

The Internet Engineering Design Agent System: *iEDA*

Clark J. Radcliffe
Mechanical Engineering
Michigan State University
East Lansing, MI 48824
(radcliffe@me.msu.edu)

Jon Sticklen
Computer Science & Engineering
Michigan State University
East Lansing, MI 48824
(sticklen@islnotes.cse.msu.edu)

Gary J. Gosciak
Mechanical Engineering
Michigan State University
East Lansing, MI 48824
(ggosciak@yahoo.com)

ABSTRACT

Approaches to engineering design and manufacturing such as *integrated design and manufacture* and *just in time fabrication* depend on interaction with and among component supply companies which most often use very diverse technologies. The Internet Engineering Design Agents (*iEDA*) approach is a distributed, component-based, agent methodology that is realized following a strong black box approach to modeling. An individual Design Agent (DA) is a *virtual product* capable of encapsulating both descriptive and model based information about the product it represents. Hierarchically recursive agents for sub-systems and/or components are linked via a communications network to form larger integrated model systems. A crucial part of the *iEDA* architecture is a global ontology of questions; these questions in the ontology are standardized engineering domain queries that form a sufficient set of queries to allow compositional modeling methods; the global query ontology is itself realized as a networked agent. Most importantly, Design Agents interact **without divulging the proprietary descriptive information or models** contained within component or device represented. The *iEDA* architecture thus enables both proprietary security and compositional modeling of component parts/systems. The structure and function of the *iEDA* architecture and its current implementation is discussed. A two dimensional bridge system model is used as an example to illustrate the distributed nature of assemblies and components registered as DA's on a communications network. *iEDA* forms a distributed modeling environment that enables communication and coordination required for effective and efficient global collaborative distributed design.

INTRODUCTION

A strongly evolving need of the scientific and engineering communities in the United States focuses on the necessity to rapidly assemble computational models for *parts* into an effective model for a *whole*. Engineering design and manufacturing using “integrated design and manufacture” and “just in time fabrication” of products likewise depend on global interaction with component supply companies using diverse technologies. Rapid, systematic, assembly of product models from models of the subsystems and component provided by suppliers is critical to rapid product engineering, manufacture and marketing. The technologies employed by suppliers use individuals from very different disciplines in which practitioners often use, and even more importantly think in different vocabularies. Solutions of large scale, real world product development problems depend on knowledge and expertise from multiple disciplines. A key to enabling efficient and effective multidisciplinary research and development lies in effective use of shared engineering and business models.

An illustration lies in the engineering design communities within the automotive sector. As the large U.S. automobile manufacturers have moved increasingly away from ground-up manufacture of their products and towards the role of integrator, a loose spiderweb has evolved linking the large *integrators* with first and second tier *suppliers* of component parts. These suppliers can be either historically independent or spinoffs from the automotive integrators such as Visteon Automotive (Ford) or Delphi Automotive (GM). This integrator-supplier business model has arisen in response to two business issues: quality control and price. Individual suppliers commonly deal with an integrator using other components supplied by competitors. Individual product integrators use suppliers also providing parts to competing product integrators.

This automotive integrator-supplier business spiderweb has resulted in increasing difficulty associated with engineering proprietary information. We focus on solutions to the engineering computational modeling problem the diverse, global, supplier-integrator business spiderweb while protecting the proprietary information.

Manufacturing integrators and their suppliers are doing more computational design with less prototype testing employed in the design process. At the current state of the art, to reach a substantial level of function confidence in a *supplier's component*, an integrator must typically obtain a computational model of a proposed component (a structure, a motor, a transmission, ...) and integrate it into an computational *integrator's assembly* model (a car...). The cardinality of the part count for such a model is large; Daimler-Chrysler's Intrepid vehicle model contains over 5500 parts. Holding a complete computational model of a component places one well on the way to being able to manufacture that component. Thus the *supplier* will seek intellectual property protection via some type of proprietary information agreement with the *integrator*. Such legal protection is a major focus of the integrator/supplier relationship currently, and a primary impediment to a rapid design-to-manufacture industrial cycle.

There are three resultant broad problems. First, the incorporation of new engineering developments into the *integrator's* repertoire is substantially slowed. The establishment of each new thread in the spiderweb requires substantial "legal time" to establish. Second, direct interaction between suppliers even within one *integrator's* supply chain spiderweb is impeded; proprietary information agreements are typically bilateral arrangements between one *integrator* and one *supplier* - not a blanket arrangement covering all participants in the spiderweb. And third, information exchange beyond the walls of a given spiderweb is seriously impeded because it would be outside the legally "safe" zone of information sharing.

A specific example of the third general problem tangibly illustrates the difficulty. Automotive design engineers need engineering design data for their new vehicle. The design uses tires and wheels from separate suppliers within the automobile assembler's supply spiderweb. The tire supplier manufactures wheels while the wheel supplier also manufactures tires. The automobile assembler needs detailed design information to integrate computational finite element models for structural properties of the wheel and tire into the vehicle system model. The suppliers are unwilling to share design model data until proprietary legal safeguards are in place. The assembler critically needs the tire/while model for their vehicle structural model to predict ride and handling properties, however, months of legal interchange may be required before the situation is resolved. This specific example is real and typical.

In summary, free distribution engineering/business component performance information is needed to build system models while protecting the proprietary nature of that engineering/business component model data. A modeling system is required that does not directly distribute

computational model data by any means. Distribution of model data modules encrypted by Java or other languages is not adequate for data security. The best solution is a modeling mechanism which links models through distributed performance information rather than the model data that underlies that component performance. *We have developed methods for linking virtual components based on internet distributed agents modeling physical components.*

VISION, GOALS, AND OBJECTIVES

Our project vision is to lay the foundation for replacing the legal spiderweb in engineering communities of practice with design-oriented, open, internet-based collaboration between integrators and suppliers. Our central goal is to enable sub-system and system integrators to rapidly generate and use distributed computational models from suppliers' internet-published models of components that protect suppliers' proprietary information. Moreover, we seek to improve economic performance for a key knowledge generating activity of integrators: integrated reuse of supplier models.

The **kernel technical goal** of the project lies in enabling distributed, model agents that return to the caller information on a part's properties (e.g., color, strength, weight, ..) and model responses (static response, dynamic response) without revealing the details of proprietary engineering designs that generate those properties and model responses.

There are two research sub-goals which are precursors for achieving our broad project goal.

First we must develop device modeling methods which allow distributed component models to interact within an integrated device model, but *while hiding proprietary details of the model*. In the computer science sense, we seek to broadly apply principles of information hiding and abstraction to address the problems of the integrator/supplier legal spiderweb as discussed above.

Second, we must develop a fixed internet-agent architecture which supports task directed, distributed problem solving including the use of the component models. The "*Web-Agent*" is currently one of the most active research areas in computer science. A generally accepted, task-based description for a general web agent has not emerged from the community. Likewise, no generally accepted web agent architecture has emerged.

AGENT TECHNOLOGY AS A SOLUTION

The idea of software "agents" can be viewed as somewhat similar to the root concept of task specific architectures, but contains additional features. Software agent research has evolved from Artificial Intelligence and Software Engineering. Agent technology is still in a formative stage, as can be seen by compiling from recent literature a list of "defining features" which agents should have [Wooldridge, 1995 ; Nwana, 1996 ; Franklin, 1996 ; Shoham, 1997 ; Sycara, 1996 ; Etzioni, 1994 ; Gilbert, 1997]: autonomous, goal-driven, reactive, temporal continuous, social ability, cooperative, reasoning capability, learning and adaptive capability, personality, mobile, taskable,

network-centric, persistent, believable, benevolent, rational etc...

Despite the difficulty in reaching consensus on what constitutes an "agent" the kernel intuitions about software agents persists: agents perform useful work for their "owner" and typically do so by interacting with other agents at some other physical location. For some agent researchers, the confluence of task specific architectures and agent research is clear. The RETSINA software system at CMU [Sycara, 1996] employed a layered structure in which agents were assigned different task specific roles. The central idea of agents that perform very targeted tasks is resonant with a Generic Task (GT) viewpoint.

Agent software technology allows tools to assist global collaborative engineering. An agent (Tecuci, et al, 1998) is a Knowledge Based System (KBS) that perceives its environment; reasons to interpret perceptions, draw inferences, solve problems, and determine actions; and acts upon that environment to realize a set of goals or tasks for which it was assigned. Software agents have autonomous, responsive, proactive, reasoning/learning, collaborative behavior (Murch, Johnson, 1999, Brenner, et al 1999, Wooldridge, Jennings, 1998, Bradshaw, 1997). An agent's environment may be the physical world, a person using a graphical user interface, and a collection of other agents, the Internet, or other complex environment. Agents are able to act on their own in a changing environment prescribed by a set of goals and with other agents, further developing as they proceed with a task. Agents provide the opportunity to autonomously represent the engineering behavior of individual physical components in a distributed engineering design environment. Software with this capability can be coupled with existing corporate KBS comprised in part by CAE technology. This combination provides tools capable of bringing global collaborative engineering design to the next level.

An agent's autonomous, responsive, proactive learning nature addresses global collaborative engineering design's 24-hour synchronization challenge. The agent can function autonomously. Having a goal, rule orientated composition allows for operation without constant supervision from the corporation providing the agent. In any time zone around the globe, the agent is operational, reacting promptly to changing operational conditions. Timely agent response facilitates synchronization.

An agent's collaborative behavior addresses global collaborative engineering design's proprietary and software platform handicaps. Agent software can effectively collaborate on a solution to a problem in a distributed environment. A global multi-corporate engineering design team is a distributed environment where expertise and knowledge is not contained in a central location. A MAS (Multi-Agent System) allows the integration of existing Design Agents of an individual corporation into a larger system without the need of collocating the individual DA's into a single DA at a single location (Brenner, et al, 1998). A collection of agents representing the individual corporate expertise and knowledge makes up the

MAS. A Multi-Agent System's, as well as the global collaborative engineering design team's competency is the result of the aggregate competency of the individual Design Agents not contained at central DA. Appropriately developed DA's would allow communication and interaction of engineering information without compromising the protection of participant's proprietary nature. The partners of a multi-corporate design team through the DA's communication protocol may cooperate seamlessly through the MAS regardless of specific partner software platforms. Exclusive proprietary information and independent software platform challenges are circumvented through agent technology.

Recent years have shown agent methodology as a tool for assisting the workforce is becoming realized more and more every day as a viable asset to keeping a competitive edge. Agent software technology has been finding its way into industrial and commercial applications such as process control, manufacturing, information and business process management, and electronic commerce (Wooldridge, Jennings, 1998). This has lead to current research and exploration into the use and validity of agents in engineering design. The CADOM project (Component Agent Design Orientated Modeling) utilizes agent methodology to assist application integration at the design data level (Rosenman, Wang 1999). This methodology functions to facilitate in-house collaborative engineering. The DOME project has agents as modules, which perform the service exchange in an integrated product model via a network. There work suggests that after the overhead of evaluation time in creating the first integrated system model, subsequent system evaluations with design tradeoffs can be reduced from months to minutes (Wallace, et al, 1999). The DOME project has recognized that detailed proprietary information is a barrier to this methodology's success.

Software agent technology provides autonomous, collaborative capability towards evolving globally distributed engineering design environments. The system of Internet Engineering Design Agents (*iEDA*) described below uses a MAS approach to represent physical components developed for a distributed modeling mechanism environment. *iEDA* form virtual components leveraged by individual corporations for assisting in effective and efficient global collaborative engineering design. The encapsulation of a corporation's knowledge in *iEDA* protects detailed proprietary engineering information.

INTERNET ENGINEERING DESIGN AGENTS

The system of Internet Engineering Design Agents is organized to facilitate the exchange of engineering design performance data between corporate organizations while protecting the proprietary design information yielding that performance. A minimal implementation of a system of *iEDA* (Figure 1) includes query software, an agent registry, global ontology, and a distributed set of Design Agents(DA). The query software is used by a customer seeking design performance information and generates queries to design agents. The agent registry is the agent that contains the

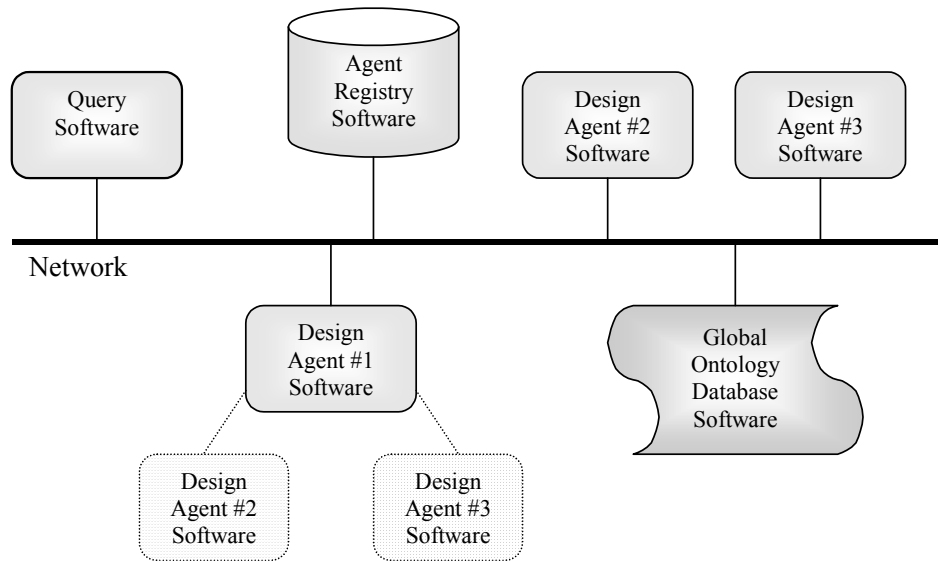


Figure 1. The Internet Engineering Design Agent System: *iEDA*

locations of all the design agents connected to the network. A list of all valid queries for any design agent is standardized and resides in the global ontology. Each DA is a virtual model of a physical component, sub-system, or system. The collaboration of these entities through the system of *iEDA* allows distributed design performance modeling.

The *iEDA* system user first uses query software (Figure 1) to select a valid agent query from the global ontology. The user's query software then selects a Design Agent (#1) to provide an answer to that selected query. When the standard query is sent to DA #1, that agent may use other Agents (#2 and #3) registered on the system as resources to generate the response to the query. This interaction occurs within DA #1 and is not visible to the query software. The interaction between DA #1 and DA's #2 and #3 uses the same agent registry and global ontology that generated the original query to DA#1. The *iEDA* topology allows decomposition of the knowledge base used to answer queries while protecting proprietary knowledge.

A global collaborative engineering system model formed using *iEDA* is comprised of sub-system and components from participating suppliers. Participating suppliers would publish their product sub-system or component in DA form. When the product is connected to the network, the *iEDA* knowledge is kept in-house and only the internet address of the agent is published in the agent registry insuring protection of proprietary data. The client software queries the agent registry to find a particular component product. The agent registry responds with the location of the desired product and the query client utilizes the global ontology of standardized queries to then generate queries to the cooperating DA in the *iEDA*. The cooperating DA may be a complex product assembly consisting of many components and subassemblies. An individual DA in

the system of *iEDA* represents each component or sub-assembly. Through the ontology, these component-based *iEDA* cooperate above application dependant platforms to formulate an answer to a particular collaborative design query. Although DA's within the *iEDA* may respond to the entire global ontology, it is anticipated that most will only respond to queries from the global ontology relevant to the agent's physical component and for which a knowledge base is available from the DA's parent organization. Relevant ontology responses are developed from the in-house knowledge that experts, physical testing, or CAE models that the corporation may provide. As the design of product progresses, the available ontology responses would increase in number and detail.

Our *iEDA* approach involves component-based agent methodology utilizing a strict input/output, query/answer protocol. Anything from a shock, brake-pad, tire, alternator, engine, to a propeller, turbine blade, etc. can be an *iEDA*. The *iEDA* system's purpose is to coordinate DA's with other DA's in a MAS making a decomposable collaborative modeling environment. The MAS model composition of a system, sub-system, and component *iEDA* consists of a decentralized distributed tree structure. Through the agent topology, no matter what level of complexity is represented an equivalent agent functionality is retained. Communication through the global ontology sets a strict input/output methodology for interaction. Recent modular modeling research has shown that modular model elements may be assembled into larger models through a strict power-based input/output communication (Byam, and Radcliffe, 2000, Byam, and Radcliffe, 1999). These results provide the fixed input-output causal structure needed for independent agent-based distributed modular models.

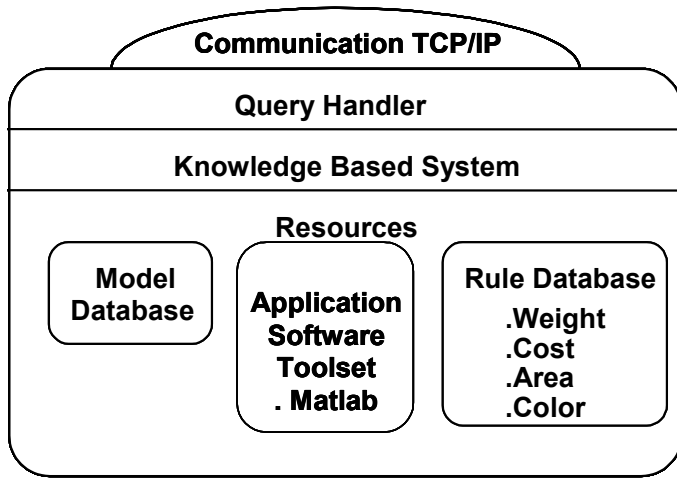


Figure 2. *iEDA* Agent Architecture

INTERNET ENGINEERING DESIGN AGENT STRUCTURE

Individual Design agents in the *iEDA* system contain communication methods and the resources required to respond to queries. Connecting communication and resources are the methods required to connect individual queries with the specific resource required to respond to that query. The structure of the DA proposed (Figure 2) includes communication with the internet through TCP/IP or other standard protocol. The query handler that parses the incoming query, schedules it for processing and passes it to the agent knowledge base. The knowledge base that identifies the resources required to answer the query, passes the query in whole or part to those resources and assembles the agent response. The query handler then receives the response generated and passes it to the communication system to the client or agent generating that query.

The query handler of the DA defines a DA. The strict query and answer protocol (Figure 3) enables the Design Agent to communicate with the outside world without comprising its internal nature. Because the queries, along with the answers, are structured in a standard way, query and answer context do not convey any information available through contextual inferences.

A basic definition of a Knowledge-based system (KBS) (Tasso, Oliveira, 1998) is a software system able to explicitly represent the knowledge of a given domain and capable of high-level problem solving through reasoning mechanisms through the knowledge base. A KBS is built up from the declarative knowledge of a domain (Dymm, Levitt, 1991). The domain may be any of the disciplines included in the use or design of a product. The declarative knowledge may include the information databases, past experience, and most importantly expert knowledge of a particular task. The abstract view of a KBS consists of a central kernel along with a collection of special purpose modules. The kernel here is the problem solving capability of the *iEDA* to return to the caller

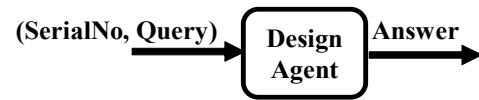


Figure 3. *iEDA* Agent Standard Communication Protocol

information on a part's properties (e.g., color, cost, weight, ...) and model responses (static response, dynamic response) without revealing the details of the proprietary engineering designs that generate those properties and model responses.

A Design Agent within the *iEDA* system may either be a single physical component, or constructed as system using the responses of the physical components from which it is physically constructed. Although these are different virtual representations internally, the agent architecture remains the same. Conceptually, the only difference between system and a component is that the resources of systems may consult agents representing the system's physical components as resources.

EXAMPLE

The *iEDA* prototype demonstrated here includes distributed Design Agents representing eight companies collaborating to provide product information. This implementation includes an agent registry and global ontology to coordinate the interaction between the agents and client query software. The user interface of the client software (Figure 4) may remotely access the network and connect to any one of the various companies. The prototype has a total of eleven networked nodes: a query client, an agent registry, a global ontology and eight companies publishing 32 products. For this example, the products are components and assemblies of bridges: bars trusses and spans.

When started, the *iEDA* client software first consults the agent registry and obtains a list of those agents currently available and the network address of each agent. This list includes the address of the ontology agent and is used to obtain a list of valid queries from the global ontology. Once the client is initialized with the agents available and a list of valid queries, the design agents can be queried for information.

The *iEDA* client software (Figure 4) allows a user to choose a company and part of interest. There are 3 bar companies, 3 truss companies, and 2 span companies providing DA's. The DA representing the bar companies are resources to the DA of the truss and span companies, and the DA of the truss companies are resources to span companies' DA. Once a company is chosen the company's list of available product serial numbers may be accessed and a product chosen. The client software allows the user to choose from (Table 1) different queries and initiate a query to the selected design agent. The Design Agents use the standard communication protocol to receive the query, assemble the solution, and return an answer.

Table 1. Typical User Queries

Cost	Weight	Area	Color	Delivery	Stiffness	Strength
------	--------	------	-------	----------	-----------	----------

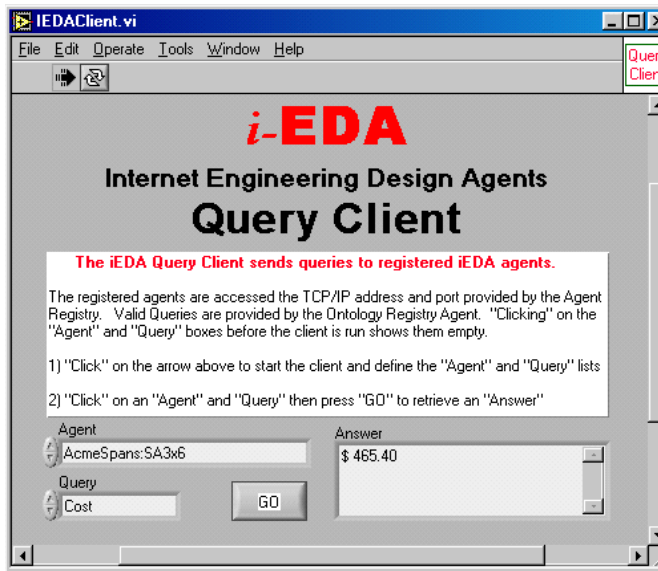


Figure 4. The user interface for the client software

In the example of (Figure 4), the user is interested in the cost of a span from the Acme Spans Company. The generation of an answer to the cost query sent to the Acme Spans SA3x6 system begins with the receipt of the User Query string (Figure 5). The client software has sent a query that includes the SA3x6 serial number plus the Cost query. The query handler of the AcmeSpans Agent parses the query string into a form suitable for the Agent's knowledge base and the resources it uses. Many of these resources may then employ queries to agents representing the physical components of systems such as the Acme Span SA3x6. The appropriate query or queries are then sent to these components. Figure 5 shows such a case with both the query and answer generated at each level.

The Acme Spans agent uses its knowledge base and resources to answer the cost query for span SA3x6. The knowledge base uses the serial number to identify the design and component manufacturer and serial numbers of the components used to assemble span SA3x6. In this case (Figure 5), two TAx6 trusses from Beal Truss Company and a BA1x2 bar from Allied Bar Company are used to assemble the Span. The locations of these subsystems and components are obtained by the Acme Spans agent from the agent registry. The AcmeSpans Agent uses its internal client to query the subsystem and components comprising SA3x6. BealTrusses and AlliedBars companies provide DA's for the truss subsystem and bar component located on remote *iEDA* nodes. As it happens, Beal Trusses are assembled from bars purchased from Allied Bars and require further queries to the agents representing those components.

The dissemination of the cost query through the agents required to generate an answer is shown in Table 2. This process begins at query 1 (Figure 5, 6 and Table 2) network

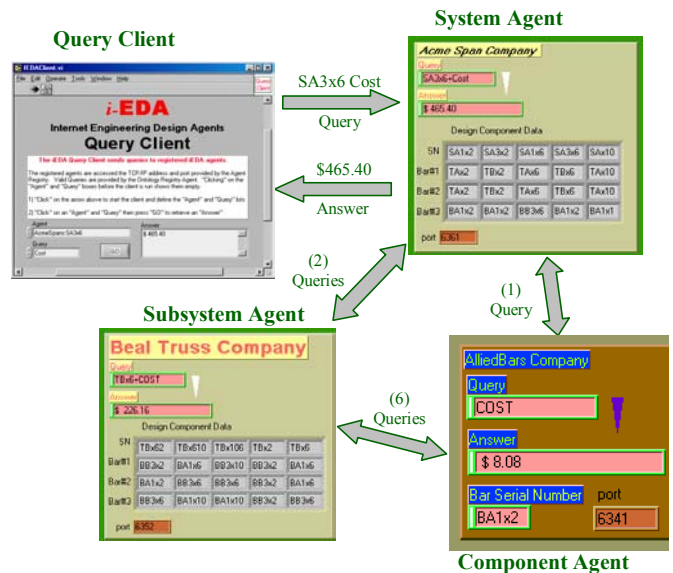


Figure 5. AcmeSpans agent interaction for SA3x6 Cost solution

call. The query handler of the BealTrusses Agent receives the query string of a serial number plus the desired query from the AcmeSpans Agent. Since span SA3x6 is comprised of two subsystem trusses from BealTrusses, this also occurs at network 9. The AcmeSpans Agent receives the answers to the query from BealTrusses in networks 8 and 16 for 1 and 9, respectively. The gap between network calls 1 and 8 results from the BealTrusses Agent performing network calls to obtain the answer supplied at network call 8. In an identical process, the AlliedBars Agent query handler receives a query string at network call 17 and returns an answer at network call 18. The AcmeSpans Agent then assembles the answer information using a set of rules developed by the AcmeSpans Company and sends the Query Answer back to the Client Software.

The *iEDA* agent system is decomposable and the activity of the BealTrusses Agent (Figure 6) is identical in structure to that generated by the AcmeSpans Agent. The resources utilize the serial number obtained from the query handler to know the components that comprise the specific Beal Truss part. The locations of these components are obtained from the registry and valid queries from the *iEDA* ontology. The BealTrusses agent uses its internal client to query these components from the AlliedBars and BessyBars Agents. These agents query handlers receive the network calls 2, 4, 10, 12, 6, and 14. The answer is returned in network calls 3, 5, 11, 13, 7, and 15 (Table 2 and Figure 6). The BealTrusses Agent, then assembles this information using a set of rules developed by the BealTrusses Company, and sends the subsystem answer back to the AcmeSpans Agent.

The BessyBars and AlliedBars Agents are identical in structure to any of the Truss or Span Agents, however they

Table 2. TCP/IP Network Message Calls

Network Call	Sender	Type	Content	Receiver
User Query	<i>iEDA</i> Client	Query	(SN, Cost)	AcmeSpans
1	AcmeSpans	Query	(SN, Cost)	BealTrusses
2	BealTrusses	Query	(SN, Cost)	AlliedBars
3	AlliedBars	Answer	\$24.24	BealTrusses
4	BealTrusses	Query	(SN, Cost)	AlliedBars
5	AlliedBars	Answer	\$24.24	BealTrusses
6	BealTrusses	Query	(SN, Cost)	BessyBars
7	BessyBars	Answer	\$172.68	BealTrusses
8	BealTrusses	Answer	\$226.16	AcmeSpans
9	AcmeSpans	Query	(SN, Cost)	BealTrusses
10	BealTrusses	Query	(SN, Cost)	AlliedBars
11	AlliedBars	Answer	\$24.24	BealTrusses
12	BealTrusses	Query	(SN, Cost)	AlliedBars
13	AlliedBars	Answer	\$24.24	BealTrusses
14	BealTrusses	Query	(SN, Cost)	BessyBars
15	BessyBars	Answer	\$172.68	BealTrusses
16	BealTrusses	Answer	\$226.16	AcmeSpans
17	AcmeSpans	Query	(SN, Cost)	AlliedBars
18	AlliedBars	Answer	\$24.24	AcmeSpans
Query Answer	AcmeSpans	Answer	\$465.40	<i>iEDA</i> Client

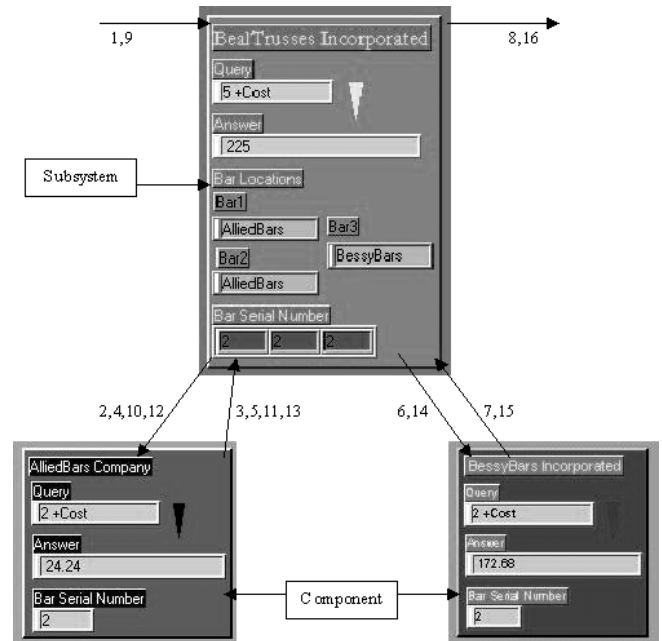


Figure 6. BealTrusses agent interaction for SA3x6 Cost solution

have no components because they are at the component level. These Agents through their resources know the cost of the serial number bar component. They request service from their resources. Through their internal set of rules, BessyBars and AlliedBars Agents generate a component answer and this information to the appropriate subsystem, system, or client software.

CONCLUSIONS

The *iEDA* approach illustrated here differs substantially from current modeling approaches. Our agent methodology focuses on facilitating interaction between virtual components and subsystems rather than facilitating interaction between computer applications. The decomposable nature of the *iEDA* system allows for orderly, distributed model growth *via* compositional modeling. The standard ontology based communication protocol allows for proprietary security. The viable set of queries represents the services provided by the agent, and hence the corporation. The approach is system based and supports collaborative engineering from a systems perspective.

The work presented by this paper is an initial step toward a useful and leveraged collaborative engineering design tool. Insight has been gained into the evolution of an *iEDA* network through the implementation of the simple prototype example.

This prototype, although limited, demonstrates a bare-bones implementation of an *iEDA* system.

The topology of the *iEDA* is not complete. As the prototype shows, memory is an essential aspect needed to be included in the resources of the *iEDA*. The topology augmented with memory, closely resembles a basic knowledge-based system structure (Dym, Levitt, 1991). The current system is limited to simplistic model building structures for performance attributes such as cost, size, weight, etc. Current work has is extending these models to static structural performance and dynamic response strategies are also in development. All these *iEDA* performance models must function in the distributed agent architecture discussed here while minimizing system bandwidth. We will present these new systems as they are developed in future work. The knowledge gained from this work will help extend capabilities for globally distributed engineering.

ACKNOWLEDGEMENT:

Much of the early implementation of the *iEDA* agents that provided the material for this paper was done by Mr. Gary Gosciak and documented in his M.S. Thesis "Internet Engineering Design Agents", May 2001.

REFERENCES

AEI, 2000a, "Redesigning Work Processes And Computing Environments", Automotive Engineering International, SAE International, Brimfield, OH, July, pp. 147-149.

- AEI, 2000b, "Behind GM's Global Design Success", Automotive Engineering International, SAE International, Brimfield, OH, December, pp. 74-75.
- Bokulich, F., ed., AEI 1999, "CAD software integration", Automotive Engineering International, SAE International, Brimfield, OH, April, pp. 52.
- Bradshaw, J., ed., 1997, Software Agents, AAAI Press, Menlo Park, CA., ISBN 0-262-52234-9, pp. 1-46.
- Brenner, W., Zarnekow, R., and Wittig, H, 1998, Intelligent Software Agents Foundations and Applications, Springer-Verlag, Berlin Heidelberg New York, ISBN 3-540-63411-8, pp. 1-86.
- Bucholz, K., ed., AEI 1998, "SAE Global Vehicle Development Conference", Automotive Engineering International, SAE International, Brimfield, OH, November, pp. 105.
- Byam, B. P. and Radcliffe, C. J., 1999, Nov. 14-19, Nashville, Tenn., "Modular Modeling of Engineering Systems Using Fixed Input-Output Structure", Proc. of the ASME Dynamic Systems and Control Division, 1999 ASME Intl. Mech. Engin. Conf and Exposition, DSC-Vol. 67, ISBN 0-7918-1634-6, pp. 545-551.
- Byam, B. P. and Radcliffe, C. J., 2000, Sept. 10-13, Baltimore, Maryland, "Direct Insertion Realization of Linear Modular Models of Engineering Systems Using A Fixed Input-Output Structure", Proc. ASME Design Engineering Technical Conferences, 26th Design Automation Conference, DETC2000/DAC-14236, ASME
- Dorf, R. C. and Kusiak, A., ed., 1994, Handbook of Design, Manufacturing and Automation, John Wiley & Sons, New York, ISBN 0-471-55218-6, pp. 25-33, 977-990.
- Dym, C. L. and Levitt, R. E., 1991, Knowledge Based Systems in Engineering, McGraw-Hill, Inc., ISBN 0-07-018563-8, pp. 1-80.
- Esterman, M. and Ishii, K., 1999, Sept.12-15, Las Vegas, NV, "Challenges in Robust Concurrent Product Development Across the Supply Chain", Proc. ASME Design Engineering Technical Conferences, DETC99/DFM-8920, ASME
- Etzioni, O. a. W., D. (1994). "A Softbot-Based Interface to the Internet." Communications of the ACM 37(7): 72-76.
- Franklin, S. a. G., A. (1996). "Is it an Agent, or just a Program: A Taxonomy for Autonomous Agents", Third Int'l Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag.
- Gilbert, D. (1997). "Intelligent Agents: The Right Information at the Right Time", IBM Corporation, <http://www.networking.ibm.com/iag/iaghome.html>
- Murch, R., Johnson, T., 1999, Intelligent Software Agents, Prentice Hall, Inc., Upper Saddle River, New Jersey, ISBN 0-13-011021, pp. 1-45.
- Nwana, H. S. (1996). "Software Agents: An Overview." The Knowledge Engineering Review 11(3): 205-244.
- Rosenman, M. and Fujun W., 1999, "CADOM: A Component Agent-based Design-Orientated Model for Collaborative Design", Springer-Verlag London Limited, Research in Engineering Design, pp. 193-205.
- Shoham, Y. (1997). An Overview of Agent-Oriented Programming. Software Agents. J. M. Bradshaw. Menlo Park, CA, AAAI Press: 271-290.
- Sycara, K., Decker, K., Pannu, A., Williamson, M. and Zeng, D. (1996). "Distributed Intelligent Agents." IEEE Expert(Dec).
- Tasso, C., Oliveira, E., ed., 1998, Development of Knowledge-Based Systems For Engineering, Springer-Verlag, Wien New York, ISBN 3-211-82916-4, pp. 11-35, 201-209.
- Tecuci, G., 1998, Building Intelligent Agents, Academic Press, San Diego, Cal., ISBN 0-12-685125-5, pp. 1-12.
- Wallace, D., Borland, N., Senin, N., and Abrahamson, S., 1999, Sept. 12-15, Las Vegas, NV, "Integrated Engineering, Geometric, And Customer Modeling: LCD Projector Design Case Study," Proc. of the ASME Design Engineering Technical Conferences, DETC99/CIE-9084, ASME
- Wooldridge, M. a. J., N. R. (1995). "Intelligent Agents: Theory and Practice." The Knowledge Engineering Review 10(2): 115-152.
- Wooldridge, M. and Jennings, N. R., 1998, "Pitfalls of Agent-Orientated Development," Proceedings of the Second International Conference on Autonomous Agents, pp. 385-391.
- Wooldridge, M. and Jennings, N. R., ed., 1998, Agent Technology Foundations, Applications, and Market, Springer-Verlag, Berlin Heidelberg New York, ISBN 3-540-63591-2, pp. 1-4