DESIGN AND MANUFACTURING

IN A DISTRIBUTED COMPUTER ENVIRONMENT

Nicholas M. Patrikalakis, Kawasaki Professor of Engineering
Chryssostomos Chryssostomidis, Doherty Professor of Ocean Science and Engineering
Lieutenant Konstantinos Mihanetzis HN

Massachusetts Institute of Technology, Department of Ocean Engineering, Cambridge, MA, USA

"Our visions begin with our desires"

Audre Lorde

Abstract

The design, analysis and manufacture of complex system, such as ships, automobiles, and aircraft, involves processing on a variety of computers using different programming languages running on different operating systems as well as massive data exchange between different computational processes.

The Internet and the emerging distributed computing paradigm provides a unique opportunity for Simulation Based Design (SBD) spanning the entire range of ship design, analysis and actual manufacture. Due to previous investments, there are already large monolithic computer programs that address specific aspects of the above processes. These programs operate in a specific environment and their execution requires specialized expertise. Typically, a complex computational workflow that involves several programs, needs to exchange a considerable amount of data in order to accomplish the overall task. Using latest software technologies, based on Object Oriented Programming (OOP), modern protocols such as the Common Object Request Broker Architecture (CORBA) or the Distributed Component Object Model (DCOM), and the acceptance of metadata standards such as the already well developed STandard for the Exchange of Product data (STEP), it is possible to link together these different programs and make them cooperate. With this new architecture, it is possible to provide functionality such as remote execution and control. Furthermore the new architecture leads to a service-style method of design, analysis and manufacture as opposed to the current product-style methodology pervasive in the industry.

So far we have experimented on a small scale with this type of architecture for a range of applications in the Ocean Sciences and Ocean Systems Management and have provided an experimental system that linked together previously isolated computational programs.

Introduction

It is common knowledge that scaling requires standardization. At the end of the 19^{th} century, when the railroads in the US had began to cover a large portion of the mainland, it became evident that it was necessary to move to a standard for operating them. They moved to standards for gauge width[1], track gauges, coupling hooks, repair items and much more. It took almost 45 years from the introduction of the first public railway in the world[2] to establish a standard for the gauge [1].

Today standards are necessary for anything that needs to be used in many different places and by many different operators. Screws, iron plates and pipe couplings are all coded and standardized. The list is large and is growing. We need standards to predict the behavior of the item to be used. We need a standard to easily find what kind of screw we need to hold two flanges together. On this notion even professionals are standardized. We use a trained lathe operator to work the lathe. If a person fulfills the specifications for working with the lathe we can assign him to work with it. However the need for a person to be a certified lathe operator gets value only in the perspective of a scaled system. An individual person that has a lathe at his home workshop has no need for a certificate (standard). On the other hand an industry seeking lathe operators needs to identify that the person is capable of working with it. It is obvious that standards get value proportionally to the scale of their use.

The information technology revolution created a field for new developments and has already accomplished a great deal. It is told that the last ten years of the information technology revolution have influenced our civilization more than the 100 years of the industrial revolution. With the emergence of the information technology revolution there was a new field for standards to be built. In the early 80s there were no operating system standards. Computer manufacturers used to supply their custom built operating system with the computer they sold. Then the progress of programming languages and the introduction of new ones brought a new field that produced more "entropy" in the computing environment. Recently a new issue has arisen. The increased amount of data used today from computers either to work with, exchange or store has unveiled a new and difficult problem to be faced.

The understanding of the problem can be viewed in the same way that the industrial revolution viewed the need for standardization of mechanical parts. If the system has to scale it has to be described by a set of standards. Up to today, scaling was mostly achieved by increasing the computational power of individual computer systems. Although this seems to be a technique that will continue in the foreseeable future, it is not the only and the most efficient way to move ahead. As an

---

[1] Gauges used varied in width from 4 feet 8 1/2 inches (it had first been adopted in England by the Stephensons, because it was based on the width required for a horse to walk within the rails of wagonways) to 4 feet 10 inches, 5 feet, and even to 6 feet gauges. During the 1870s, most of the railroads in the northeastern part of the United States standardized their rails to the 4 foot, 8 1/2 inch width.

[2] The first public railway in the world was opened in 1825 in England on the Stockton and Darlington line.

example, in the case of a shipyard, we could wait for a future powerful system that could handle all the computational needs including CAD, CAM, logistics, security, manufacturing monitoring, etc and put them all to interact in this very powerful machine. If all the software was built from the same vendor, the need for standards of communications between computers and applications would be unnecessary.

However, today there is a better way of making significant progress. Based on the ancient wisdom of “Divide and Conquer” the same system described above can be broken down into parts and distributed into many machines. With a set of protocols for communications, application identification and data interchange, we can build a network of computers that will cooperate in an efficient way. On the basis of this philosophy we initiated the Poseidon project.

The Poseidon Project

The Poseidon project funded by Sea Grant and the National Oceanographic Partnership Program (NOPP), is being developed by MIT’s Design Laboratory [10]. The initial idea was to find the means to link together existing applications that were developed during the last few years and were related to the prediction and monitoring of the ocean environment. In the context of the Poseidon computer environment, it will be possible to link applications with other applications overcoming the problems of different operating systems, programming languages and data storage formats. Applications and databases are distributed around the world and the Internet has been used as the standard medium of communication. With the Poseidon system, a user will have the option to open a standard Internet browser and use it as the user interface to link distributed applications and prompt them to interact while sending back to him the results he needs.

Due to the fact that the Poseidon system utilizes technologies such as the Internet and IP protocols, it will be easy to set up a distributed computational environment on a “protected” network using the existing Intranet of a corporation. In the case of a shipyard, the shipyard’s Intranet can be used to enclose the distributed environment and at the same time protect it to certified users only.

Technologies Underlying the Design of the Poseidon System

Common Object Request Broker Architecture (CORBA)

In May 1989, eight US companies (3Com Corporation, American Airlines, Canon, Inc., Data General, Hewlett-Packard, Philips Telecommunications N.V., Sun Microsystems and Unisys Corporation) founded the Object Management Group (OMG). The role of this group was the establishment of industry and detailed object management specifications to provide a common framework for application development. In October 1989, OMG began independent operations as a non-profit corporation. Today OMG has more than 800 members from the software industry and other institutions. The main product developed by the Object Management Group was a set of specifications code named CORBA (Common Object Request Broker Architecture).

CORBA has been noted as the "the most important (and ambitious) middleware project ever undertaken by our industry" [2]. The Common Object Request Broker Architecture (CORBA) was designed by the OMG to solve the problem of interoperability among an increasing number of hardware and software products. At a very abstract level of description, CORBA allows applications to communicate with one another no matter where they are located nor in what language they are written. The most important part of CORBA is the Object Request Broker (ORB). The ORB is responsible for all the required communications between two applications (objects) that need to interact. Although this can be a complicated operation (especially between heterogeneous platforms and programming languages) the ORB copes with a large part of the complexity, making the deployment of new applications on the network a simple task. In a client/server relationship, the ORB is responsible for all of the mechanisms required to prepare an object implementation for receiving a request, and for communicating the data making up the request. The client does not have to be aware of where the object is located, its programming language, its operating system, or any other system aspects that are not part of an object's interface. The ORB provides an extra flexibility that allows programmers to choose the most appropriate operating system, execution environment and even programming language to use for each component of a system under construction. More importantly, it allows the integration of existing components.

It has to be noted that two more technologies exist which are related to the functionality provided by CORBA. These are DCOM and Java's Remote Method Invocation (RMI). In the first case DCOM is similar to CORBA and developed by Microsoft. However the late introduction of DCOM compared with CORBA and its limited functionality (especially in its first editions) established CORBA almost as a standard for the development of applications that required transparent communications and object linking. In the case of RMI, the Application Programming Interface (API) provides far less functionality than CORBA. RMI was a first approach to the philosophy of object interaction over a network. However the CORBA standard has become an integral part of the Java API in the latest editions.

Java

Java is a relatively new programming language. It is often described as the World Wide Web (www) programming language. It was created in 1991 (and since updated) by SUN Inc. [3] and is widely known due to the extended features it provides for developing applications that run across networks such as the www.

SUN claims that Java is a "write once run anywhere" programming language. Although this has proven to be not 100% true, it is very close to accurate. Using the Java API a developer can write applications that are able to be executed in almost all the available platforms and operating systems without any changes to his code. SUN was able to achieve this by the introduction of the Java Runtime Environment (JRE). By porting platform specific requirements into the JRE, SUN was able to remove them from the Java API. SUN focused a lot of the Java capabilities on networking and Graphical User Interface (GUI) design. The latest API (1.2) includes an extended set of classes that provided functionality for designing complex 3D graphics and using advanced multimedia capabilities.

Another interesting and very useful characteristic of the Java language is the use of applets. Applets are applications that can be executed in the context of a Java enabled browser on a client machine. Applets have become very popular because they allow the execution of an application that was downloaded from the Internet without breaking into the client's computer file resources (file system, hard disk, etc).

The Poseidon Architecture

As described before, the concept behind the Poseidon system is to provide the means for linking and executing applications (including data exchange) over a network (Internet) and with control from a standard internet browser. This idea would help to use large software programs that can't be executed on a personal computer with limited computational power. It will also be very helpful since the user will be able to bring together information from all over the world. To succeed with such a goal, the Poseidon is using CORBA to overcome communication difficulties that arise when linking different computer platforms and operating systems. The user interface is built in Java so that it can become an integral part of an HTML (HyperText Markup Language) document. This was necessary in order to make possible the use of available standard browsers.

The architecture of the Poseidon system can be reviewed in Figure 1. The figure describes the elements that take part in a distributed computation over the Poseidon network. In such a network there are three basic elements:

- The client that requests a service
- The sources that are available to provide services and
- The middleware that is used to link all together.

The client is usually a single user that needs to get some results (visualization of sea properties in specific regions, etc). The sources are objects that provide specific and fully described functionality. These are software applications that provide services such as sea-bottom grid-creation and ocean current prediction, and data sources such as databases and monitoring equipment (satellites, buoys, etc). Each service component in the Poseidon environment is described by means of a set of data that the service provider creates. The middleware is everything else that is required to make the clients and sources interact. Part of the Poseidon system (belonging to the middleware part) is the Poseidon Repository (PR). The PR is used to keep track of all the services that are available on the network. It also stores the information about the services that the service provider created. In this way, there is a central point where someone can refer to identify which services are currently available.

As Figure 1 shows, a user by means of his personal computer connects to the Internet and links his browser to the Poseidon homepage. The HTML page that is sent back to the client includes a Java applet that encapsulates the client side functionality of the Poseidon system. As soon as the applet starts on the client's machine, it interacts with the Poseidon Repository and asks for all the available objects that can be used. The PR replies and the applet creates a list of all the available objects on the client's screen. The client then uses the applet's design area to create a workflow by placing objects from the list into the design area and linking them together. When the client has finished he asks the applet to tell the services to start. The services then start up and interact with each other according to the workflow that was created by the user. When they have finished they respond back to the client as described again by the workflow.

The underlying technology for achieving the interaction of these applications that can be running on different machines with different operating systems and written using different programming languages is CORBA. By wrapping the existing applications with a CORBA layer we transform all these diversified objects to a standard type of object which can be linked. Based on the fact that all available objects are fulfilling the CORBA specifications, it is possible to use the Object Request Broker (ORB) to handle the communications.

The philosophy of linking objects together is very similar to that found in control theory. Objects act as transfer functions getting a number of inputs and supplying a number of outputs. The outputs are the controlled products of the inputs. In a control system, an engineer combines various components (transfer functions) to get a desired output. In the Poseidon system the user combines various applications to produce the desired output. For example, a service object could be an application that calculates the mean of an array of numbers. The input would be the array and the output would be the mean value of them.

In most cases of control systems, the combined components are physically close to each other. In the Poseidon system the components in most cases are far from each other. In a control system, components are linked (most of the time) by means of direct contact. For example a mechanical drive is used to propagate power to the next component or an electrical circuit is used to propagate a signal to the next component. In the Poseidon environment we need to propagate data. These data can be either control data (data provided from one application that describe how the next application must work) or raw data (data collected or created). As described above, the technology used to link together different objects is CORBA. However there is one more obstacle to overcome in order to link two applications. They have to agree on the type of data that they exchange. For example a routine that requires a string as input could be linked only to an application that provides a string as an output. In order to automate the process of data checking and verification for object linking we use another tool called metadata.

Metadata is information about data. It is used to describe a set of data in a standard way so as it is easy to store, identify, retrieve and use. Metadata has become a very important field in the area of information technology mainly due to the large amount of data that already exist and also the even larger amount of data that are expected to be created in the future. Metadata can also be used to conceptually describe in a standard way the behavior of software programs [4]. The Poseidon architecture is based on metadata standards for identifying and exchanging data as well as for storing an object's operational information [15].

The Poseidon User Interface

The heart of the Poseidon architecture is the Model Management System (MMS). The MMS consists of a Graphical User Interface (GUI) that allows a user to build an information workflow. An information workflow consists of distributed resources such as databases and applications that are linked together to exchange data. The parts of such a workflow are called nodes or objects. The MMS is responsible for:

- Creation of the GUI to be used by the user in order to build the workflow
- Validation of the workflow and
- Management of the execution of the workflow

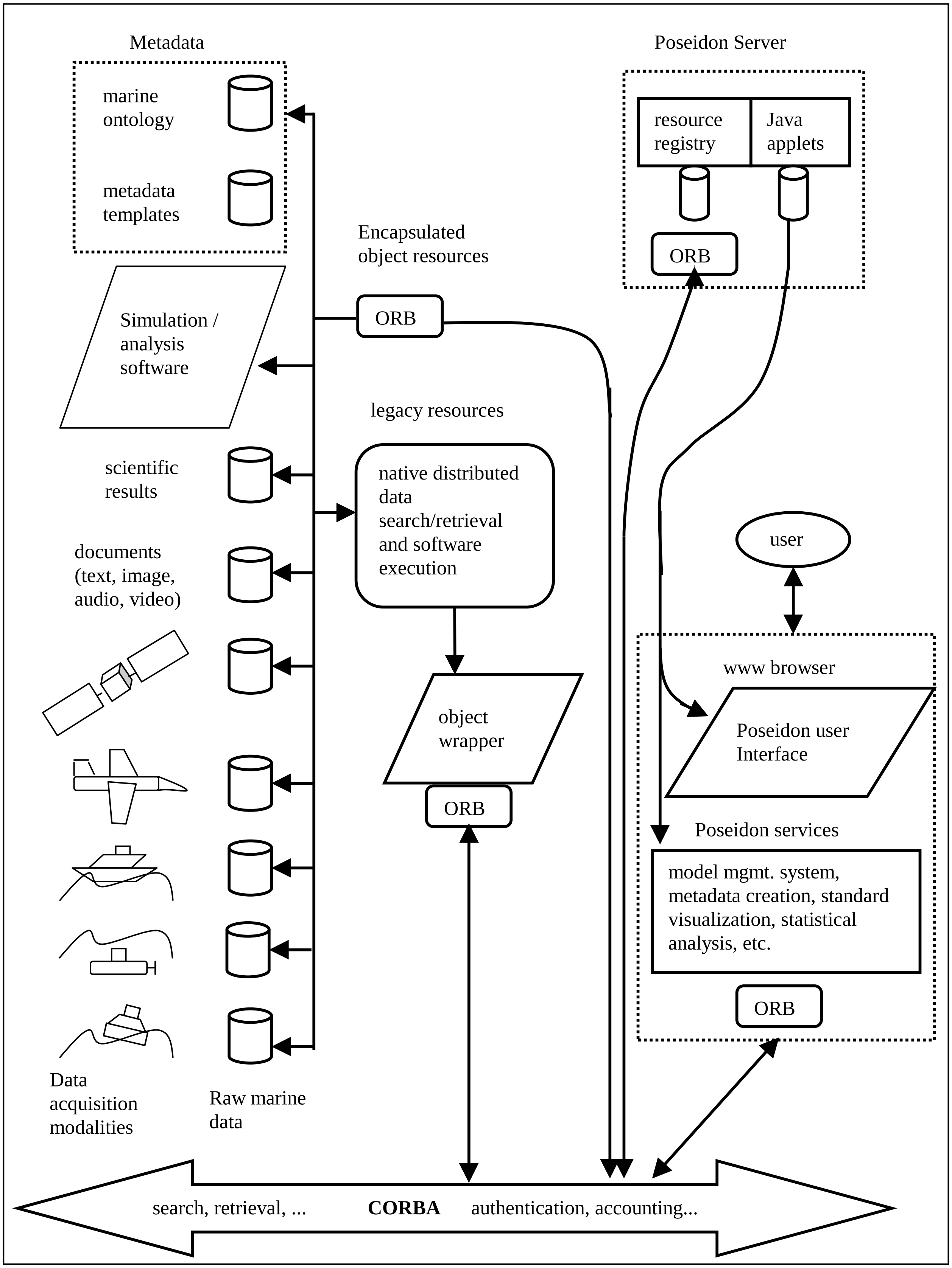

Figure 1 - The Poseidon Architecture

The MMS is not a stand-alone application. It requires the execution of other applications before it is able to operate. First, a Smart Agent[3], that will be responsible for object identification on the network. Second a GateKeeper[4] that will handle Client (applet) requests from different hosts[5] and finally the Poseidon Repository which is a central registration authority for storing information about the available objects. The Smart Agent and Gatekeeper are ready-built off-the-shelf applications that perform some important operations related to networking. The Poseidon Repository on the other hand is an application that was built to cover the needs of the Poseidon architecture. A web server also is required for serving the files of the Poseidon Applet. Such a server can be any of the many available in the market (Netscape Enterprise Server, Microsoft's Exchange Server, Apache Server, etc). The Messenger that appears in Figure 2 is responsible for the communications between the applet and the objects. It is another CORBA object that is been initiated by the applet.

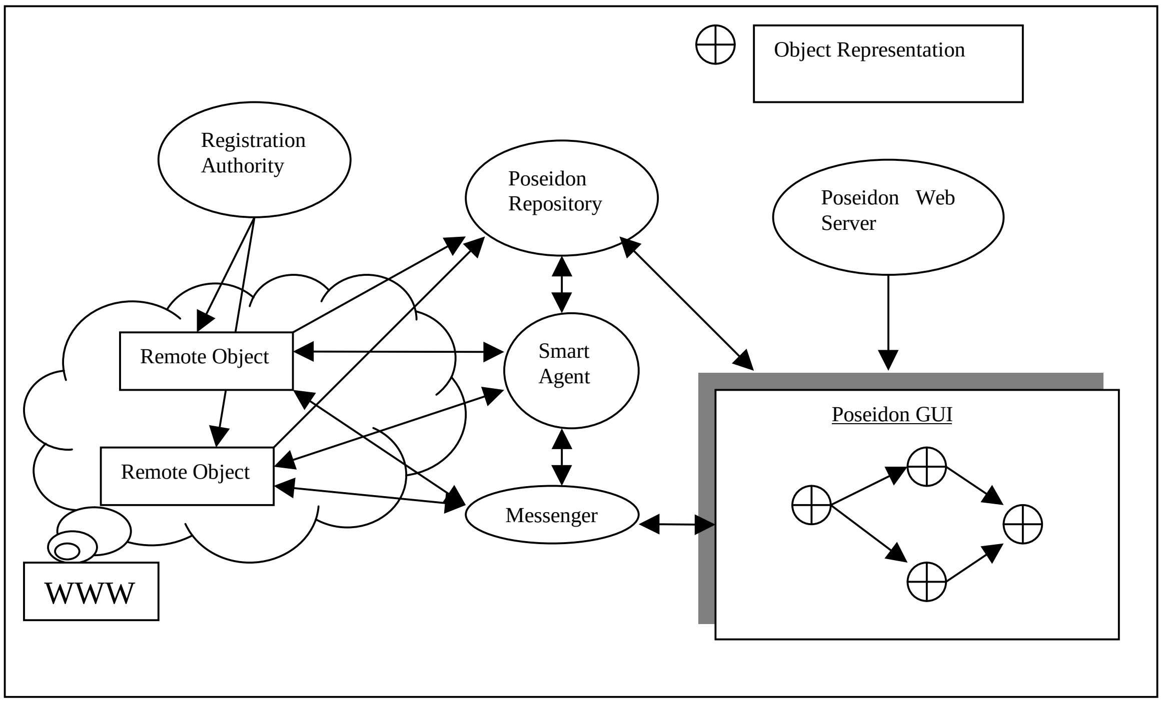

Figure 2 – Object Communications

[3] A Smart Agent is an application provided by Inprise's Visibroker development environment and is used to provide basic networking functionality.

[4] A GateKeeper is also an application provided by Inprise's Visibroker development environment

[5] A Java applet that is viewed on a browser has a security restriction that forbids it to communicate with servers on a different host. In order to overcome this problem (which is vital for distributed computing) Inprise's Gatekeeper creates a proxy server on the primary host from where requests can be redirected to other hosts.

The Poseidon MMS works as follows:

- The Poseidon Central Authority (PCA) starts up at least one Smart Agent and a Gatekeeper.
- The PCA starts up a Poseidon Repository (PR). The PR registers with the Smart Agent and declares its existence.

Since the PR is up and running, remote objects can be activated. Activation of a remote object is a three-step procedure:

I. Execution of the Generic Object Server (GOS). The local server is a generic server that enables the object to respond to requests using the CORBA standard. In this first step it is used just to enable communications with the PR.

II. Register the object with the PR, which will respond by issuing a Serial Number for the object.

III. The object starts a new instance of the GOS but using the serial number provided. This will uniquely identify the object on the network. The object's server then falls in an infinite loop process waiting for incoming calls.

- A User logs in the Poseidon website and downloads the Poseidon Applet.
- The applet loads from the Poseidon web server.
- During startup the applet contacts the PR and asks for a list of all the available objects.
- The PR replies by sending the list. The list contains basic information about the objects.
- The applet uses the list of objects to create a tree-like representation of the available objects.
- The user creates his workflow by adding and linking the objects that he needs into the design workspace.
- As soon as the workflow is ready, the user can ask for the objects to start operations.
- Before starting the objects (applications) however, the MMS will perform a number of checks. It will check to see if the connections made are valid and then it will communicate with each object to see if there are any special initialization conditions that have to be set. If everything is fine, it will start asking the objects to execute and provide results as their outputs. The rule for executing is "any object that has to provide data to another will be executed first".
- Eventually all the connected objects will be executed and the user will be able to see the results by means of some objects GUI (this is not totally necessary however; a user can ask for objects to send the results to a file or email without any intermediate GUI to display the results).

An Example of the Poseidon System Using the Harvard Ocean Prediction System (HOPS) and MIT's Ocean Acoustic Tomography Code.

The Poseidon system was tested by using existing applications developed by MIT and Harvard University. Harvard University's HOPS performs physical prediction of the ocean. HOPS requires a number of inputs to perform its calculations. For this example we transformed HOPS into a Poseidon object that required as input an array of data representing the sea floor elevation. A number of other input parameters were supplied as default values. These values were set as default only for reasons of simplicity; they could also be included as a required input. The HOPS code supplies as an output salinity and temperature data for the related ocean column in a standard format. HOPS is written in Fortran and was running on a Silicon Graphics computer running a UNIX operating system. A third application, called GRIDS, was used to supply a matrix of data representing the sea floor elevation. GRIDS was used by HOPS to receive a contour of the sea floor elevation. For this example GRIDS needed no inputs to operate; it was preprogrammed to provide the data for elevation of a specific region of the Massachusetts Bay. GRIDS was supplied by MIT and was running on a UNIX workstation. MIT's acoustic tomography code can produce acoustic tomography images of the sea column. It requires information about the salinity and the temperature of the water as well as the sea floor elevation map. MIT's code is written in Fortran but has been wrapped by a C++ layer and runs on a Silicon Graphics computer with a UNIX operating system.

The 2DColorContour code is an application that will create a graphical representation of a matrix of data. The representation is a color contour with capabilities of zoom-in, zoom-out and pan. The application is written in Java and runs on a Windows NT machine.

With the use of the Poseidon system we were able to start each of the mentioned applications as Poseidon objects. In the first edition of the Poseidon system we used a custom built applet to send parameters and initialize the remote applications (HOPS, GRIDS and OAG[6]). The three applications were running on different machines in MIT's Design Laboratory subnet. The applet was loaded using a common browser (both Microsoft Internet Explorer 4.0 and Netscape Navigator 4.0 were used) on a Windows NT machine.

Figure 3 describes the way these applications were connected together. The success of this operation was mainly based on the fact that the three applications (HOPS, Acoustics Code and Visualization) were previously unable to communicate. With the Poseidon system we were able to overcome the communication difficulties and link together programs that were built as stand-alone. In the next version of Poseidon, the linking of the various programs will become easier. In this next version, each individual program that is required to be introduced to the Poseidon system will be wrapped with a CORBA layer that will create a generic adapter. This generic adapter will allow for automating a large part of the procedure required in the first version of Poseidon.

---

[6] OAG is been used to identify MIT's Ocean Acoustic Tomography code.

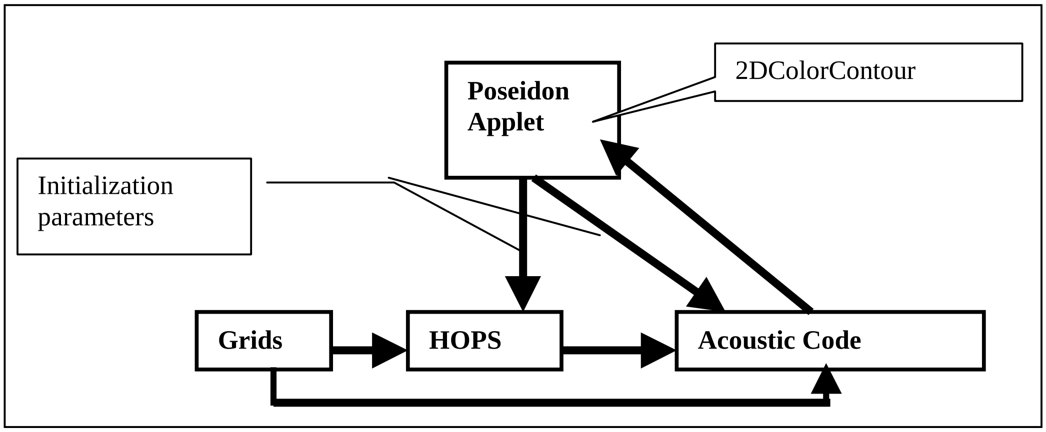

Figure 3 – Linking of HOPS with MIT's Acoustic Code with Poseidon

Distributed Design and Manufacturing

The Poseidon system will become more useful in areas where many different processes are required to work for one single final product. For example, in the case of a shipyard there is a large number of different applications involved in the production of the final product (ship). There are applications related to CAD, CAM, logistics, and personnel control, manufacturing control, quality assurance, security and much more.

Currently there are two different approaches, using existing technology, that can satisfy shipyards' computational demands:

1. A lot of different applications, each dedicated to a specific task or group of tasks. For example one software package (application) responsible for CAD, one for logistics etc. They are applications that are designed to handle specific tasks; they are very efficient and have a lot of capabilities.
2. A few applications with multiple capabilities. These usually come as a package of software applications, designed from the same vendor. They can be packages that include CAD, logistics, project management or manufacturing monitoring. As a rule these packages work well if considered as a group, but each individual element of the package is usually less capable than one of the first category. The second problem with these packages is that they are very difficult to scale. A better CAD program has to be supplied by the vendor that created the package. If the shipyard wanted to use a new CAD system from another vendor, they would have to find a way to integrate it with the rest of the system.

It has to be mentioned that some shipyards use a mixed version of the above. However, for each case there is a big problem to face. How to use very good software tools and put them all to interact with each other without compromising on performance. Also how will this system be able to integrate new applications in a seamless way and allow it to scale to face future demands. The solution for this problem can be found in the basic architecture of Poseidon. With the Poseidon system, existing (or new) applications can be wrapped with a layer of CORBA code transforming them into Poseidon objects. Then using the Poseidon system, link the applications in the way they should interact and improve the system for better performance.

An example could be the linking of applications such as logistics, Project Management (PM) and Manufacturing Control (MC). In a very simplistic approach we could say that project management is an object that takes information from the logistics department (material and parts availability, personnel availability, holidays, etc) and creates a plan for the production. On the other hand MC could take as input information from PM and set up the production process on the production floor. The MC can also be used as a feedback to the PM (e.g. inform about the progress of the job) and the PM as a feedback to the logistics (e.g. inform that more steel should be bought). In the case of a problem in the workflow, the system would be able to react rapidly. For example, if there is rain and the painting of the hull has to be delayed or completed in a closed hangar, the MC application would give feedback to the PM application. The PM would consult the logistics for availability of a free hangar space, or available material and personnel to move the painting inside, or start another job accordingly.

Since the whole design and production model is broken into parts and is distributed, there is no need to keep it local any more. The shipyard could sign a contract with a CAD office that would provide an object to do the CAD operations. The CAD office, being specialized in CAD would be able to produce much better results and more efficiently. The shipyard could also benefit much more by outsourcing the work to countries with low labor rates. In the same way all processes can be distributed. The shipyard could keep only the necessary (for privacy reasons) parts of the system in its local computers. In this way, it can reduce operating costs by maintaining fewer personnel for their information technology needs and get the same result at a lower cost. At the same time the shipyard hedges its risk of losing valuable and trained personnel[7].

Research Opportunities in Design and Manufacturing

In our opinion, future opportunities in research areas exist for the field of manufacturing technology in combination with information technology. In the following paragraphs we speculate about what the future will bring into information technology and how this will allow the optimization of the manufacturing process.

Metadata

We already mentioned metadata as being data about data. However metadata are more than that. Their importance in the field of information technology will become evident with the increased use of larger amount of data sets. Metadata will be a catalyst to help computers orient more efficiently in an area where data “entropy” is getting higher and higher. Metadata are somehow like meta-words

[7] According to federal government forecasts, the recruiting for IT professionals between 1994 and 2005 is estimated to reach 1 million people. However, the current supply of IT professionals from US universities and colleges is almost one third of the demand [8]. The high demand is likely to drive the salaries for IT professionals to higher values, and if companies employing specialized IT professionals failed to keep track with the new salaries, they face a high risk of losing valuable personnel.

or keywords used by search engines. They store a condensed amount of information that can describe a full set of data. They are formatted in a way that can be read both by computers and by humans with little effort. Metadata can be realized like another layer added upon the existing applications and data. Like CORBA, metadata will provide a layer of standardization that will allow applications to discover and identify each other. For example, a task taking place in a shipyard, such as the painting of a ship’s hull, could effectively be described using metadata. In such a case the metadata could include information that would be used by a computer to identify the properties of the task (e.g. the metadata file would contain a field describing the weather conditions required to do the job, what are the required equipment, what are the required skills of the personnel, etc). Another program (e.g. project management) that needs this information would browse through this metadata file and be able to discover if it is the type of data it requires. Metadata are also used to describe applications. Although it is much more difficult to be generic in describing applications, metadata have taken a step toward this. With applications, metadata are used to describe the application’s behavior and properties (e.g. structure of input/output, range of validity, accuracy performance requirements, memory requirements, etc. [15]).

However the big advantage of metadata is that they are getting standardized through international effort such those described in [9]. The manufacturing community over the last fifteen years, through the IGES and the STEP (STandard for the Exchange of Product data) efforts has effectively developed a metadata standard for all the data needed in the design, analysis and fabrication of manufactured products [5]. Furthermore this community has also developed a specialized non-procedural language called EXPRESS [6] for computer encoding of metadata and data information for these products. This allows for a standard way to describe manufacturing data. With the ability to have applications discovering each other through the use of metadata we can move to the next level of optimization, smart agent technologies:

Software Agent Technologies [11][12]

Smart agents are software applications used to automate procedures such as data mining and optimal selection. It is a technology that has advanced considerably during the last few years due to the large expansion of the Internet and electronic commerce. A large number of websites today use smart agents to offer services to their visitors. Smart agents are mostly used today to identify a predefined pattern from an information source (web site, databank) and collect data from it. They are used, for example, by news agencies to collect information from other online agencies. In this case, the pattern can be as simple as matching a set of keywords related to a specific story[8].

Other agents exist that perform various tasks (review published documents and prepare summaries, continuously search the web until they meet a number of criteria etc). Agents are becoming more complicated by adding elements of artificial intelligence. Today they are “smart” enough to handle complicated tasks such as automatic price negotiation[9] or product evaluation [13].

---

[8] For example, if a user asks for news about a specific stock, such as xyz, from Yahoo Finance (quote.yahoo.com) he will actually activate a smart agent that will start parsing other websites that are related to financial news and start looking for the keyword xyz. The agent will collect all the related documents and return to the user a list with all the articles.

[9] A relatively simple agent for automating the bidding process of the price to be paid for goods is currently available to the visitors of the well-known online auction service eBay (www.ebay.com).

The application of agents in a distributed computing environment can dramatically change the way manufacturing is done today. They can be used to discover organizations that offer products and services (preliminary design, classification societies, CAD, project management, parts manufacturing, delivery, etc) and interact with them in an efficient way.

Virtual Design and Manufacturing Marketplace

Using a distributed architecture, the owner of the system has the option to upgrade the individual parts of it without disturbing the operation of the others or without the need to make major changes to the rest. Expanding the system with a new piece of software (object) will be an easy process which will involve the wrapping (if not already done) of the new software with a layer that will be necessary to satisfy the architecture's requirements. The simulation of a large process will become easier because each individual part of the process will take care of simulating its own part of the total simulation. In this context, large processes can be simulated and optimized before the actual job been started. The following example provides a simple use of such a simulation. It takes advantage of the current and expected (in the future) capabilities of smart agents to perform real time negotiations and provide feedback to the simulation process.

The example describes the process of painting the hull of a ship[10]. In large shipyards, such processes are controlled from a department, usually the Project Management department, which interacts with the Logistics department for parts and personnel availability. Although the system is quite dynamic in its current form, changes in production plans are always undesirable and increase the total cost of the job. The purchase of parts or other consumables is always handled from a different department, usually the Logistics department. Communication between departments often causes bottlenecks in the workflow and can affect the shipyard's production.

In the future, a shipyard could have a single object (software application) that would handle each of the mentioned tasks. There would be an object for the logistics requirements and one for the project management. Placed on a network of distributed computers they will be able to communicate with each other. In such a scenario, the logistics object will hire a smart agent from the network to handle negotiations for buying the required paint [13]. The negotiator will contact other objects on the network that were placed by paint suppliers to negotiate the paint price and other details (time to delivery, specifications, etc). Real time feedback will be provided to the logistics object which in turn will inform the rest of the objects on the shipyard's local network (for example a settled deal for buying paint will inform the paying department to issue a purchase order and the management department to schedule the painting of the hull after the arrival of the paint).

---

[10] The example is restricted to a small shipyard operation just for reasons of simplicity. The procedure could easily be scaled to cover the whole shipyard operations.

References

[1] C. Watner, Businessmen versus Neocheaters - The Industrial Revolution, Black and White Publishing Company, Las Vegas, Nevada 1986.

[2] R. Orfali and D. Harkey, Client/Server Programming with Java and CORBA, second edition, Wiley Computer Publishing, 1998.

[3] G. Cornell and C. Hortsmann, Core Java, second edition, The JavaSoft Press, 1997.

[4] N. Patrikalakis, P. Fortier, Y. Ioannidis, C. Nikolaou, A. Robinson, J. Rossignac, A. Vinacua, S. Abrams, Distributed Information and Computation in Scientific and Engineering Environments, *D-Lib Magazine*, vol. 5, no. 4, April 1999, http://www.dlib.org/dlib/april99/04abrams.html.

[5] The STEP Project, http://www.nist.gov/sc4/www/stepdocs.htm.

[6] P.Wilson, STEP and EXPRESS, *NSF Invitational Workshop on Distributed Information, Computation, and Process Management for Scientific and Engineering Environments*, May 1998, Herndon, VI, http://deslab.mit.edu/DesignLab/dicpm/index.html.

[7] The Agent-mediated Electronic Commerce (AmEC) Initiative, http://ecommerce.media.mit.edu.

[8] G. Gilchrist, Growing Fruitful Information Technology Recruiters, *ComputerWorld*, December 15, 1997.

[9] FGDC/ISO Metadata Standard Harmonization, http://www.fgdc.gov/metadata/whatsnew/fgdciso.html.

[10] K. Mihanetzis, Towards a Distributed Information System for Coastal Zone Management, Engineer and Master Thesis, Massachusetts Institute of Technology, May 1999.

[11] S. Janson, Intelligent Software Agents, Swedish Institute of Computer Science, http://www.sics.se/ps/abc/survey.html#assistant.

[12] Software Agents Group, http://agents.www.media.mit.edu/groups/agents/, Massachusetts Institute of Technology Media Laboratory.

[13] P. Maes, R. Guttman and A. Moukas. "Agents that Buy and Sell: Transforming Commerce as we Know It.", *Communications of the ACM*, March 1999.

[14] M. Dertouzos, What Will Be: How the New World of Information Will Change Our Lives, HarperEdge, 1997.

[15] P. Wariyapola, N. M. Patrikalakis, S. L. Abrams, P. Elisseeff, A. R. Robinson, H. Schmidt, K. Streitlien, Ontology and Metadata Creation for the Poseidon Distributed Coastal Zone Management System, *IEEE Advances in Digital Libraries Conference*, ADL ’99, NY: IEEE, 1999.