JOURNAL OF AIRCRAFT
Vol. 38, No. 2, March–April 2001

Multiagent Design Architecture for Intelligent Synthesis Environment

A. Deshmukh* and T. Middelkoop†
University of Massachusetts, Amherst, Massachusetts 01003
A. Krothapalli,‡ W. Shields,§ and N. Chandra¶
Florida State University, Tallahassee, Florida 32310
and
C. A. Smith**
NASA Marshall Space Flight Center, Huntsville, Alabama 35812

The intelligent synthesis environment (ISE) represents the integration of a broad range of high-fidelity knowledge at the design stage to facilitate the product realization process. ISE has the potential of reducing the product realization time by several orders of magnitude. The key characteristics of ISE are integrated evaluation of product life-cycle stages in a virtual environment, coordination of asynchronous design activities, design configuration selection based on multiple design criteria, and interoperability among heterogeneous software and hardware environments to accomplish an effective collaborative computational system. Multiagent design architecture (MADA) is designed to provide a flexible integration framework for ISE. This architecture provides seamless integration of product realization activities across heterogeneous machines, computing platforms, programming languages, data, and process representations using distributed intelligent agents. An agent in the context of MADA is an autonomous computational entity that is capable of migrating across computing environments asynchronously. These agents have intelligence in the form of individual goals, beliefs, learning mechanisms and interact cooperatively to accomplish overall product design objectives. An application of the MADA information framework for the design of aerospace components is presented. Specifically, the use of integrated design and analysis tools for creating a high-speed civil transport exhaust nozzle is demonstrated.

Introduction

IN an increasingly global market, the need for more efficient product realization methodologies is driven by the requirement to introduce new, cost-effective products quickly into the market. This is especially true in the aerospace industry, where shrinking defense budgets and international competition are forcing manufacturers to significantly reduce the concept-to-delivery time for products. To achieve these objectives, intelligent synthesis environments, which incorporate design tools, high-fidelity analysis, manufacturing, and cost information, have been proposed for reducing product realization time. The term product realization, in the context of this discussion, is defined as the description, design, manufacture, testing, and life-cycle support of a product.

The general concept of an integrated environment for product realization has evolved steadily over several decades. In the past, product realization occurred in distinct phases, such as requirements definition, design, manufacture, and product support. Each phase involved activities by distinct engineering disciplines, such as fluid dynamics, material science, structural analysis, machine design, electrical design, and process design. Each phase and engineering discipline was characterized by different timescales, ontologies, data formats, analysis techniques, and communication media. The differences among project phases and engineering disciplines resulted in inefficiencies in the product realization process, the fundamental cause of the resulting inefficiency in the process being the lack of communication and coordination among various product realization phases and engineering disciplines.

In recent years, increasing attention has been paid to a total system approach in the management and engineering of large and complex development programs. The techniques of system program management introduced the organizational concept of a dedicated, cross-disciplinary matrix of people and resources, drawn from the traditional engineering and support disciplines. As the systems approach to the engineering of complex products evolved, the disconnects between the project phases, most notably between design and manufacturing, proved to be a major impediment to efficient product development. This observation led to the development of project management and systems engineering approaches that have been referred to as life-cycle development, total package procurement, design for manufacture, and concurrent engineering. These approaches enhance the probability of creating high-quality products, without cycling through the product redesign iterations. The use of multidisciplinary knowledge at the product development stage, early in the product life cycle, presents opportunities for significantly impacting the product cost and performance. When design errors are caught early, downstream reengineering costs can be greatly reduced. Proper communication and technology transfer can also result in cost saving innovations or infusion of new technology in the designs.

Although these practices have improved the product design process, the communication and coordination among project phases and engineering disciplines still needs considerable improvement. In response to recent business pressures on the product realization process, a refinement and extension to the systems approach has been proposed, referred to by several names, such as virtual product development, integrated design environment, or intelligent synthesis environment. The fundamental goal of the new approach is to improve the design of complex systems by improving communication and coordination among the various project phases during early stages of the product life cycle. Implementation of this new

Received 4 August 1998; revision received 15 October 2000; accepted for publication 11 November 2000.

*Assistant Professor of Mechanical and Industrial Engineering.
†Graduate Research Associate, Department of Mechanical and Industrial Engineering.
‡Don Fuqua Chair Professor and Chairman of Mechanical Engineering. Associate Fellow AIAA.
§Adjunct Professor of Mechanical Engineering.
¶Professor of Mechanical Engineering.
**Director, Systems Engineering and Requirements Definition, Second Generation Reusable Launch Vehicle Program. Associate Fellow AIAA.

approach involves several elements: 1) immersive virtual engineering environments, wherein the entire product team has access to all of the product and process information needed for each team member to complete their task; 2) common user interfaces, such as a web browser interface, to coordinate access to project information gathered from diverse sources; 3) common data formats, which permit the integration and translation of different information from different disciplines; and 4) integrated analytical and process management tools for collaborative engineering.

Several prototypes of integrated design environment are being developed in the aerospace industry. Lockheed Martin Tactical Aircraft Systems has developed a comprehensive integrated design environment to support their participation in the Air Force Joint Strike Fighter program.[1] The Lockheed Martin Virtual Product Development Initiative has developed a virtual modeling, simulation, and design environment that has the goal of improving competitiveness in the initial phases of the affordability-driven Department of Defense acquisition process. This initiative was directed toward improved functioning of a multidisciplinary product team. Thus, a major component of this system is a central database, a product data manager that can be accessed through a common browser interface. This architecture also includes visualization tools and access to enterprise knowledge available at other company locations, such as company best practices in design engineering.

Boeing Space and Communications Division developed an integrated design infrastructure to provide access to the engineering tools and processes that support their integrated product teams.[2] Their motivation was to improve design integration during the early conceptual phase, when sensitivity to new technology and design changes is the greatest. In the Boeing system, the CAD tool provides a master solid model database, which is accessed by all other elements of the system. The component geometry data is expressed in the International Standards Organization (ISO) standard ISO 10303, commonly known as the Standard for Exchange of Product (STEP) model.[3] A virtual reality tool provides the capability of accessing the geometry database for fly-through inspections to check clearances, form, fit, and function. This system also provides a product data management system, with access to legacy databases such as material and parts libraries.

Boeing Information, Space, and Defense Company, through its participation in the Defense Advanced Research Projects Agency Rapid Design Exploration and Optimization program, has developed an integrated design environment that emphasizes standardization of database and data exchange formats. The objective of this program was to demonstrate integration of diverse analytical tools, including CAD, fluid mechanics, and structural analysis, as well as nonanalytical information, such as requirements traceability and manufacturability, into a standards-compliant, application-neutral, object-oriented design environment. The operation of the system involves translation of the application specific outputs of various tools into a neutral, object-oriented data format, called units of functionality, that resides in a common database.

The National Institute of Standards and Technology (NIST) Design Repository project focuses on developing a modeling framework for the representation of artifacts in a design database.[4,5] Their emphasis on an object-oriented, application-neutral representation scheme for product data in an integrated design system is consistent with the central importance of the product and process information base. The NIST project provides an object-oriented modeling language that transcends the traditional geometric modeling format to include form, behavior, and function. Their work acknowledges that the standard for geometry representation, STEP Application Protocol (AP203), is much further advanced than standards for nongeometric information. The NIST development also includes a suite of tools for implementing commercial object-oriented databases, an information browser for a user interface, and visualization capability.

In response to the changing needs of the aerospace product development community, NASA has recently launched the intelligent synthesis environment (ISE) program to foster research in the virtual product development area.[6] This program has identified five large-scale applications, including 1) the International Space Station, 2) the space shuttle, 3) a reusable space transportation system, 4) an advanced Earth observation system, and 5) the integrated exploration of science applications, as potential domains for the tools developed in the ISE program. The primary emphasis of ISE is on the distributed collaborative nature of large-scale designs. This requirement forces the ISE researchers to address the issues of geographical separation, heterogeneous computing platforms, and diverse design goals.

However, in all of the systems discussed in this section, as the tools in the integrated design environment increase in both numbers and sophistication, the ability to use these tools effectively becomes increasingly difficult.[7] To use high-fidelity analysis and manufacturing tools at the design stage, designers need to communicate with domain experts and their analysis tools, which may be geographically dispersed over heterogeneous computing platforms. The designers also need to be able to locate the right tools dynamically during a design iteration because different tools may be appropriate at different stages of design, or a tool may be unavailable due to machine failure or excessive load. Furthermore, in the case of large-scale systems, it is essential to be able to decompose the problem into smaller subproblems that are tractable.

The ability to effectively communicate between diverse tools is strewn with technical difficulties and ontological issues that impede the integration process. A major difficulty is the number of proprietary software systems and data formats. With the advent of the world wide web and related technologies, the barriers to data transfer have been significantly reduced. Data interoperability, however, still remains a major roadblock in any large-scale design activity. The ability to transfer information, not just raw data, from one analysis domain to another is complicated by differences in semantics and timescales. From the design perspective, we need to ensure semantic interoperability, which includes design rationale, ontological issues, and other metadata. These data are either not captured by present systems or are often lost in data translation. Network collaboration is an important part of the integrated design and analysis process. Currently, commercial software products are available that overcome many technical barriers of multilocation collaborative design. However, these tools provide only the basic communication channels and point-to-point connectivity between specific applications, which are not sufficient to meet the complex requirements of an integrated design architecture. As the product design process becomes more decentralized and there is proliferation of new software and hardware systems, the need for a flexible solution, capable of handling the variety and changes, for integration is acute.

This paper presents a multiagent design architecture (MADA) model that provides an information framework for an ISE by enlisting the use of mobile software agents. The paper is organized as follows: The next section introduces agent-based systems and provides a rationale for using agent-based architecture for design systems. The following section describes the MADA system architecture in detail by describing the functionality of each agent in MADA and giving details of the software implementation. The next section presents an implementation of the MADA model for a high-speed civil transport (HSCT) aircraft exhaust nozzle. Each of the domain specific tools and their integration in the multiagent architecture are described. We also present computational results for the HSCT nozzle design under different operating conditions. The paper concludes with a discussion of plans for future enhancements to MADA.

MADA

Agent-based architectures are being increasingly used to control and coordinate complex systems. Several applications of agent architectures have been reported in the areas of robotics, manufacturing, and distributed computations.[8–10] In our context, agents are defined as autonomous entities, either hardware or software, that perform tasks to achieve an overall system objective.[11,12] These agents possess intelligence in the form of individual goals, beliefs, and/or learning mechanisms. Intelligence in an agent permits it to be proactive in performing tasks and communicating with other agents instead of being purely reactive to the system state. The autonomous

characteristic of an agent refers to the self-contained computational capability to carry out tasks without external support. Thus, the design of an individual agent can be changed without propagating the changes to other agents. The asynchronous nature of the agents allows them to run simultaneously with other agents in their own environment or context. The agents may be mobile, allowing them to migrate to different locations to perform the tasks. Mobile agents can move from one context or operating environment to another, performing their tasks locally while maintaining their identity. Mobile agents provide specific advantages over traditional software by providing the ability to query intelligently the information directly at the data storage location, thus allowing high bandwidth communication between the data source and the agent. This dramatically reduces network load and query times. Agent mobility also allows distribution of computing requirements by offloading tasks to other machines. The distributed nature of multiagent systems provides advantages, both in terms of performance and fault tolerance, over traditional architectures. Mobility also allows the agents to gather information and provide services across inter-, intra-, and extranets spanning several organizations.

Rationale for Agents in Design Environments

In the design context, agents provide a means for developing a flexible, intelligent architecture for integrating design, analysis, and manufacturing tools. Agents provide customizable interfaces to large and diverse data sources. They also provide ways to standardize data access by placing a software layer between the user and the data. The need for proprietary interfaces to information sources can thus be avoided by abstracting the user from the data. By the use of network-based agents, the information residing on a remote host or a server can be accessed by multiple users.

The main idea behind the MADA is to provide an automated and distributed approach to solving a wide range of parametric design problems. The focus of this research is on the distributed nature of the solution generation, that is, to develop a system that not only decomposes the problem into distributed tasks, but also searches the design space in a distributed manner by taking advantage of the underlying structure of the decomposed problem. By the placement of the focus on distributing both parts of the design problem, the design tasks and the solution search, the resulting system should be both robust and scalable to large design projects, such as the space station or the next generation space shuttle.

MADA provides a research platform to study novel approaches for distributed product realization. The overall architecture can be divided into three broad stages. The first stage involves decomposition of a design into a parametric form, whereby a design instance is completely defined by a finite set of parameters. The second task focuses on the integration of the design, analysis, and manufacturing tools to provide an automated system for repetitive analysis of design alternatives. Finally, the third task focuses on the development and analysis of distributed search methodologies to evaluate a large number of design alternatives.

We now present an overview of the three stages of the MADA environment, using HSCT nozzle design as an example. The HSCT nozzle design scenario requires multidisciplinary analysis, such as aerodynamics analysis, structural analysis, and material selection. The design objective is to select a nozzle design that minimizes the cost while meeting the performance criteria, which are related to the Mach number at the exit, maximum allowable stress, and limits on the dimensions of the nozzle. The design variables in this case are the shape of the nozzle (we consider axisymmetric and rectangular geometries), geometric dimensions, and material.

The overall HSCT nozzle design task is initiated in MADA by representing the design as a set of parameters. These parameters represent the constituent dimensions, tolerances, materials, and other information required to define the nozzle completely. Information related to the performance of the design is also expressed in this form. In the HSCT nozzle scenario, parameters that define a nozzle design alternative are the type of geometry, length, thickness, outlet area, and nozzle material, along with performance information such as exit temperature, weight, maximum stress, and cost. The relationships between design parameters are derived from parameter maps, which show the sequencing of design and analysis tasks for a given domain. These maps also encode the conversion of one parameter to another by analysis tools and are used extensively throughout the system. The parameter map for the HSCT nozzle design is shown in Fig. 1. Note that the finite element analysis and the manufacturing analysis tools can be executed in parallel once the information is available in the product data repository. The details of the data flow in a section of the parameter map are shown in Fig. 2.

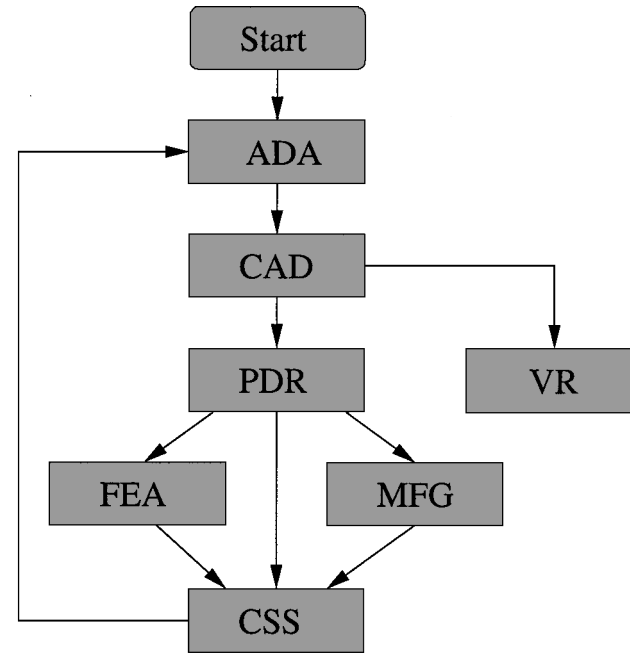


Fig. 1 Parameter map for HSCT nozzle design.

The second phase of the design process involves integration of all of the tools required to traverse completely the parameter map. This phase has three elements, incorporating design and analysis tools in an information architecture, coordinating the design tasks specified in the parameter map, and transferring information between tools. MADA uses different types of agents to accomplish these tasks. There are three broad categories of agents, as shown in Fig. 3: tool agents that interface with an analysis tool at a specific location, such as a tool agent that interacts with a CAD tool; facilitator agents that are responsible for sequencing the different tasks in the parameter map, hence, each facilitator agent coordinates the design process of a specific object; and worker agents, which are responsible for transferring information between different tool agents and the facilitator agents. The design of each of these agents will be discussed in detail in the following section.

The third phase of the HSCT nozzle design process involves searching through the feasible design space to identify a good alternative. MADA uses a constraint satisfaction and search tool to identify feasible design alternatives. Multiple design instances can be evaluated in parallel by creating independent facilitator agents that manage parameter maps for each alternative.

The key features of MADA that differentiate it from other integrated design systems are the ability to decompose the overall design problem into ordered sequence of tasks, the capability of adding new tools in the system without affecting the entire system, the flexibility of using new resources introduced in the agent environment, asynchronous operation of different tools, and the distributed problem solving capability of the design search process.

The computational environment in MADA differs from other integrated design systems in the methods used to integrate design tools. The integration and automation is accomplished by using a

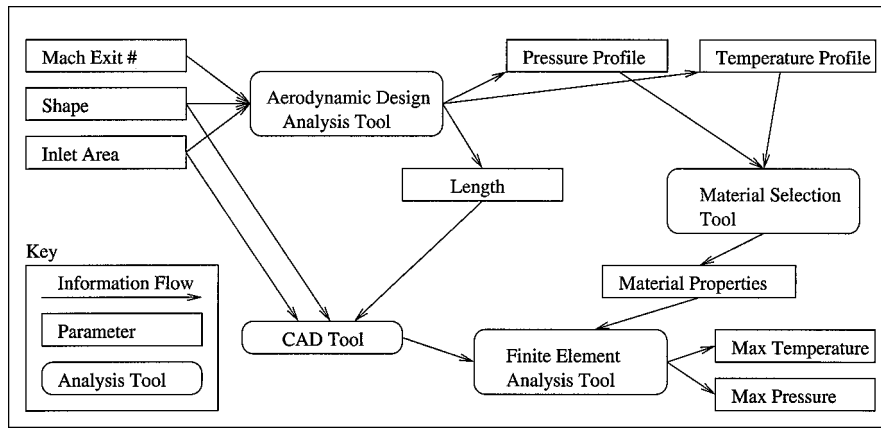
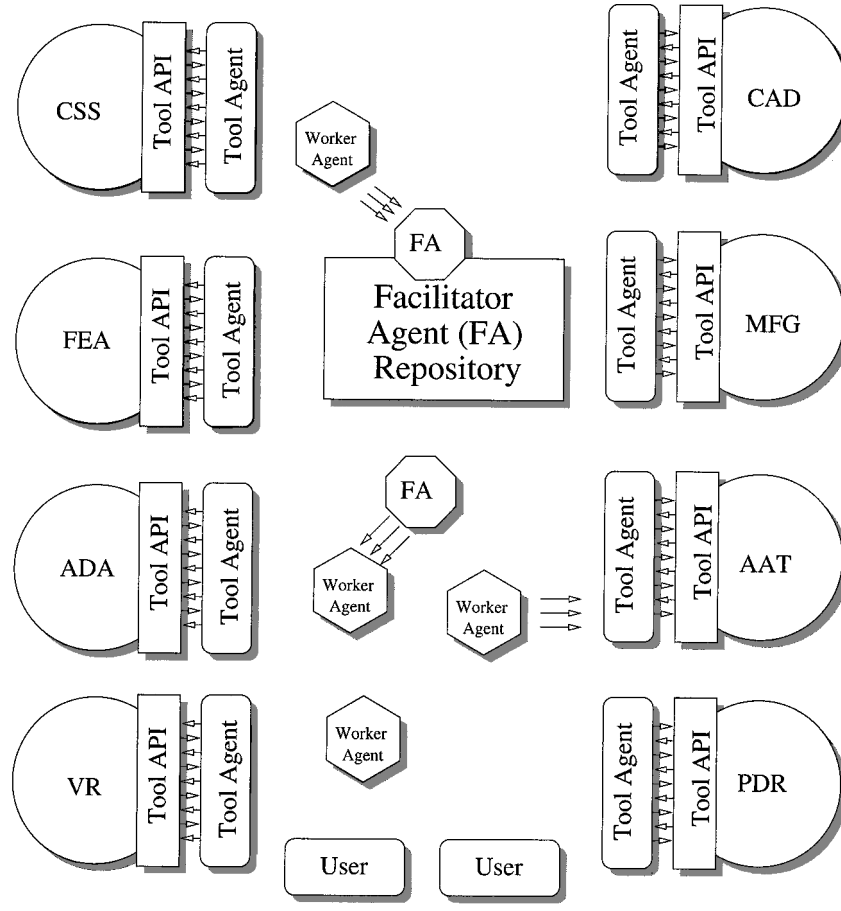


Fig. 2 Details of the data flow for a section of the HSCT nozzle parameter map.



CSS: Constraint Satisfaction and Search
FEA: Finite Element Analysis Tools
ADA: Aerodynamic Design Analysis Tools
AAT: Acoustic Analysis Tools
CAD: Computer Aided Design Tools
MFG: Manufacturing Analysis Tools
VR: Virtual Reality Environment
PDR: Product Data Repository
FA: Facilitator Agent
WA: Worker Agent
TA: Tool Agent
API: Application Program Interface

Fig. 3 MADA model.

community of agents to perform the necessary tasks to complete the design process. Agent-based systems offer advantages in terms of flexibility and fault tolerance over point-to-point integration methods. Another important distinction between MADA and other integration architectures is in the underlying paradigm for accessing the product information reservoir. In most systems described before, an intelligent database contains the common information reservoir for a product design. In MADA, the information remains distributed in the analysis tools and the environment in which it is generated. The exchange of information is not conducted through a central database, but is brokered among distributed tools by worker agents. Thus, MADA provides an open and scalable architecture in which the details of application programming interfaces and application-specific data formats can be made transparent to the end user. MADA provides seamless integration, coordination, and cooperation among diverse analytical tools, which can be changed with minimal impact on the system operation. MADA also presents a single, user friendly interface for accessing the entire range of design processes and data.

System Architecture

The overall functioning of MADA is shown in Fig. 3. MADA is a community of agents residing in a multiagent facility[13] (MAF), which represents a collection of tools, knowledge, and procedures required for collaborative design and analysis tasks. The MAF provides an environment in which the agents exist and perform all of their tasks. The MADA framework contains four distinct software entities: 1) facilitator agents, 2) worker agents, 3) tool agents, and 4) application program interfaces (APIs), in addition to the application or analysis software. The facilitator agents and the worker agents provide means of converting overall design goals into manageable and coordinated tasks, whereas the tool agents and backend APIs provide an abstraction layer that allows the application programs to consume and produce information in a common data and command/communication infrastructure. Specialized facilitator and worker agents interact with multiple tools based on the parameter map for a specific design, thereby providing the means for multidisciplinary collaboration.

An application software may conduct analysis that requires input information from multiple sources. For instance, the manufacturing analysis tool in the HSCT nozzle example needs both material and geometric information. The tool agents provide the necessary intelligence to ensure complete and valid data before the analysis is initiated. The tool agents present information to the rest of the MADA agents though a common agent communication language, KQML. The tool agents reside at a particular node or application location and manage the input/output requirements of application software to accomplish tasks required by the design process. The tool agent resolves any missing data by utilizing the services provided by the worker agents and the product data repository. Once a data set is complete for a specific input/output combination, it is cached and is available for retrieval or future use. If any other agents have requested the information, they are notified when the information set is complete. Each tool agent is concerned only with a specific instance of information and its internal dependencies. The tool agent does not contain any design knowledge or rationale. The tool agents interact with analysis tools via standard wrapper interfaces, thereby isolating their proprietary requirements from the multiagent system. Thus, tools with similar analysis capabilities, such as different finite element analysis software, can be interchanged with minimal system redesign.

Once a design instance information is introduced in the agent environment, it must be managed and transferred between different applications. A class of agents, called worker agents, are created to handle the information transfer. The worker agents contain specific information about individual tool agents and can transfer information between two or more application packages. The worker agents also provide simple data translation services, such as unit conversion. The worker agents monitor any design conflicts or errors that occur within the tool agents. These agents possess knowledge only about information dependencies of different domains.

The job of sequencing and assembling design tasks so that the result is a valid product design is accomplished by the facilitator agent. The facilitator agent takes design requests and translates them into individual worker tasks to accomplish an overall design. The facilitator agents contain specific design knowledge about the product being created. This knowledge is obtained by coding parameter maps that determine the sequencing of the analysis tasks to be undertaken. For example, to do the finite element analysis on the HSCT nozzle, the geometric information must first be produced. The facilitator agents creates worker agents to transfer data from one tool agent to another. The facilitator agent controls the life cycle of the worker agents. Based on the data dependencies, the facilitator agents can also spawn worker agents in parallel to take advantage of the distributed nature of the MADA framework. The facilitator agents collect the necessary information from the worker agents, including the design conflicts and failures. Thus, a facilitator agent produces a single design instance from the input information given by the user or prescribed by the design search process, characterized by a set of input parameters and design constraints. Because all of the design knowledge resides in the facilitator agents, the agent creation

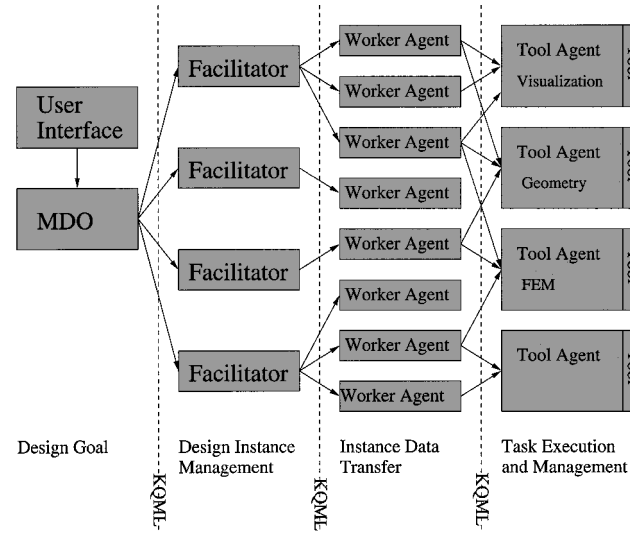


Fig. 4 Execution logic flow in MADA.

represents a crucial task in the MADA framework. The facilitator agents are created by the user input interface if only one iteration is needed. In the case of multiple iterations, the constraint satisfaction and search routine can create multiple facilitator agents with different input parameters to explore the design space.

Figure 4 shows the execution logic in MADA. The user initiates a design cycle by specifying the parametric definition of the object, parameter map of the analyses to be conducted, and the design objective. The design optimization routine then created several facilitator agents to explore the design space, with different attribute values. Each facilitator agent in turn creates worker agents that manage information transfer between specific tool agents. The performance or analysis results are compiled by facilitator agents and used by the optimization routine to select the next set of design alternatives to be evaluated. The design search process terminates when the design constraints are satisfied and a desired performance level is reached.

Computational Framework

The MADA computational environment is composed of domain specific tools, agents, and the environment in which the agents reside. The agent system has been designed using object-oriented and layered programming techniques. Close attention is paid to conforming to standards and isolating proprietary system requirements where possible. This gives the environment the ability to adapt to the changes in specific implementation technologies without affecting in other areas. MADA needs to operate in heterogeneous computing networks and, therefore, must utilize technologies that facilitate interaction between diverse operating platforms. The MADA environment uses JAVA as its core programming language and inherits many key features from it, including the ability to execute the environment any operating system that has a JAVA virtual machine (JVM).[14] The architecture uses remote method invocation (RMI) protocol, which enables JAVA applications to invoke methods remotely through a lightweight API. This technology provides the basis for agent subsystems to be mobile in nonproprietary ways. JAVA and RMI by themselves only provide the means with which heterogeneous, mobile agent systems can be built. An agent management system or MAF, on the other hand, provides a common set of resources and services for mobile agents, such as mobility tracking, data persistence, message passing, naming services, and life-cycle support. Individual agents exist in the MAF environment and rely on it for life-support services.

The MADA agent system uses an agent communication language called KQML.[15] All interagent communications are accomplished using KQML. This provides MADA agents with the ability to interact with third party agents, if necessary. To cooperate, the agents must interact with each other in a meaningful manner, or, in other words, interact socially. Therefore, it is not sufficient just to have

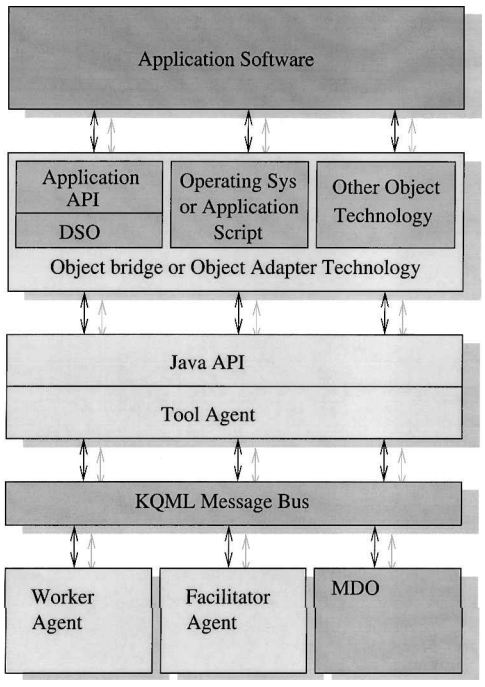


Fig. 5 MADA layers and agents.

communication pathways provided by the MAF between agents, but the agents need to be able to convey intent and goals. KQML provides a means to transfer this information between agents. This is an instance of the layering approach used in the MADA environment. As long as the MAF provides the basic messaging capability, the underlying technology used in MAF is transparent to the user and does not have significant impact on the rest of the architecture. Message passing between agents is accomplished though a message bus or blackboard system using the KQML protocol. Worker agents broadcast information to be used by the tool agents and wait for updates on that data. The tool agents wait for data inputs and analysis requests from worker agents and perform necessary calculations to meet the requirements of the worker agent.

The architecture is designed using a multilayered approach, isolating the complexities of the application domain from the integration framework. This approach gives the system the flexibility to change tools and applications without disrupting other parts of the system. The MADA framework can be decomposed into multiple layers, with different agents embodying one or more of these layers. The details of MADA software layers are shown in Fig. 5.

The first layer is the application software layer. This layer contains the majority of the domain-specific analysis codes. Currently, in MADA, this layer consists of software packages such as Pro-Engineer for CAD, MARC for finite element analysis, Oracle for databases, and several proprietary analysis packages for aerodynamics analysis, acoustics, constraint satisfaction and search, and virtual reality rendering.

To interface the application programs with MADA, they must have a direct connection to the JVM (not shown in Fig. 5). This is accomplished by creating dynamic scripts or linking shared libraries, which connect the applications to the JVM. In the case of dynamic linking, the connection is accomplished by using the JAVA native interface.[16] This interface is shown in the dynamic shared object portion of Fig. 5. This connection is the pathway along which commands and data flow occurs. The direct connection provides an efficient and flexible way to communicate with application packages. Several commercial application packages provide developer libraries that can be used to invoke the application's functionality through the use of public APIs. Pro-Engineer, for example, has Pro-Developer libraries that allow data access.

Once the data connection is established, a JAVA API (JAPI) is used to standardize the method in which the agents store the design instance data. The design instance data represent the parameters or data related to a specific design alternative. The JAPI also provides a standardized method by which application procedures can be invoked. For example, the parameters involved in producing a design geometry, such as the length and thickness of the HSCT nozzle, would be used by the JAPI to interact with the CAD tool to generate detailed geometric specifications. These product independent interfaces are designed to facilitate the plug and play capability of MADA. The agents interact with each other, using the KQML message bus, via this layer. Thus, the interaction among different agents is not dependent on the specific application software used to perform the analysis.

The software tools are wrapped by a JAVA class to provide both the agents and the software tools with a set of services. These services provide information about a running software tool and its information requirements such as the relationships between input and output parameters. The JAVA wrapper stores data related to a specific run of the software for use by the worker agents or other tool agents. The tool agent only manages requests for access to design instance data it creates. If the application software tool has its own set of APIs, this class is used as a transfer and storage point between the JVM and the software native operating systems libraries.

Design Example

We now discuss the use of the MADA framework for a specific product. The design of an exhaust nozzle operating at supersonic Mach numbers, reminiscent of a HSCT nozzle, is chosen as an example. The individual tools used in the integrated design system are kept simple enough to run efficiently on desktop computers with low cycle times. However, the MADA framework is capable of accommodating more complex tools, such as computational fluid dynamics, NASTRAN advanced finite element analysis products, etc. Even with the simple product chosen, the communication and coordination between different tools remains significantly complex. The issues of scalability of the MADA architecture to complex products and assemblies, such as the space station or the next generation space shuttle, are currently being examined.

The different tools used in the integrated system for the HSCT nozzle are 1) the aerodynamics tool, 2) the geometric design tool, 3) the finite element analysis tool, 4) the manufacturing tool, and 5) the constraint satisfaction and search tool. The following sections outline the functioning of the specific tools in the MADA framework.

Aerodynamics Tool

The purpose of the aerodynamic tool is to determine the changes of flow properties inside the nozzle. A simple one-dimensional isentropic flow analysis that describes with reasonable accuracy the global characteristics of the flow inside the nozzle is used. Assumption of sonic flow at the throat gives rise to to an area/Mach number relation that relates the Mach number at any location in the nozzle to the ratio of the local nozzle to the sonic throat area. An ideal expansion at the nozzle exit is also assumed in the analysis. The ambient conditions, nozzle pressure ratio [(NPR) stagnation pressure/ambient pressure] and the temperature ratio [(TR) stagnation temperature/ambient temperature)] are specified as the initial nozzle parameters. In practice, the turbine outlet conditions are typically known before the design of a nozzle is initiated. The performance of the nozzle is then characterized in terms of the NPR and TR. For a given throat area and exit nozzle angle, the length of the nozzle was calculated. Two different exit geometries, axisymmetric and rectangular, were considered for the design. Although the aerodynamic analysis carried out here is relatively simple, the values of most of the important flow parameters inside the nozzle such as the pressure and temperature on the nozzle walls are captured adequately. This is the first tool used by the nozzle design facilitator agent.

Geometric Design Tool

Pro-Engineer is used as the geometric design module. After the CAD tool agent receives the nozzle parameters from the facilitator

agent, the data are used to generate part geometry. The geometry is generated using Pro-Engineer trail files, which give a parametric definition for each of the two nozzle exit geometries. The output of the geometric design tool is stored in the initial graphics exchange specification format. The part geometry file provides the base model for all other analysis tools.

The geometric description of the nozzle is also stored in a three-dimensional VRML format, thereby presenting a standardized viewing platform. To enhance the platform-independent nature of the environment, the VRML2.0 three-dimensional standard was used. All visual data to be presented to the user are converted from their native representation into VRML.

Materials Database

To achieve an optimal product design, the selection of materials plays a crucial role. In the context of current design practices, this important step is carried out largely from the experience of the materials engineer in consultation with the engineering mechanist and the manufacturing engineer. This poses a major threat to the rapid deployment of evolving but promising candidate materials and manufacturing methods. The materials system can be chosen from a range of metals, ceramics, and plastics in monolithic or composite form. They are largely chosen based on the physical, chemical, thermal, and mechanical properties apart from cost and manufacturing considerations. In the context of MADA environment, we have established a material database, called MATDAT, comprising a wide range of materials relevant to the design of aerospace components operating under elevated temperature and severe loading conditions. MATDAT, when accessed through JAPI, is able to locate the right material in addition to providing all of the required thermomechanical data (Youngs modulus, Poisson ratio, thermal conductivity, thermal expansion, yield strength, and inelastic constitutive law) and allowable limits, for example, maximum von Mises stress, allowable fracture toughness, and maximum strain limits. These data are not only essential in the analysis stage but also in driving the iterative design solutions based on cost and weight considerations.

Finite Element Analysis Tool

In the present case, where two different geometric configurations are considered, several iterations may be undertaken for each geometry with different parameters. The finite element analysis tool uses the analysis software MARC and its graphical front-end MENTAT. When the facilitator agent triggers the MARC tool agent, the preprocessor MENTAT accesses the geometric data generated by Pro-Engineer. MENTAT discretizes the domain using bilinear thick shell elements. This element is specifically chosen because it can model variations of thickness and pressure loading along both the x and y directions. The element is based on membrane, and the formulation captures both displacement and curvatures essential for nonlinear analysis, but is still inexpensive compared to full three-dimensional elements. About 800 elements are used in the analysis, and the pressure loading is directly input from the aerodynamics module. Though automatic mesh enrichment and remeshing are essential for capturing sharp stress gradients, they are not needed for the current problem. The finite element code MARC is invoked automatically once the mesh generation and boundary conditions are available. MARC analyzes a given design configuration and evaluates the maximum stress, strain, displacement, and the specific location. The actual stresses in the nozzle compared to the allowable stress supplied by the MATDAT tool then determines the next design iteration based on the goals specified in the constraints module.

Manufacturing Tool

The manufacturing planning tool in Pro-Engineer is used in the current implementation of MADA. The manufacturing tool agent receives the geometry and material data from the facilitator agent. The output of the manufacturing module is a set of numerical control tool paths for generating the required geometry and an estimate of the total manufacturing time. In case of manufacturing errors or infeasibilities, the tool agent alerts the facilitator agent, which can initiate a new design instance with proper corrective actions.

The manufacturing module provides a time estimate, which, in addition to the material selection module, can be used to estimate the manufacturing cost for the design instance. This information is used by the constraint satisfaction and search tool to determine the quality of the solution generated.

Constraint Satisfaction and Search Tool

The constraint satisfaction and search routine contains global knowledge of the system. This information is not as detailed as the knowledge contained in the facilitator agents because it pertains to the critical performance metrics for evaluating the design instances. The constraint satisfaction and search routines create the facilitator agents with specific parameters and evaluate the resulting designs. The designs are refined via an iterative process, based on the design goals and search routines used. A specialized subgradient search procedure is used to guide the iterations in the present implementation of MADA. The asynchronous, distributed nature of the MADA agents allows several design instances to be created simultaneously. Thus, the constraint satisfaction and search routines drive the entire system. They manage both design conflicts and constraints. The search routines monitor the facilitator agents and the design instances. If a specific design instance violates a constraint, it is removed from consideration. However, this information is used to initiate another design instance that is within the feasible region. Thus, by monitoring intermediate data, the constraint satisfaction and search routines can also detect conflicting design goals and alert/query the user for explanation.

Design history is preserved by storing input and output parameters in the persistent datastore. Any data that cannot be recreated by invoking tools are stored here along with the resultant outputs and actions. This way the designers, as well as the search routine, can use the information to review and search the previously explored design space. This reduces the need to process the data again. Designers can annotate design instances to provide design intent information.

As the design evolves, the related information is fed into a virtual environment, where the user can monitor the progress and, if necessary, modify or halt the process. This environment provides a user friendly interface to the product data and allows the user to control the design process.

Figure 6 shows the screen capture of MADA in operation. The example shown in Fig. 6 is of a square nozzle configuration. The first window shows the input worker. The following images show the MAF environment, the nozzle geometry in Pro-Engineer, the numerical control tool path simulation, finite element analysis in MARC, and, finally, the VRML realization of a feasible nozzle design.

Computational Results

We now present computational results for a supersonic nozzle design. Table 1 shows the inputs and outputs of the system for the nozzle design. To control the search of the design space, MADA allows the specification of various input parameters and constraints. The parameters that determine the performance characteristics of the nozzle are listed in the input section of Table 1. The thickness parameter has the additional capability of specifying a range. Two nozzle input specifications were selected from Ref. 17 and are listed as cases 1 and 2 in Table 1. These two cases produce a total of four resultant configurations, axisymmetric and rectangular geometry for each case. The design search for these configurations was constrained by the material selection and thickness parameters. Once additional information is known about the design domain, it can be included in the constraints to accelerate the search process. The results of the design search by MADA are summarized in the output section of Table 1.

The metrics listed in the output section of Table 1 are a subset of the design details and performance metrics generated during the design cycle. Thickness, length, and material are the output parameters for each configuration. The cost coefficient is the metric used to drive

Table 1 Design instances generated by MADA for a supersonic nozzle

	Case	
Specification	1	2
Input		
Nozzle pressure ratio	3.4	4.5
Ambient pressure, Pa	1.00E+05	1.00E+05
Ambient temperature, K	294	294
Temperature ratio	3.0	3.45
Mach number at exit	1.46	1.66
Starting area, m^2	0.125	0.125
Thickness upperbound, m	0.01	0.01
Thickness lowerbound, m	0.005	0.005
Output		
Geometry	Axisymmetric	Axisymmetric
Thickness, m	0.005	0.0075
Length, m	0.1652	0.3206
Material	Aluminum	Aluminum
Cost coefficient	7.2684	7.2702
Number of iterations	9	75
Av. run time for failed runs, ms	4.78	4.42
Av. run time for successful runs, ms	91,444.25	15,198.4
Geometry	Rectangular	Rectangular
Thickness, m	0.005	0.005
Length, m	0.1652	0.3206
Material	Aluminum	Aluminum
Cost coefficient	13.4896	24.3036
Number of iterations	150	75
Av. run time for failed runs, ms	4.91	4.68
Av. run time for successful runs, ms	110,353.7	115,578.3

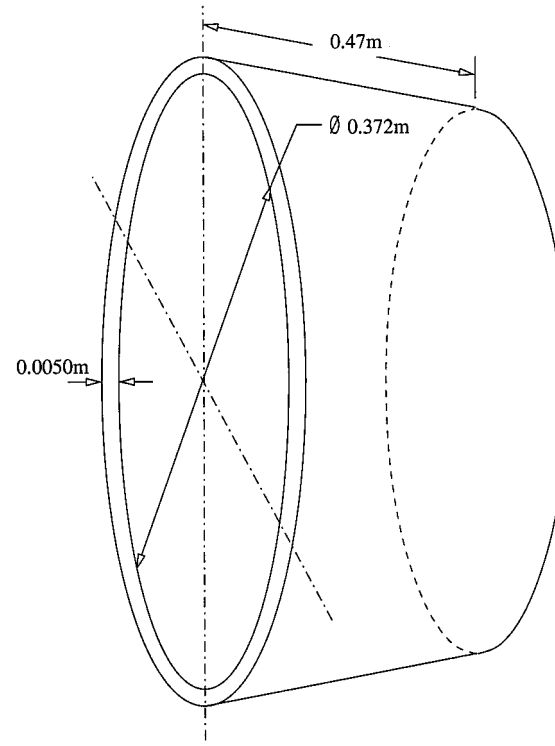


Fig. 7 Axisymmetric nozzle design instance.

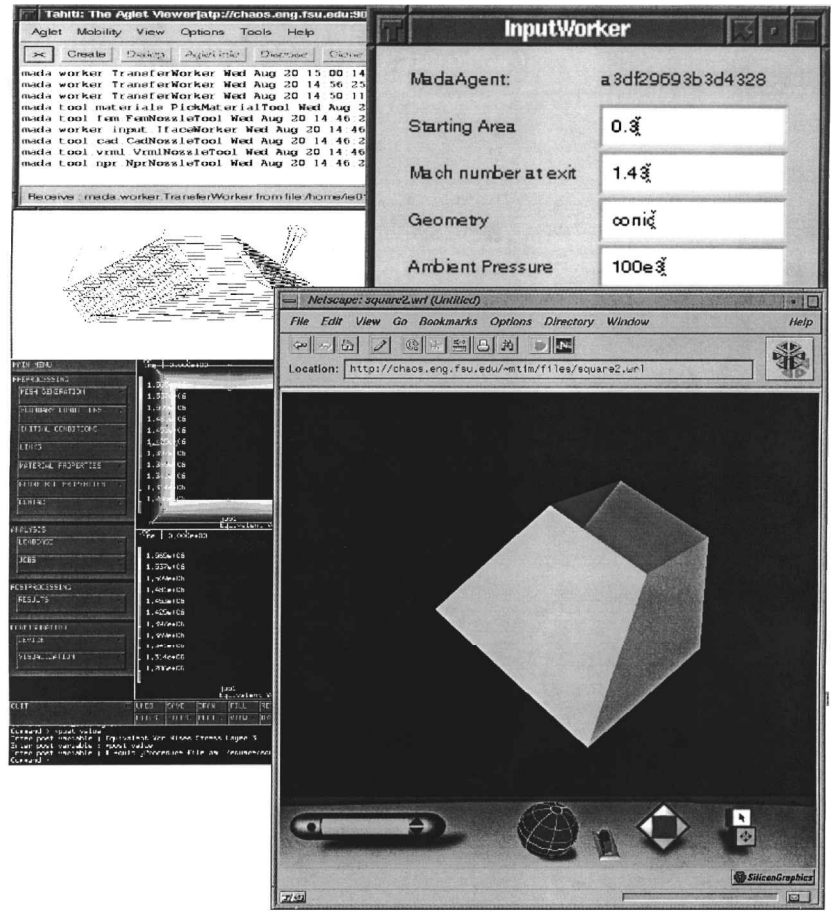


Fig. 6 Screen capture of MADA environment for HSCT nozzle.

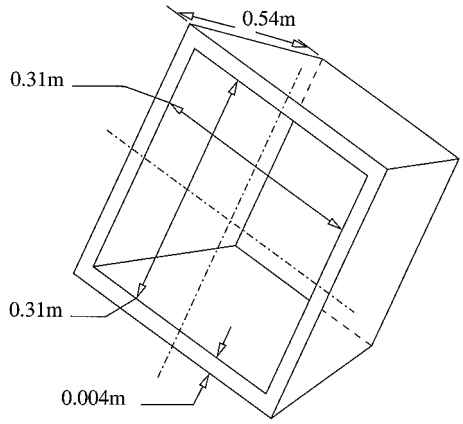


Fig. 8 Rectangular nozzle design instance.

the search routine. Note that the cost coefficient is a dimensionless metric. Actual cost values depend on the manufacturing equipment used and the raw material costs. The final three outputs in Table 1 are performance metrics of MADA as a design system. The number of iterations is the number of individual design attempts made by the search routine. This number does not reflect the number of invalid designs immediately rejected by the facilitator agent. The processing time of these invalid designs is listed as failed runs. The time for successful runs represents the instances when the design cycle was completed and resulted in a feasible design. The final configurations for the axisymmetric geometry for case 1 is shown in Fig. 7 and that for the rectangular geometry for case 2 is shown in Fig. 8.

Summary

The current implementation of the MADA environment has static knowledge encoded in the layers and agents. Learning systems, such as neural nets, expert systems, or other artificial intelligence tools, can be used to enable the agents to obtain knowledge dynamically and learn intelligently about the environment as the system evolves. Software utilities could be created to reduce the work needed in making the data connection between the tool agent and the application programs. The addition of a strong link between the tool agents and the application programs would increase the implementation domain for MADA.

The current methods used to implement the transfer of information from the application to the JVM involve scripts and proprietary library interfaces. As the JAVA matures, object interface products can be used to connect JAVA to existing application data communication protocols. These technologies include N-OLE by Microsoft and CORBA by the Object Management Group.

The future extensions of the MADA environment will address the scaling and communication issues for complex products and assemblies.

Acknowledgments

This work was funded in part by Grant NAG-2-1114 from the NASA Ames Research Center. The authors acknowledge the assistance of S. Awoniyi, K. Leibkuchler, I. Dinov, and M. Challa in creating the MADA framework and of J. Carlson in collecting information about other integrated design activities. We thank the Associate Editor and anonymous referees for insightful comments.

References

[1]Mecham, M., "Lockheed Martin Develops Virtual Reality Tools for JSF," *Aviation Week and Space Technology*, Vol. 147, No. 14, 1997, pp. 51–53.

[2]Valenti, M., "Re-Engineering Aerospace Design," *Mechanical Engineering*, Vol. 120, Jan. 1998, pp. 70–72.

[3]Bloor, S., and Owen, J., *Product Data Exchange*, Univ. College London Press, London, 1995.

[4]Regli, W. C., and Gaines, D. M., "National Repository for Design and Process Planning," *Computer-Aided Design*, Vol. 29, No. 12, 1997, pp. 895–905.

[5]Sriram, R., Gorti, S., Gupta, A., Kim, G., and Wong, A., "An Object-Oriented Representation for Product and Design Processes," *Journal of CAD*, Vol. 30, No. 7, 1998, pp. 489–501.

[6]Goldin, D. S., Venneri, S. L., and Noor, A. K., "Ready for the Future?" *Mechanical Engineering*, Vol. 121, No. 11, 1999, pp. 61–64.

[7]Lander, S. E., "Issues in Multiagent Design Systems," *IEEE Expert*, Vol. 128, No. 3, 1997, pp. 18–26.

[8]Parker, L., "ALLIANCE: An Architecture for Fault Tolerant, Cooperative Control of Heterogeneous Mobile Robots," *In Proceedings of the 1994 International Conference on Intelligent Robots and Systems (IROS '94)*, IEEE, Piscataway, NJ, 1994, pp. 776–783.

[9]Smith, R., "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver," *IEEE Transactions on Computers*, Vol. 23, No. 12, 1980, pp. 1104–1113.

[10]Lewis, W., "Data Flow Architectures for Distributed Control of Computer Operated Manufacturing Systems: Structure and Simulated Applications," Ph.D. Dissertation, Dept. of Industrial Engineering, Purdue Univ., West Lafayette, IN, May 1981.

[11]Garvey, A., and Lesser, V., "Representing and Scheduling Satisficing Tasks," *In Imprecise and Approximate Computation*, S. Natarajan (ed.). Kluwer Academic, Norwell, MA, 1995, pp. 23–34, Chap. 2.

[12]Jennings, N. R., Sycara, K. P., and Wooldridge, M., "A Roadmap of Agent Research and Development," *Journal of Autonomous Agents and Multi-Agent Systems*, Vol. 1, No. 1, 1998, pp. 7–36.

[13]"Mobile Agent Facility Specification," Object Management Group, TR OMG TC cf/97-06-04, Crystaliz, Inc., General Magic, Inc., GMD FOKUS, and IBM Corp., June 1997.

[14]Lindholm, T., and Yellin, F., *The JAVA™ Virtual Machine Specification*, 2nd ed., Addison-Wesley Publishing Co., Longman, Reading, MA, 1999.

[15]Finin, T., Fritzson, R., McKay, D., and McEntire, R., "KQML as an Agent Communication Language," *Proceedings of the 3rd International Conference on Information and Knowledge Management, CIKM'94*, Association for Computing Machinery, Gaithersburg, MD, 1994, p. 456.

[16]Liang, S., *The Java Native Interface: Programmer's Guide and Specification (Java Series)*, Addison-Wesley Publishing Co., Longman, Reading, MA, 1999.

[17]Krothapalli, A., Soderman, P., Allen, C., Hayes, J. A., and Jaeger, S. M., "Flight Effects on the Far-Field Noise of a Heated Supersonic Jet," *AIAA Journal*, Vol. 35, No. 6, 1997, pp. 952–957.