AeroWEB: An information infrastructure for the supply chain

J. Mills, M. Brand, R. Elmasri
The Automation & Robotics Research Institute,
The University of Texas at Arlington
7300 Jack Newell Blvd. S.
Fort Worth, TX 76118
Phone (817) 272-5900 Fax (817) 272-5952
{jmills, relmasri, mbrand}@arri.uta.edu

Abstract

Information flow up and down the supply chain, particularly for complex products such as automobiles, aircraft, and weapons systems is complex, difficult, fraught with errors and time consuming. In the defense business, for example, just the engineering trade-off process in the conceptual design phase can take up to a year before all the information requested is available for the design engineering team to compare alternatives (Schwart97). Some of the problems encountered include the inability of small suppliers far down the chain to access technical data created with CAD systems to which they do not have access, inaccurate or wrong versions of technical data when the supplier eventually obtains it, and difficulty in contacting people to request clarification. EDI and geometric modeling standards (e.g. STEP) and Email have started to address these problems, but they do not address all of them. Email, for instance often results in huge CAD files bogging down a suppliers email system (often using a dial up modem) only to have it discarded because it was not appropriate to their business. EDI/EC and STEP standards focus on standards for the exchange and sharing of data, not on their smooth and efficient movement up and down the chain.

A new approach is proposed which has the potential to eliminate or minimize many of these problems. Based on the Systems Integration Architecture described elsewhere (Mills95aa, Mills96), it uses the concepts of Distributed Object computing, remote computing, Internet access, X-Windows, references and associations to facilitate (a) access to remote applications and data and (b) the flow of information up and down the supply chain. Details of SIA and an application of it, called AeroWEB, to the problems described above are presented.

Keywords

Information infrastructure, virtual enterprises, supply chain

INTRODUCTION AND BACKGROUND

Manufacturing, defined herein as the overall process by which ideas (or customer demands) are converted into products, is undergoing the most radical transformation since Ford introduced the assembly line. Substantial changes in the way Manufacturing companies operate, the way people interact within those companies and the way those companies interact with their customers, suppliers and competitors are all occurring. For instance, virtual or extended enterprises are forming, operating and dissolving all within a time frame which would have been thought impossible a few years ago. Virtual enterprises are formed of core competencies of a variety of companies, brought together into a new enterprise to meet some short lived market demand (Goldman95). Supply chains are becoming extensions of the parent enterprise, forming what has been called "Extended Enterprises".

These changes are both major problems and major opportunities for Manufacturers. As the Next Generation Manufacturing Project puts it: "Unprecedented, interrelated changes in the global business environment are creating entirely new success factors for industrial competition (Hardt97).

One significant problem for virtual enterprises and within extended supply chains is the integration of their information systems (Adams98). While efforts to create standards for geometric data exchange and sharing (e.g. STEP), operating systems (e.g. POSIX), computer languages and object message passing e.g. (CORBA) (omg98) are underway across the globe, there are also many other existing approaches to these same problems (e.g. DXF and IGES for geometric data, Microsoft Windows, the Distributed Common Object Model, etc). Consequently, the environment in which an organization wishing to form a virtual or extended enterprise or to establish a supply chain integrated into their own operations, is heterogeneous, where just about everything is different. This is also a fairly dynamic environment since the virtual enterprise lifecycle is rarely more than a year and often much shorter. Current approaches take far too long to be useful in this environment.

Another problem, which smaller suppliers with multiple large customers face is access to, advanced computer technology. To solve the integration problem of their particular supply chain, many large companies demand that their suppliers purchase the CAD systems that they use. These systems often cost tens of thousands of dollars and require advanced computers. A small company with several such customers is then faced with having to purchase several such CAD systems, but only use them occasionally.

The research issue intrinsic to these problems is as follows:

- What is "Integration" in this dynamic, heterogeneous environment?

A second issue which is a complement to the first in that if we define "Integration" appropriately, we can also address it simultaneously, is

- How can we provide the small suppliers access to the same advanced technology that their largest customers have?

This paper presents our approach. First we present a basic philosophy or general approach which we have called "Integration from the User perspective", then present some of the requirements for a system implementing this philosophy. Then an Infrastructure called the ***Systems Integration Architecture, which supports this approach,*** is presented. Finally some results from applying this infrastructure in a demonstration of an engineering trade-off process involving a supply chain are presented and the results discussed.

APPROACH

Underlying assumptions

Our basic approach is based on defining "Integration" from the user perspective - a user-centric approach. Many of the previous attempts to define integration have been based on what we term "implementation approaches". That is, they have focussed on the details of how data is provided to users or how processes should be "integrated". The traditional view of integration of the central database from which views of the data were extracted is such an approach.

Defining integration for the users perspective provides quite a different perspective on integration. From the user's perspective, a user simply would like to have access to the data and functions they require to perform their task. They do not want to worry about where the data resides, how it is formatted or stored or where the application is. They would prefer to have a simple interface, which presents to them an icon representing the data, and/or function they need at that point in time or a menu with the name of the data or application in it. They would then like to be able to access either of these with a simple command (voice or mouse click). This is the approach taken within Microsoft Windows and Macintosh. It is perhaps one reason why these approaches have so much appeal to the average user. Our approach is to expand this concept to data and functions located anywhere on the net.

Requirements

The first and major requirement for a user-centered, dynamically reconfigurable heterogeneous information system is that the user *must* be able to access data and functions without regard to where they are stored, how to access them or what data model is used, what data format the data is stored or how, physically (flat file, data base, etc), it is stored.

Note that this requirement makes no mention of such things as reconfigurability, interoperability etc, which appear so frequently in analyses and discussions of "integrated systems"(Senehi91, Chen93. In our view, this type of requirement is imposed because of the implementation view taken. They are almost, but not quite, requirements imposed by the approach taken.

This major requirement has secondary requirements as follows. The system *must* be able to record and keep track of functions and data sets. This record must include not only the machine and path where they are located but how to access it if it is a data set and how to launch it if it is an application. It *must* also contain other information about the entities that is important for the user and the functions they perform. We discuss integration in terms of functions instead of applications because it is a function, which a user needs to perform a task. Applications supply functions, usually as an ad-hoc collections. This focus on functions moves us towards what Johnson has termed "the ubiquitous service environment" in which users will have access to functions anywhere on the net (Johnson96). A function can be as small as an applet to add two integers together or as large as a major Enterprise Resource Planning system.

We use the term "data sets" to avoid identifying the data with a particular data storage approach (e.g. data base, simple file) or format. A data set is simply a collection of data, which is useful at some point in the product realization process.

We call functions, applications and data sets "integratable entities", since these are what must be "integrated" if we are to have an integrated system. There are other "integratable entities" which will be described later. We call the records the system must keep about these entities "references". They are essentially pointers to the entity with additional information.

If the system has to be able to record and keep track of integratable entities, then a further requirement is that it *must* be possible to create, delete and edit such references either by a user or automatically.

Functions require data. Indeed, elsewhere we have suggested that a new basis model is needed for "integration". The proposed basis model, called the Transformation of Aspects and Representations model, notes that all software simply "transforms" data sets into other data sets of different form or type(Mills95b). This implies that the system must supply some mechanism for identifying and storing relationships between the functions and the data sets. This is an extension of the idea of an association between the file extension in Microsoft Windows and the application, which created it. We also call our relationship an "Association" but it is more generic as described later.

If the function the user requires resides in an application on a computer remote from their site, the system *must* be able to launch remote applications to provide the function to the user at their machine. It *must* also be able to display the output of that remote application on the user's screen. This could be a simple telnet session in a window or a full 3-D graphics window session.

If the data set which the user requires is not on the same machine as the application providing the function required, the system *must* be able to provide a data set to that application regardless of the location of the application or data set. The users should not need to know how to do this. This implies that the reference *must* contain information on what format, data model and the physical storage mechanism, etc the application requires for that data set.

If the data set is not in the required format, physical storage method etc, this requirement also implies that the reference *must* be able to provide automated translation of the data from the format, storage system, etc it is in to that required. It should not matter to the user whether the data set is simply in a file or is managed by a product data manager. On a side note, we believe that the system itself should not provide product data management functions but it *must* allow for

them. We regard the services typical of a product data management software as just functions, which a user can use to act on data.
The system *must* know what specific data set is appropriate for the application at the instant of launch. If a user frequently performs an activity requiring a specific function, they may accumulate over time many different data sets, which that function, requires. This makes it difficult, if not impossible for the system to recognize what data set is required for the specific instance of launching that function. The system *must*, therefore, provide some mechanism for limiting the data sets available to that function.
For some organizations, the management may like to organize the functions into specific processes, also called workflow management. The system should allow for workflow management concepts, but need not provide workflow management functions itself.
Clearly, for a user to access remote functionality and data, any machine involved *must* be physically connected to other involved machines. This implies that the system *must* provide physical network connections between any machine involved. A machine can be "involved" by having a user access the system through it, or by having applications or data sets reside on it. Further, the system *must* provide communications between these machines. Note that we have not specified what kind of communications at this point.

Technical Approach

We have been developing an information infrastructure called the ***Systems Integration Architecture (SIA)*** which meets these requirements and is briefly described below. Details are available elsewhere (Mills95a,Mills96).
Basically SIA is a modular information infrastructure which "integrates" a short list of integratable entities. The integratable entities include data sets, functions, applications, entities called Functional Transformation Agents, users and projects. Applications provide the functions required by the user. In this initial implementation we have not made clear distinctions between applications and functions. Essentially, at this point we use functions and applications interchangeably. Our long-range intent is to provide individual functions from applications to move us towards the ubiquitous service environment. Functional Transformations Agents or FTAs are objects, which include both the function and the input and output data sets. This is the only entity, which is launched by the system. The FTA concept provides one way of bringing together a data set and a function or application. It is included to implement our ideas that functions transform an input data set into an output data set. It also allows us to account for workflow management functions in the future but this feature is not yet fully implemented. Users and Projects are required to provide context to limit the number of functions, FTAs and data sets a user can see at any one time. Thus a Project can have many users each of whom has access to only a limited number of all the possible functions and data sets which may exist on the system. Each user can be a member of many projects, but can access only a limited set of functions

and data sets depending on the active project. The reference to each of these entities is an object within SIA.

SIA also allows for Associations to be defined among the References to the Integratable Entities. This allows data sets to be associated with functions and users, projects to be associated with users, etc, all focussed on "integrating" these various entities.
Figure 1 illustrates the various modules of SIA. The ***Librarian*** consists of objects, which interface the rest of the system with an object-oriented database. This database provides for persistent storage for the references and associations among the entities in the system. Note that the Librarian stores only references: pointers to entities and information about these entities. In this, it is close to but contains more information than the naming service provided by CORBA implementations.

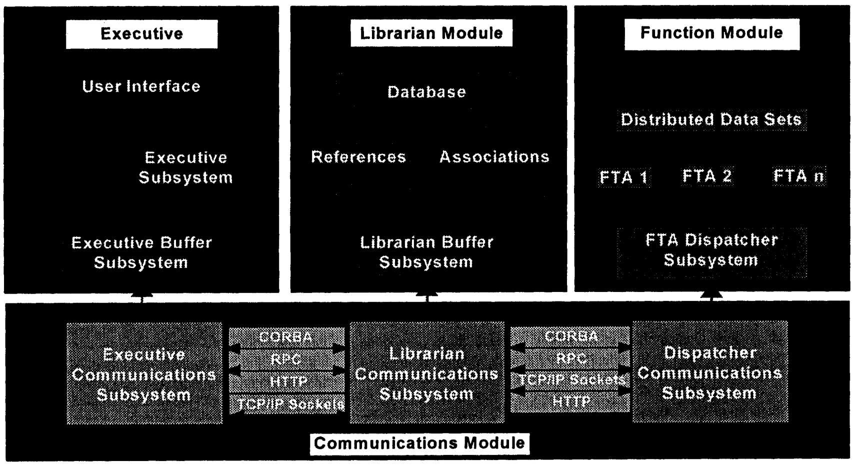


Figure 1. Modular design of SIA facilitates change

The ***Communications Module*** allows for message passing among the other components in the system. Currently it is based on the Common Object Request Broker Architecture (CORBA) which allows for - among other services - a standardized way of passing messages among distributed objects. CORBA sits on top of a physical network which uses both TCP/IP and IPX protocols. Each machine in the system is required to have either client CORBA software (for a user) or a CORBA server (for applications and data sets). Other distributed object message passing protocols such as http, Distributed On-line Linking and Exchange, is being investigated.
The ***Executive*** component consists of client CORBA software, which interprets actions on the Graphical User Interface to commands, which the other modules understand and using IDL, transmits them across the network. The Executive also stores certain, frequently accessed, information from the Librarian locally to minimize network traffic. Commands recognized by the Executive include: Getlist,

Storelist, etc for accessing information in the Librarian, and launch, stop, etc for controlling remote functions.
The ***User Interface*** module interprets actions by the user into commands, which are sent to the executive. There is one user interface and one executive for each user in the system. A user interface can be of any type and in fact we have integrated several different user interfaces with the Executive.
The ***Launcher*** or ***Dispatcher*** module is a server object based on CORBA, which sits on each machine where there is an application or a data set. This software takes the IDL Launch message from the system and converts it into a command to launch the application. It also checks if that function requires a data set, retrieves information from the Librarian on its location and sends a command to that Launcher to automatically transmit it over the network. The launcher software essentially spawns a new process which then runs independently of the user so that the user can let applications run without having to be logged on at all times.
SIA has been implemented in C++ on Unix Workstations. The Communications module uses ORBIX from Iona Technologies and the Librarian is based on ObjectStore. It is operational on a Digital ALPHA with the Ultrix operating system, a HP with HP-UX, several Suns with SunOs and Solaris, a Silicon Graphics Indigo and an Octane, both running IRIX. Access to SIA from PCs is currently provided by X-Windows software such as Exceed, which allows the PC user to log on to SIA on one of our Unix machines. Various linked X-Window User Interfaces, developed in Tcl/Tk allow users to perform the following functions: create, edit and delete any of the References to integratable entities; associate specific entities with others, launch applications by clicking on FTAs. Applications, which have been launched from various machines, include typical X-Windows demonstration software such as Xclock, Xeyes, Emacs, and such full applications such as AutoCAD, Pro-Engineer, Unigraphics, CIMstation, and IDEAS.
Two separate user interfaces were initially developed for early testing. The first was a simple X- Windows based interface developed using Tcl/Tk. This was used in early development and testing but has since been abandoned. A second, text based command line interface was also developed to allow the developers to test various components without the complexity of a graphical user interface. This is still in use. These interfaces plus that described below illustrate that the design of the system with the executive module separated from the user interface through a set of Application Programming Interfaces means that SIA can "Integrate" user interface quit easily.
SIA provides a basic framework for integration from the Users' perspective. To address the problems outlined in the Introduction and to test it under fairly realistic conditions, we devised a scenario for the engineering trade-off process across a supply chain. This project was called AeroWEB. In this process, the designer requests information and quotes from other engineers and suppliers, and requests reviews from supervisors. The suppliers request information and quotes further down the supply chain. The engineer then conducts engineering trade-off studies comparing various alternative designs. To develop the scenario for this process we

enlisted a set of potential users including designers, procurement officials from several large aerospace manufacturers and several small suppliers to that company. In early meetings with these potential users, we identified the processes, which apply in the supply chain. Upon analysis, it was recognized that the process of requesting information of any kind (quotes, product details, etc) is recursive: a person sends out a request (by telephone, email, etc), to a recipient or recipients, who either accept it or reject it, perform some activity and then respond. The request/respond cycle is identical for anyone in the recursion. Only the data required and the activities they perform are different. The activity they perform can be to send a request for information further down the supply chain and so on. The recursion closes when each person in the recursive chain has responded. A particular level in the recursion only closes when all the lower level recursions have closed. We also identified the need to allow users down the supply chain to request clarification up the supply chain. These clarification requests were also identified to be recursive in nature.

This process recursion turned out to be important because it allowed us to design a single set of user interfaces which allows all users to send and receive different kinds of data and applications, and to clarify, accept, reject and respond to requests. This simplified development substantially. Thus, the adaptation of SIA to the needs of an "Integrated" supply chain predominantly focussed on developing specialized Graphical User Interfaces.

The concept behind "AeroWEB" was to send the References up and down the chain and to allow users to change the Associations of these References when they needed to. No data would be sent up an down the supply chain until it was actually required. The user group indicated that often several files were needed, particularly when quotes were requested. For example, a request for bid "package" might contain text documents describing requirements, CAD files and text terms and conditions. We realized that a "package" is a useful concept for the information system and that it could be implemented easily as a kind of project within SIA. This Package entity contains references to data sets, users and even to applications. The project object was modified to allow for this. These "packages" are then what is sent up and down the chain.

RESULTS

The scenarios

The purpose of the scenarios was to test the infrastructure under a wide variety of circumstances to see if the users definition of "Integration" has been achieved and if the approach solved the integration problems across the supply chain. We have tested the infrastructure and its services with three related, yet increasingly complex scenarios. The first involved a small number of files and applications and only three organizations/roles: The designer, a procurement official and a single supplier. The collaboration diagram is shown in figure 2. Upon viewing the demonstration of this scenario, the user group indicated that the approach had the potential for facilitating the flow of information across a supply chain. The demonstration also showed that suppliers using PCs with low cost X-Windows software could access the system and launch remote applications like AutoCAD, Unigraphics and Emacs to view drawings and textual information without having the files on their machines. This demonstration involved files, which were created ahead of the demonstration and registered with the Librarian by the students.

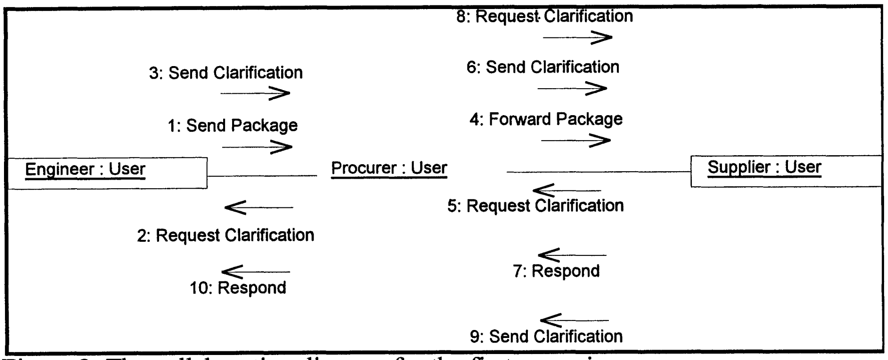


Figure 2. The collaboration diagram for the first scenario.

The User group made several recommendations for improvements. These included (a) increased complexity of the scenario to make it more realistic, (b) having the ability to edit or modify exiting files and to create new files and register them with the system, and (c) adding more functionality for the users.

The second demonstration involved a more complex scenario involving a couple more people in different roles within the prime contractor and at least two tiers of suppliers. A significant feature of this demonstration was that the degree of complexity was increased simply by registering more people with the Librarian and assigning them roles. The extra loops in the recursion were added simply by sending a package to another user who then forwarded it to a third user, and so on. More functions and data sets were added to the system simply by registering them with the Librarian and creating the appropriate Associations. This scenario also used previously prepared files of information.

The third demonstration involved more people doing "real" work at each stage, with those people registering the files they create with the Librarian before shipping them off in packages to other users. Only a few prepared files were used. A Chief Engineer review cycle, and several more clarify/respond cycles were added. The collaboration diagram for this scenario is too complex to present here.

The complexity of the scenario, Figure 3, was easily increased by simply registering more people with the Librarian.

Changing the demonstration scenario requires nothing more than identifying the people who will be involved, identifying their roles and adding them to the system. The workflow is then controlled by adding the person or persons to the recipient list in the package. Adding functions/applications and data sets is also straightforward.

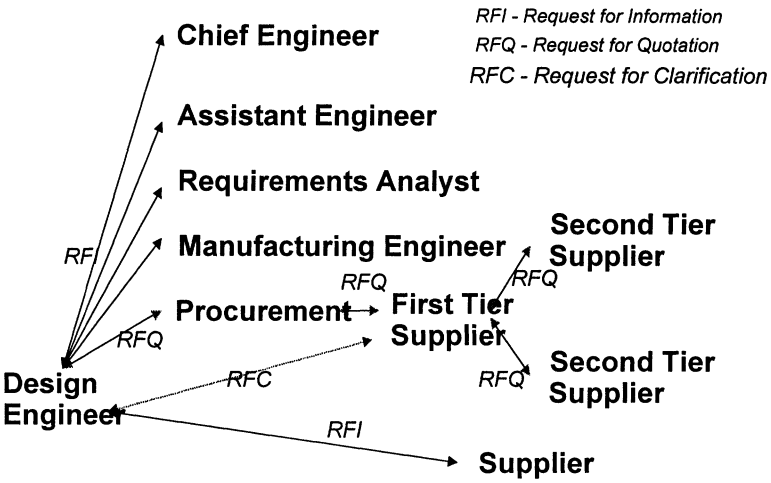
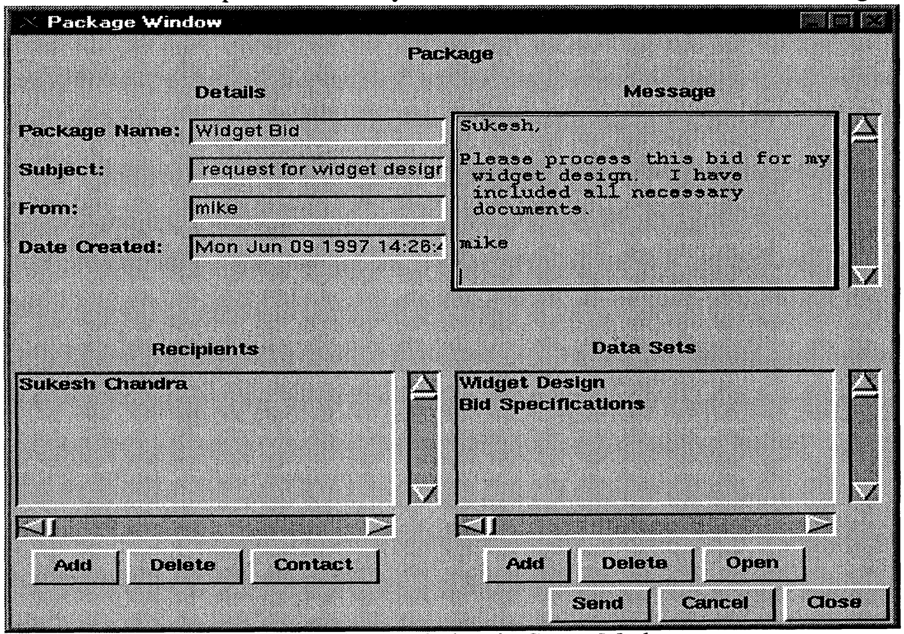


Figure 3. Illustration of the Final AeroWEB Scenario

The AeroWEB package user interface has three modes. Figure 4 illustrates it in the create mode. This same interface is used for both sender and recipient but it changes slightly depending on the state of the system. This figure shows the information, which can be included in the package. The details box requires the name and the description filled in by the user. The user can also add a message.



Package Name: Widget Bid
Subject: request for widget desigr
From: mike
Date Created: Mon Jun 09 1997 14:26:

Sukesh,

Please process this bid for my widget design. I have included all necessary documents.

mike

Sukesh Chandra

Widget Design
Bid Specifications

Figure 4. Illustration of the Package Window in Create Mode

All messages added when the package is forwarded are included in this box. In the create mode, the Add button under Recipients and Data Sets boxes brings up a small window which permits users to add the appropriate person or data set to the package. This package is then sent to the recipient by clicking the "send" button.
The review mode is what the recipient first sees on opening up a package. In the Review mode, the only actions the user can take are to open a Data Set, accept or reject the whole package or clarify. Instead of the Send and Cancel buttons, the Review mode has Accept, Reject, and Clarify buttons.
The list of entities in the data sets box are all launchable either by the sender - to check contents - or by the recipient before they accept the package. This allows the recipient to review contents before accepting. The data sets may be on some remote machine and the applications with which they are associated can be on a third machine. Clicking on the Data Set and then "Open" will automatically transfer the file from the machine it is located on to the machine where the application is located and launch that machine with that file. In the review/accept mode, the Add and Delete buttons under recipients and data sets are muted because the recipient can only launch the data sets (and their associated applications). They must click on the Accept button before they can do anything more.
The Accept button transfers the association of the data sets to the recipient and allows them to then associate them with other applications they might use. This is done in another window, which is not shown here. This action also changes the mode of the package window to a third mode called the "Accepted' mode which is discussed below. The Clarify button starts a small window which allows the user to enter a small message which is then appended to the package message and the whole package sent back automatically to the sender. Note that "Clarify" button does not appear on the Create Package Window. In the accepted mode, two of the buttons at the bottom of the Review window change to Respond and Forward. Clicking on the Respond button allows the recipient to respond to the sender. In between accepting and responding he can create his own packages to request information or quotes from other people on the system (i.e. sub-tier suppliers), register new data sets with the system Librarian and generally perform those activities they normally perform when responding to requests. The user can also use the Forward button to send the package on to other persons from whom he might need help. He can add data sets, and a message to the package before sending the package on. Before clicking on Respond or Forward, the user can delete data sets and/or add more to the package. For example, they might want to add their formal Quotation or a file with the information requested. They can also create new packages to enclose the data sets just received and send these on to another recipient.
This AeroWEB application of SIA has been demonstrated on several occasions to potential users. The degree of complexity can range form the full blown scenario which takes about 20 minutes to a simple demonstration of creating and sending a package to user who then launches one or more of the data sets. It has been demonstrated within the laboratory environment using the internal network, and from a PC logged onto the internal network through both a 28.8kb modem and a 2

channel ISDN line. It has also been accessed internationally across Internet by a user logging on to one of the machines mentioned above. This last test proved quite challenging and identified a number of factors, which had not been considered before. However, these users were able to log on, create and send a package and launch a data set, both of which, of course, were located on our machines. This test has several further steps planned.

DISCUSSION

The first question, which must be addressed, is whether the tests under the scenarios described demonstrated "Integration" from the users perspective. The second question is: Does SIA/AeroWEB facilitate the information flow up and down the supply chain and allow all users access to the same software applications. To answer the second question first, the users all agreed that such a system would indeed make their work a lot easier.

In these tests, a variety of users on totally different machines at different locations were able to create files with different applications and share them with others, chosen from lists, by sending packages containing references to them. The recipients of the packages were able, by simply clicking on a menu item, to launch the application associated with that data file regardless of its location, and regardless of the location of the application. The interface of the application appeared on the screen in front of the user. At the basic level, therefore, we claim that our approach does facilitate the information up and down the supply chain and allow everyone to access the same technology. We also claim that our tests did demonstrate Integration from the Users perspective. However, there are caveats, which must be considered. First, the tests really did not move data sets from one application to another. Changes to data sets were made with the application, which created it, and these new files were registered with the Librarian and with the original application again. To provide full "integration", the system must allow users to associate data sets with other applications which then raises the issue of the correct data format and storage and access methods of the data set for that new application. This aspect of "Integration" was not tested. We also did not address issues of simultaneous sharing (i.e. collaboration) of data files or of data stored in other ways such as in data bases.

Further, the method of registering a new data set with the Librarian and associating it with an application is clumsy involving several windows and entering data such as the IP address of the machines where it is located, its path and name etc. We have realized that that information can easily be acquired automatically and have recently tested software that does that. This raises issues in version control and naming conventions, which we have not yet addressed.

A further caveat is that the current version of SIA does not allow applications and data residing on PCs to be accessed. We believe that we know how that could be achieved and are currently addressing this limitation. For instance, we have ported the GUI and Executive to PC and this development is currently under test. Another limitation is that we require our dispatcher software and ORB on every machine, which contains an application or data set. Some users may not want such expense and may have http or ftp servers available. We have started investigating how these different servers may also be included in our view of integration.

CONCLUSIONS

We have demonstrated an initial version of a system, which will allow integration from a user's perspective. While not providing "full" integration in the sense that we cannot integrate activities into processes, transfer files to different applications or accommodate different server technologies, the system does provide basic mechanisms for allowing users to access data and applications regardless of their locations. We are currently testing the infrastructure or planning test with several organizations to see if it can solve their problems.

ACKNOWLEDGEMENTS

The authors would like to thank the National Science Foundation who provided funding for this project through Co-operative Agreement No DDM 9320949 for the Agile Aerospace Manufacturing Research Center. We would also like to thank the industry members of that board for their support of the project, and to the users who helped guide its development.

REFERENCES

Schwart97	Schwartzkopf, B., Lockheed Martin Vought Systems, Private Communication, April, 1997.
Mills95a	Mills, J. J., An Agile Information Infrastructure: Integration Issues and Approaches, the *Proceedings of the Fifth National Agility Conference*, Boston, March 5-7, The Agility Forum, p.p. 267-276, 1995.
Mills95b	Mills, J. J., A Basis Model for Agile Computer Integrated Manufacturing, *Proceedings of the Third International Conference on CIM*, Singapore, July 9-14, Published by World Scientific Publ., Singapore, Vol. 3, p.p. 53-63, 1995.
Mills96	Mills, J. J. and Brand, M., The System Integration Architecture: A Framework for Extended Enterprise Information Systems, the Proceedings of the Second International IFIP Working Conference on Design of Information Infrastructure Systems for Manufacturing, Eindhoven, 1996, Publ. by Chapman Hall, p.p. 144-168, 1997.
Goldman95	Goldman, S., Nagel, R.N., and Preiss, K., *Agile Competitors and Virtual Organizations*, Publ. By Van Norstrand Reinhold, New York, 1995.

Hardt97 Hardt, D., Patterson, R, and Neal, R, "Next Generation Manufacturing, A Framework for Action: Executive Summary," published by The Agility Forum, Leaders for Manufacturing, and Technologies Enabling Manufacturing, January 31, 1997.

Adams98 William Adams, President of Agile Web, Bethlehem, PA, USA, personal communication, January 1998.

omg98 Anon, The Common Object Request Broker: Architecture and Specification, Revision 2.2, Publ By the Object Management Group, February 1998; available at URL - http://http://www.omg.org/corba/corbiiop.htm.

Senehi91 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.

Chen93 M. Chen and R.J. Norman, "A Framework For Integrated CASE," IEEE Software, March 1993.

Johnson96 Larry Johnson, Raytheon TI Systems, Personal Communication, Summer, 1996.

BIOGRAPHY

John J. Mills is the director of ARRI and AAMRI (Agile Aerospace Manufacturing Research Institute). He also is Fort Worth Chamber Foundation Chair in Automation & Robotics. He is a member of Industry Steering Board, Technology Enabling Agile Manufacturing, DOE and Board of the Academic Coalition for Intelligent Manufacturing Systems.

Ramez Elmasri is a professor in the department of Computer Science and Engineering at the University of Texas at Arlington. He has worked with the Information Technology Group at ARRI (Automation & Robotics Research Institute) for several years on Systems Integration Infrastructures. He is co-author of the textbook "Fundamentals of Database Systems", and has over 60 research publications.