

A Component Based Implementation of Agents and Brokers for Design Coordination¹²

Dr. Richard Weidner
Jet Propulsion Laboratory
MS 168-522
4800 Oak Grove Drive
Pasadena, CA 91109
818-354-2135
Richard.Weidner@jpl.nasa.gov

Abstract—

NASA's mission design coordination has been based on expert opinion of parametric data presented in Excel or PowerPoint. Common access is required to more powerful design tools supporting performance simulation and analysis. Components provide the means for in-expensively adding the desired functionality.

An information exchange was developed to provide the physical models required to perform performance analysis of mission designs. The exchange product is continuous polymorphic model data over finite time intervals. Using DCOM components, information brokers were developed to provide controlled access to the products. Each access is a persistent contextual transaction specific to a design team.

Information agent DLL's were developed to translate information requests, search among the brokers, retrieve the exchange products and ultimately generate high level model information such as state or attitude. An in-process dual component interface was developed to provide direct access from office productivity tools to the agent DLL's.

TABLE OF CONTENTS

1. Coordinated Design
2. Information Exchange
3. Intelligent Agents
4. Information Brokers
5. Information Agents
6. Information Agent Components
7. Conclusion

1. COORDINATED DESIGN

One of NASA's on-going goals is to facilitate coordination between the members of its mission teams. Coordination is readily justified and actively pursued during the course of gathering and analyzing science information. However, long before that point, the spacecraft, its payload, and the mission need to be designed to meet that science goal. Coordination during the early design activities is as important.

The Design Process

There is an on-going debate on what design entails. In one view, design is the geometric description of the layout of mechanical parts on a computer. Certainly Computer Aided Design (CAD) tool manufacturers encourage this view. Alternatively, a conceptual design is viewed as the paper and pencil drawing that precedes CAD refinement. A third and more expansive definition of design is as a process that includes analysis, visualization, and verification of the functional performance of the complete physical specification of a mission and its constituent systems. In this document design includes specification of the physical parts, modeling of their performance characteristics, simulation of the functional performance of the mission system and refinement of the design. The process begins on paper and culminates in generation of the physical specifications based on analysis of the system's ability to meet the mission science goals.

With the simpler view of design as mechanical layout, a limited number of people need to be involved in the generation of a blueprint. In past NASA missions, the design was captured on such paper products that were then mailed between subsystem teams. A system engineering team was charged with integrating the often-contradictory paper products. Without functional simulation nobody knew if the system met the ultimate goals. In particular, nobody could tell if the subsystems would even be compatible with each other.

Today CAD geometric information is available. More complex design tools provide subsystem performance analysis under hypothetical conditions. However, conflicts still occur because the system integration seldom includes performance analysis nor is the analysis performed with accurate models of the environment in which the spacecraft will be flown.

Based on experience with past missions, the more expansive definition of design as a process is being pursued in NASA. As a result, many more people are involved ranging from system engineers and subsystem manufacturers through the

¹ U.S. Government work not protected by U.S. copyright

² Updated October 20, 2000

science investigators themselves. Thus collaboration is critical to the design process. That collaboration entails the physical specification as well as the modeling and simulation of the function of that system in the intended environment.

Ripples

In 1994, NASA's Science Information Systems Program (SISP) began applying the wealth of physical model information available from past mission analysis efforts to simulate future systems as a part of the mission design process. Information technologies were evolved to facilitate access to the space science data sets.

In 1998, the Next Generation Infrastructure (NGI) activity of NASA's Cross Enterprise Technology Development Program was tasked to help provide the technologies required to support mission team coordination. Emphasis was placed on facilitating design coordination. Products were developed to support all levels of the communication infrastructure from low level network connectivity through secure transmissions and ultimately to high-level network objects.

In 1999, the Ripples task was initiated based on leveraged support from SISP and NGI. Ripples was proposed to develop infrastructure to support arrays of NT's connected as display arrays to support visualization of mission simulation products. The infrastructure would support an individual array as well as groups of these arrays connected remotely around the country. This paper reports on the products of the Ripples task.

User Resources

Design coordination of NASA missions consisted solely of teleconferencing just three short years ago. The model for the state of the art for current missions was taken from the architectures used by JPL's Project Design Center and from the ambitious design team for NASA's Intelligent Synthesis Environment. Cross enterprise coordination between all 10 NASA centers was required to form ISE. These two coordination activities was carried out across the country through a networked set of Microsoft's Office Productivity Tools. The tools were limited to the simultaneous presentation of viewgraphs on PC's connected through the network. Voice communication was supported by conference calls on telephone.

No other common set of computer capabilities was available across the participants. This fact along with past experience with the other NASA Missions indicates that the *de facto* standard for computer capability for mission teams is a networked personal computer. Those personal computers are most often laptop computers with limited screen space, memory, disk, and processor. The teams also have high-end computer resources. Many engineers have quite

sophisticated computers. However, they are not compatible with the network collaboration tools (PowerPoint). Thus, every one connected with NASA is currently using the ubiquitous PC as the minimal standard for collaboration.

The Ripples task was proposed to provide the high-level network objects required to work in this PC framework. More capable computers, faster IT environments, and thus more powerful engineering frameworks exist. However, the primary goal was infrastructure to connect the complete range of mission design teams. Thus, the products were designed to make optimal use of the abilities of the PC framework. Those abilities are focused around the requirements of information exchange between the members.

2: INFORMATION EXCHANGE

Coordination between a few members of a single company's design team is limited primarily by available technology. However, NASA's mission teams are composed of a diverse group of people from multiple companies or NASA centers each with their own interests. The members are geographically distributed and represent a range of backgrounds. Furthermore the endeavors are competed for large amounts of money and prestige. Thus, the environment for information exchange between the members is highly constrained.

The most obvious and highly visible requirement for information exchange is protection of the rights of the involved members. That implies a complex security arrangement. It also implies the exchange must be tailored for each individual. Some individuals must have access to some data but not to the information that could reflect on a competing design. On the other hand some individuals may sit simultaneously on competing teams and need clear indication of the information that applies to each in order to aid them in not sharing inadvertently.

The complexity of the security arrangement also lends itself to other facets of design. Individual designs and the information appropriate to that design process is contextual. That is, diverse bits of information accumulate that are appropriate to a single activity. The information systems that access that information can and should benefit from the contextual nature of the exchange.

The design activity also yields physical specifications that are time independent. That is the design is not changing in and of itself. In other words much of the design information is persistent. The behavior of the system may be time dependent but the specification does not change unless the members actively change them.

Many individuals from different companies, universities, NASA centers, and related institutions participate in the development and production of subsystems flown on NASA

missions. The design activity should recognize that information as well as mechanical parts might be provided through competitive access. Thus the activity must provide access to the various competitors just as in e-commerce.

3: INTELLIGENT AGENTS

The benefit of the World Wide Web has been to generate a huge new commerce based on information technologies. The Web is so large that companies have formed just to help find information. These helper companies have primarily provided access to their centralized search engines.

Experience with the search engines from the twenty or so helper companies is not great. A typical search may yield the desired content 30% of the time. The best success is with commercial vendors of staple products such as roses, plane tickets, and books. However, searches for more complex items fail consistently. For example a search for vendors of white "Elephant Ear" orchids would fail consistently though there are literally hundreds of vendors on the web that sell that color of phalaenopsis. However, with a bit of intelligence the engines can be used to search for orchids then phalaenopsis. Then just browsing for white varieties will yield the desired result.

The combination of search capability with some intelligence in software "agents" has been researched for years [1,2]. An intelligent software agent is a virtual person that represents the user in finding the information the user wants without requiring the user to personally wade through all of the web pages.

The intelligent agent is capable of understanding and interacting with its environment on the behalf of its user, of moving about the web, and of forming and executing rudimentary decisions. Faced with an open Web the intelligent agent shows great promise.

One subset of software agents is the information agent. The information agent delivers useful information to the user. The information agent may sit and winnow incoming information or go out on the web and search for, access, and return desired information.

This activity built an information agent to search for, access, and return information about spacecraft trajectory, spacecraft attitude, planet ephemerides, and planet kinematics to be used in the simulation phase of the mission design process. The information agent was built using Microsoft COM components in order to work across the PC platforms. The information agent was built with security, individual tailoring, context, and persistence. The agent would search specific sites with competitive brokers of mission information. The agents are software structures that provide access to the broker's products. Thus it is first with the brokers that the discussion continues.

4: INFORMATION BROKERS

Information Products

Simulated information predicts the ability of the mission to achieve its science goals. There is a great amount of information available to indicate how a mission will be performed. The quantity and diversity of the information led to the growth of standards (with emphasis on the plural of standard.) There are more than nine different standard data forms for specifying just the trajectory of a spacecraft.

The different forms vary not just in the format of the data but also in the content. Including the position of the relative reference body, trajectory data uses specifications from NORAD's modified Kepler equation coefficients to Lagrange polynomial coefficients.

However, any single source will generate just one of the data forms in a typical file. The user then requires knowledge of the format of the file as well as the code to translate the content into state vectors.

The combination of data sets from multiple sources is often required to form the relative state vectors from a space vehicle to a specific target (planet.) The state vectors are often generated with respect to one reference but are desired with respect to another reference. For example, state vectors are often propagated with respect to the Solar Barycenter during cruise. The position relative to a target then requires translating the vector using the ephemerides of the target. Multiple sources are used for the cruise data and the target ephemerides. Any single program needs the ability to recognize any of the different forms and extract the desired product. Network objects can help provide this polymorphic functionality.

DCOM Brokers

The Object Request Broker (ORB) called the Service Control Manager (SCM) in DCOM spawns low level processes and thereby instantiates components [3]. A higher-level information object was developed from an out of process component instantiated by DCOM. This higher-level object serves the function of instantiating specific network objects thus it is also called an information broker.

The function of the information broker is loosely modeled after the JINI network service [4] though it is implemented in C++ and DCOM. The JINI service has many superior attributes for robustness and mobility. However, the productivity tools currently used by NASA in design coordination require a DCOM implementation instead of the JAVA alternate architecture.

An information broker is simply a software program that handles access requests to the one or more data sets generated by that organization or individual. Therefore it is a server. But, a broker also logs requests and develops a user

profile in order to help meet specific user needs. A typical information broker for spacecraft state is shown in figure 1.

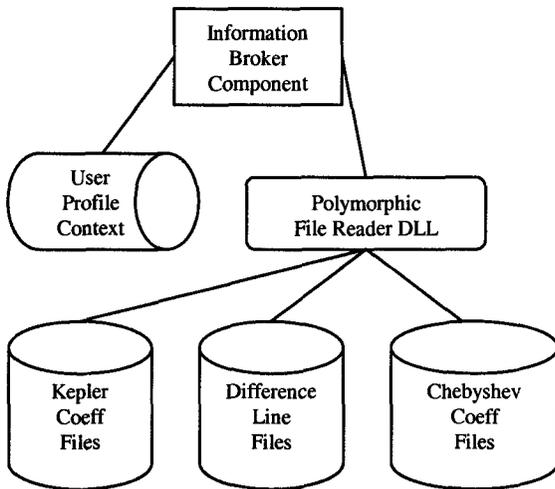


Figure 1: A Spacecraft State Broker

The broker handles identification and security appropriate to the user profile. Thus, the broker is also contextual and persistent by handling multiple requests from the same user through time. The broker can provide a limited secure environment on its local machine in order to handle ancillary data access and decision making to assure the user that he is getting the desired information.

Along with the broker component, the data supplier can provide in-process components or dynamic link libraries to read and interpret the raw products. Thus, the user need not know the actual data format or the interpretation method. The component handles both of those transparently and just provides the end product information through the registered interface.

A Spacecraft State Broker

For example, Spacecraft trajectory and planetary ephemerides represent the continuous state of the bodies using discrete datum. Interpolators and integrators are used to generate actual state vectors from the datum. Thus, the data are usually aggregated in segments representing a period of time. The broker's product is that segment of data not the individual datum. Thus the interpretation of the source format is transparent but the interpretation of the segment is not.

The information broker was formed as an out-of-process component accessed through Microsoft's DCOM [2]. The information broker hosts the user's search requests and forms the result product segment(s). It then returns the segments through a component request using a custom marshaler. [5]

The information broker provides two functional areas in the interface. One area is used to set up the context of an activity. It registers the specific data sets required by that context. It also handles security for that set of data. The second area is used to actually form the queries and return the product segments. The context functional area is persistent. The query/access functional area is logged but has no persistent effect on future queries/accesses.

A dynamic linked library was built to handle local file access to the mission data sets. The broker component then used that DLL to access information from the data sets. However, that is completely hidden by the component interface. Thus, another broker component could use the same interface but a different source data set and access DLL.

Once the segments are copied across the network they are cached in segment files. Thus, the user need not ask the broker for the same data twice. Future queries may then check the cache first before asking the broker component. The same access DLL handled the original files and the cache segment files. In this way a broker may request data from another broker and cache the segments locally to meet user needs. The same access DLL will transparently handle both data sources for the broker.

An information broker handles only one type of information. Thus, the interface is specific to the data. A segment of trajectory information has a different interface than attitude information that is interpreted and handled completely differently. Furthermore, attitude information originates at completely different organizations though it may wind up in the same archive.

The broker grants access and logs activity. It may then be used to provide accounting information for any transaction charges due during the product exchange. Currently, most information is exchanged but the cost is hidden in other very limited budgets. The growth of brokers and transaction commerce could free up the hidden charges and allow for more comprehensive support.

The information agents interpret the segments using the dynamically linked libraries (accessed through in-process components) provided by the source provider. Thus data segment and executable network objects are exchanged. The object-oriented paradigm is obeyed with a network object containing the data and the executable to access it through the registered interface.

5: INFORMATION AGENTS

Information Agents represent the user in searching for specific information among the products available from a set of information brokers. Figure 2: illustrates an agent/broker exchange using information broker components implemented in DCOM.

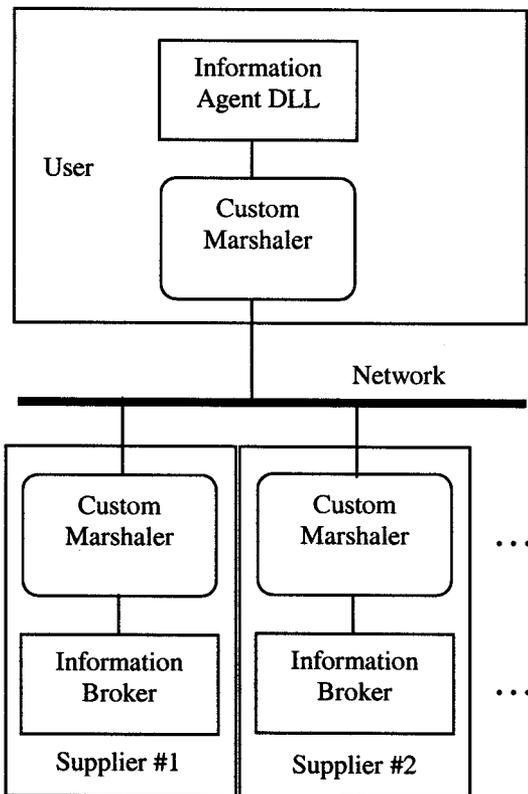


Figure 2: Agent - Broker Exchange

For this discussion a vehicle state vector is an n-tuple that expresses the geometric (location, velocity, and/or acceleration) relationship between one body and another in a specific reference frame at a fixed time. An attitude quaternion expresses the orientation of the vehicle's reference frame with respect to a standard reference frame. During performance simulations a designer wishes to have state vectors and attitude quaternions for a series of discrete times in order to analyze the feasibility of mission goals.

An information agent has been built to collect and return desired state vectors and attitude quaternions. The user specifies the desired attributes of the vectors including vehicle id, reference center, reference frame and time. Similarly the user specifies the target body, reference frame and time for the attitude quaternions. The agent performs the search and returns the desired information.

An Information Agent DLL

The fastest interface available to a custom processing tool is through Dynamic Link Libraries. Thus the primary interface to the information agent is through an Application Programmer's Interface (API) to a set of individualized DLL's. For example an information agent DLL is written specifically to access state vectors. Another information agent DLL is written specifically to access attitude information. Other agents return body kinematics

information and so on. Figure 3: illustrates the information agent DLL architecture.

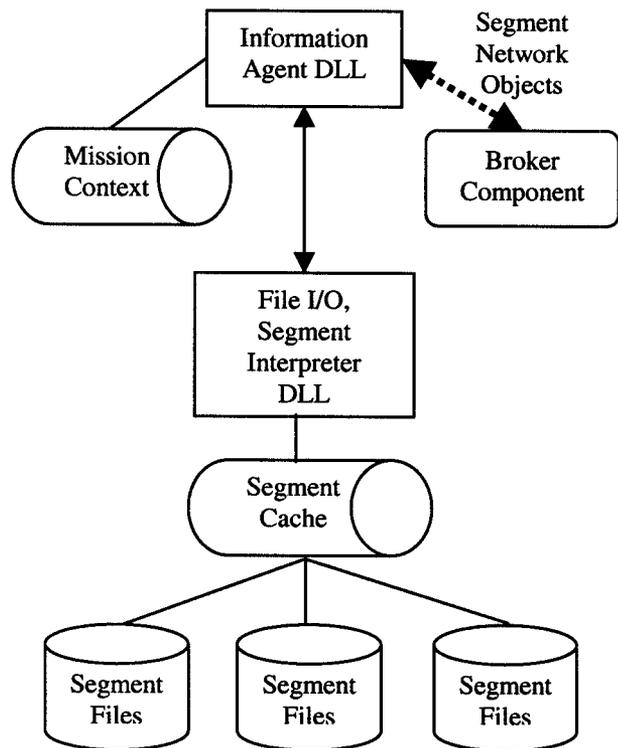


Figure 3: Information Agent DLL Architecture

The agents contain the interface to the broker components. Thus the more demanding user never needs deal with components directly. Similarly the supplier's custom DLL's that interpret a broker's specific data type and format is accessed through the agent DLL and thus is hidden from the user.

Using a standard interface and polymorphic coding practice, various broker's DLL's may be accessed without modifying the original custom processing tool. Thus, multiple individual suppliers may used to provide the desired information.

For known suppliers, a standard file access library can be used. For example, a polymorphic implementation that reads and interprets the nine most common segments was formed and dynamically linked into the agent DLL. The agent may then search the network among the available brokers and piece together the segments required for the desired information. The segments are cached using the access library. The cache segments are interpreted (i.e. interpolated or integrated) using the same access library. The cache is not destroyed between calls and thus the information is persistent just as though it was stored in a persistent memory structure.

The identification of source brokers and the cache files containing segments returned from those brokers are logged.

The next time the user wants information the cache is searched first to save network access delays. This modality is frequently used since the users most often want geometric state that is identical but only slightly later in time. Thus, the segment access mechanism saves the most frequent network access repetitions. The repeat accesses need not be made within the same process and thus may be delayed as long as the user wishes.

The access log is saved under a user-specified identification number. Multiple identification number may be used for different groups of accesses and their associated cache files. Thus, the agents are contextual. A special context (identification number zero) is used to indicate that no cache is saved or searched.

There are two types of context. The first context describes the individualized agent that contains brokers and the cached files associated with them. The second context is the context of the broker component. The later context is used to describe the data sets appropriate to one or more users. The second context identification number is often provided through the broker component interface and is thus also logged in the first context along with the other attributes associated with that broker.

The more demanding user may use the agent DLL directly and thus obtain the fastest access available on a Microsoft operating system. However, other simpler users need to access the information in more standard frameworks. Thus an agent component is also available.

6: INFORMATION AGENT COMPONENTS

An information agent component's main function is to provide a standard interface between the agent DLL and the standard frameworks such as Microsoft's Excel. The component uses a dual interface based on the built in marshaler for in-process components. The standard frameworks use Microsoft's Visual Basic for Applications (VBA). Thus the interface handles only the standard data types available through VBA. These data types also include VBA specific multi-byte character set strings and variants (unions). [6]

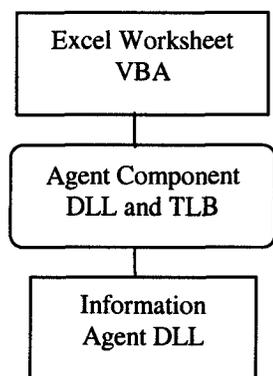


Figure 4: Information Agent Component

Figure 4: illustrates an information agent component. The information agent component is just a simplified front end to the information agent libraries. Another way of describing it is as simply a standard interface or glue. Since the agent DLL provides the functionality, the agent component can be a simple in-process component. An alternate interface may be created using Microsoft's Active Template Library (ATL) [7]. The ATL wizard is easily used. However Microsoft's products have limited support for templates and should be used with caution. This activity therefore avoided ATL.

Standard frameworks such as the interpretive office productivity tools can do very few of the functions available through a complex information agent API. The requests are typically simple and the tools are relatively slow since the information is presented directly to the user. Thus a greatly simplified agent component was constructed based on that limited functionality.

Individual components were built to provide transformations between Gregorian calendar, barycenter dynamical time and spacecraft clock (on-board counter). Components were also built to access spacecraft state, spacecraft attitude, and planet kinematics models. These later components provided access to information agent DLLs and information broker components.

7: CONCLUSION

Based on the limited resources available across a mission design team, a data exchange was implemented in DCOM using components. The exchange provides access to distributed data sets using the e-commerce model used in the World Wide Web.

Supply side servers were built using a transaction service known as an information broker. The broker maintained security and persistent, contextual user profiles to meet individual needs. The broker was implemented as a remote out of process executable component implemented in DCOM.

The exchange products were low-level spacecraft trajectory and attitude, and target ephemerides data segments. The supplier provides file reader and interpreter (integrator) dynamic link libraries to the client. Thus the formalism is of network segment objects.

On the client side, information agents were implemented to search for the appropriate segments from among the available information brokers. Polymorphic DLL's were implemented to extract the appropriate state vectors and attitude quaternions from the multiple low-level segments.

Higher-level agent components were implemented as DLL's with type libraries. The components are accessible from

standard office productivity tools such as Excel VBA. Thus, every member of the design team can access the products.

Relatively little can be done with the information strictly within the office productivity tools. However, those tools provide much of the functionality used by the science and spacecraft design teams. Decisions based on target range, target image size, solar phase angle, visibility and such rudimentary calculations are supported in the limited tools.

The office productivity tools also provide a standard man/machine interface that is familiar to the team members. Just as the components can be used to return simple information, other components can extract parametric information and drive background simulations. Thus the information agents return information that then allows the design team members to simply vary design parameters and further drive simulations and analysis

Future efforts remain to develop the interface to drive remote simulations from Excel. Similarly, a network monitor is planned to keep track of the available brokers in a similar architecture to that available through JINI.

ACKNOWLEDGEMENTS

This work was performed at the Jet Propulsion Laboratory of the California Institute of Technology under one or more contracts from NASA. The work was supported by the Science Application Information Technology element of the Science Information Systems Program of NASA's Space Science Enterprise and the Next Generation Infrastructure element of NASA's Cross Enterprise Technology Program.

REFERENCES

- [1] Joseph P. Bigus and Jennifer Bigus, Constructing Intelligent Agents with Java, New York: John Wiley & Sons, Inc., 1998.
- [2] David Pallmann, Programming Bots, Spiders, and Intelligent Agents in Microsoft Visual C++, Redmond, Washington: Microsoft Press, 1999.
- [3] Guy Eddon and Henry Eddon, Inside Distributed COM, Redmond Washington: Microsoft Press, 1998.
- [4] W. Keith Edwards, Core JINI, New Jersey, Prentice Hall PTR, 1999.
- [5] Al Major, COM IDL & Interface Design, Birmingham, UK: Wrox Press Ltd, 1999.
- [6] David Boctor, Microsoft Office 2000 Visual Basic for Applications Fundamentals, Redmond, Washington: Microsoft Press, 1999.
- [7] George Shepherd and Brad King, Inside ATL, Redmond, Washington: Microsoft Press, 1999.

Dr. Richard Weidner is the technical group supervisor of the Mission Simulation and Instrument Modeling Group in the Observation Systems Division at the Jet Propulsion Laboratory. He has developed information technology for NASA's Planetary Science Information Systems since 1981. He has developed IT for NASA's Navigation Ancillary Information Facility, Planetary Data System, and SETI Whole Sky Survey. He has developed information technology products to support planetary mission simulation including the Voyager, Galileo, Cassini, Stardust, and Deep Space 1 missions He developed image sequencing and presentation tools used to support Mars Pathfinder. He has a PHDEE from Oklahoma State University.

