features and Abstraction in Mechanical Design," *Proceedings of the Design Theory and Methodologies Conference*, ASME, Minneapolis, September, 1994.
- "The Common Object Request Broker Architecture and Specification," by Digital Equipment Corporation, Hewlett-Packard, HyperDesk Corporation, NCR Corporation, Object Design Inc., SunSoft Inc, the Object Management Group, Rev. 1.2, 1993.
- Ulrich, K., and Seering, W., "Conceptual Design: Synthesis of Systems of Components," in *Intelligent and Integrated Manufacturing Analysis and Synthesis*, PED Vol. 25, ASME., p.p. 57-66 NY, 1987.
- Wasserman, A.I., "Tool Integration in Software Engineering Environments:" *Proc. Int'l Workshop on Environments*, F. Long, Ed., Springer-Verlag, Berlin, p.p. 137-149, 1990.
- Wunderli, M., Norrie, M.C., and Schaad, W, "Multidatabase Agents for CIM Systems," *Proceedings of the 3rd International Conference on CIM*, Singapore, July, Ed. By Winsor, J., Sivakumar, A.I. and Gay, R., World Scientific, Volume 1, pp. 13-20, 1995.

7 BIOGRAPHY

Dr. John J. Mills received his Ph. D degree in Applied Physics at the University of Durham, Durham, England in June 1965. As the Director of the Automation & Robotics Research Institute, Dr. Mills manages a multi-million institute which is dedicated to helping US industry become globally competitive. Dr. Mills is also the director of the Agile Aerospace Manufacturing Center, one of three Agile Manufacturing Institutes in the nation. Funded by an ARPA/NSF Agility Initiative, the Center is helping to define agility in aerospace manufacturing in the United States. He also serves as a member of the Board of Directors for several organizations including the ASME Manufacturing Technologies Group Operating Board(MTGOB), the Agile Enterprise Quarterly Editorial Board, the Academic Coalition for Intelligent Manufacturing Systems(A-CIMS) Board of Directors and others.

In addition to his administrative responsibilities, Dr. Mills also maintains an active research program focusing on the information infrastructures required for agile manufacturing. He is author of over 100 publications including book chapters, articles in prestigious journal, and industry reports.

Mike Brand has been the project manager for the \$250,000 Systems Integration Architecture project of the Agile Aerospace Manufacturing Research Center at the Automation and Robotics Research Institute of the University of Texas at Arlington since September of 1995. He is also the chief architect for the Systems Integration Architecture, producing all high-level analysis and design and directly supervising all lower level analysis and design. Design issues include identifying client-server partitions within the architecture, implementing architectural design changes for the development of reconfigurable architectural layers, supervising the implementation of distributed object client-server applications using the CORBA compliant Orbix software from Iona Technologies, and supervising the implementation of an object oriented database using Object Design's Object Store. Mike Brand graduated with a Bachelor of Arts degree in Physics from Earlham College in Richmond Indiana and expects to receive a doctorate in Physics from the University of Texas at Arlington in May of 1997. His graduate work involved designing and implementing an application capable of processing and rendering data about solid state Fermi surfaces retrieved from a two-dimensional angular correlation of annihilation radiation (2D ACAD) experiment.