Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2008), *22*, 1–18. Printed in the USA.

DOI: 10.1017/S0890060408000012

The routine design–modular distributed modeling platform for distributed routine design and simulation-based testing of distributed assemblies

M. TANER ESKIL,[1] JON STICKLEN,[2] AND CLARK RADCLIFFE[3]

[1]Computer Science and Engineering Department, Faculty of Engineering, Işık University, Istanbul, Turkey
[2]Intelligent Systems Laboratory, Michigan State University, East Lansing, Michigan
[3]Dynamic Systems Laboratory, Michigan State University, East Lansing, Michigan

(RECEIVED May 6, 2005; ACCEPTED February 1, 2007)

Abstract

In this paper we describe a conceptual framework and implementation of a tool that supports task-directed, distributed routine design (RD) augmented with simulation-based design testing. In our research, we leverage the modular distributed modeling (MDM) methodology to simulate the interaction of design components in an assembly. The major improvement we have made in the RD methodology is to extend it with the capabilities of incorporating remotely represented off-the-shelf components in design and simulation-based testing of a distributed assembly. The deliverable of our research is the RD-MDM platform, which is capable of automatically selecting intellectually protected off the shelf design components over the Internet, integrating these components in an assembly, running simulations for design testing, and publishing the approved design without disclosing the proprietary information.

Keywords: Distributed Simulation; Modular Modeling; Off-the-Shelf Parts; Proprietary Information; Routine Design

1. INTRODUCTION

During the initial stages of a design process, designers typically seek to obtain broad and general knowledge to understand and analyze the problem, to produce a conceptual design, to select materials, and to specify manufacturing processes (Nowack, 1997). In the later stages, which are nontrivial and in general nonlinear, the knowledge gathered and organized will be used to test the simulated performance of candidate solutions (Rodgers et al., 1999). Both the conceptual design and analysis stages need a large knowledge base to be brought together and developed by the design team. The successful development of a design, and consequently, the competitiveness of a company, depends on acquiring and organizing relevant knowledge in a timely manner (Smith & Reinertson, 1991; Court et al., 1997).

Developing the necessary knowledge base, which consists of searching for and locating the required knowledge, has been estimated to take 30–40% of the design time (Cave & Noble, 1986; Marsh, 1997). Rodgers' (1997) research on the knowledge requirements of design teams in a telecommunications company in the United Kingdom revealed that the initial point of contact for the design teams is a file 40% of the time, and external personnel more than 25% of the time. "File" in the context of this research refers to a mostly electronic-based, internal, or external document. External documents include the International Standards Organisation (ISO) documents and contractors' design knowledge.

Availability of reliable, high-speed electronic connectivity enables designers turn to the Internet when knowledge is stored externally, whether the assistance they seek is design knowledge (e.g., CAD drawings) or design expertise (e.g., an expert contacted by e-mail). One of the most profound ways the Internet is affecting engineering design is by allowing companies to be more externally focused. The Internet has enabled design teams to function irrespective of the location of the knowledge, by rapid part ordering from electronic catalogs, use of distributed design and simulation tools, and so forth. On the consumers' side, the Internet allows companies to drive consumers' demands into the design process quickly, which in turn, increases the commercial agility of the company. On the supplier side, as products become more complex, 60–80% of parts are outsourced. For instance,

Reprint requests to: M. Taner Eskil, Computer Science and Engineering Department, AMF 336, Faculty of Engineering, Işık University, Şile, 34980 Istanbul, Turkey. E-mail: eskil@isikun.edu.tr

in the year 2000, Daimler/Chrysler outsourced 65–75% of each manufactured car (Ames, 2000). A closer collaboration between a company, its customers, and suppliers has the potential to bring more innovative products to market and to do so in economically viable ways.

Online design collaboration between suppliers and manufacturers is likely to increase as Internet tools become more supportive and secure. To utilize the Internet for information gathering and modeling in the design process in a secure way, a new generation of supporting tools and methodologies will be required. There are a number of studies undertaken in this area, focusing on how designers find, access, retrieve, and manage knowledge throughout the design process (Dong & Agogino, 1996; Marsh, 1997; Schott et al., 1997; Keskinocak et al., 2001; Zhang et al., 2003).

The research on online design collaboration has already generated a relatively secure environment for designers to share their design knowledge over the Internet with suppliers and manufacturers under strict legal contracts. On the surface, this competitive change should result in lower pricing and higher quality products. However, to protect intellectual property, suppliers are reluctant to share the designs of their products with "window shoppers" without strong and enforceable legal agreements. Putting in place such legal agreements (typically, *proprietary information* or *nondisclosure* agreements) requires both money and, more importantly, time. It is not uncommon for such agreements to take between 6 months and 1 year to put into place. This delay lengthens the time to market, hampering the search for lower price and higher quality products, and slowing down the response to changes in the market.

The difficulties in design collaboration do not end with the realization of nondisclosure agreements between a manufacturer and suppliers. On the contrary, the constant evolution of the market makes it intractable to keep an error-free database of models. A good example of this is from the automotive industry in which each manufactured vehicle consists up to 15,000 parts and many of these may be supplied by different vendors. Daimler Chrysler's experience with the product data management system dialog has shown that keeping a database of thousands of model constraints defect free is extremely difficult, if not impossible, because of constant change in part models (Sinz et al., 2003).

The challenges discussed above recur in the design testing stage in which running simulations for verification of the design functionality is desirable. Furthermore, the iterative nature of time-sliced simulations brings in another critical challenge in the case of distributed simulations; keeping the network traffic in the limits of its bandwidth. This problem could appear at first consideration to be simply an exercise in distributed object-oriented simulation. However, such a position would be incorrect because it does not take into consideration the reality of the Internet world.

The common currency of the simulation world, the deciding factor between any two simulation approaches that both yield accurate results, has been the speed a simulation approach engenders. Moreover, because earlier simulation approaches have dealt the "all codes available" situation, this speed factor has been pushed down to the speed with which a single computer (or a tightly coupled cluster) can complete a simulation. In other words, if a given simulation is slower than desired at completing a simulation, and no other approaches are available, then the solution has always been to obtain a more powerful computer.

However, in relation to the Internet-based simulations this view is misguided and misleading. The limiting factor on simulation time in a world in which models are distributed across the Internet is the Internet transmission time of information necessary to support the simulation. Shortening this time could be approached in one of two fundamental ways: decrease Internet packet transmission times or lower the number of Internet packets needed to transfer the needed information that drives the simulation. The first path is not practically feasible as a part of any single project such as ours. Moreover, the number of packets to be transferred in a traditional simulation problem of even moderate complexity such as dynamic response of a mechanical system (e.g., a bridge structure) would produce such a large number of packets as to bring the speed of a distributed simulation to its knees. The second path thus is the only feasible solution in the current wired world.

In this paper we introduce routine design–modular distributed modeling (RD-MDM) platform, an integrated distributed design and simulation environment, which effectively deals with these impediments on online design collaboration. RD-MDM enables system integrators to rapidly design their products and perform simulation-based design testing using intellectually protected computational models that are distributed over the Internet. We handle the challenges mentioned above by utilizing the MDM methodology that was introduced by Byam and Radcliffe (1999), and later employed by the Internet-Engineering Design Agents group at Michigan State University (Eskil et al., 2003; Reichenbach, 2003). As we will detail in Section 3, the crux of the MDM methodology is to share input–output models of engineering artifacts without disclosing their internal connections or dynamics, hence protecting the proprietary information. The MDM models are kept at the supplier side to ensure up-to-date information, and the number of transmitted packages during an online simulation is reduced to *one* for each MDM model embedded in a design.

In the big picture, RD-MDM is a cooperative engineering design tool for distributed black-box modeling and simulation of engineered artifacts in an open, competitive e-commerce. With our approach, vendors will be able to make their core models available to the public without disclosing proprietary information such as the internal architecture or utilized components. Designers, contrarily, will be able to automatically search and incorporate these models as off-the-shelf parts into their designs and simulate them as integrated components of the virtually distributed assembly.

This paper details our approach and presents the results we obtained with the proposed technique. Section 2 presents the

https://doi.org/10.1017/S0890060408000012

related studies and identifies the challenges in distributed agent-based design. Sections 3 and 4 describe the MDM technique and the RD methodology. Section 5 details the extensions we have made to both approaches and introduces the integrated RD-MDM platform. Section 6 provides an example to illustrate the problem solving process with RD-MDM. Section 7 summarizes the outcomes of this research, and Section 8 points out the limitations and possible future directions.

2. RELATED STUDIES

Early distributed problem-solving approaches assumed that knowledge pertaining to the problem domain could be gathered and represented on a network of closely bound computers, in compliance with a particular architecture. As the problem gets more complex, gathering and organizing the widespread expert knowledge becomes impractical. An example is the engineering design problems, for which the information is widespread, unorganized, and in general beyond the reach of a single designer (Alexander, 1964). MacGregor and Thomson (2001) also emphasized the lack of common terminology between teams of expertise and unawareness of existence of knowledge. In the literature there have been two approaches to solving the common terminology problem: standardizing the representation of knowledge (Augenbroe, 1995; Fruchter et al., 1995) and encapsulating the knowledge in data wrappers and enabling communication through a predetermined terminology (Cutkosky et al., 1993; Maturana & Norrie, 1996; Shen & Barthes, 1996; Chan et al., 1998; Maturana et al., 1999). We will make a brief review of these approaches below.

The most notable examples of model standardization frontier are STEP of ISO and Integrated Data Model of COMBINE (Augenbroe, 1995). STEP is oriented toward exchanging the geometric model of the data, but not the function of the model (Kopena & Regli, 2003). To incorporate the designer's ideas about the model, Fruchter et al. (1995) built Interdisciplinary Communication Medium, using the propose/interpret/critique/explain paradigm in a cycle of collaborative design. Ball et al. (1998) and Fruchter et al. (1995, 1996) developed prototypes that capture design rationale. Rosenman and Gero (1996) proposed a paradigm to describe the semantics of the model, known as purpose–function–behavior structure.

Standardization of model representation requires massive conversions from legacy design development platforms, and its success heavily depends on its widespread acceptance. The cost of interoperability efforts in the United States is estimated as $2 billion per year (Schlenoff et al., 2000). Wallace et al. (2001) state that the money spent yearly on the integration work in Ford Motor Company alone is in the order of hundreds of million dollars. A comprehensive conversion to standardized representation ontology would indeed discontinue such expenses, but it would be extremely expensive as these figures illustrate. This fact encouraged many researchers to develop ways to encapsulate knowledge in modular units called *agents* and enable communication between agents over a predetermined ontology.

Wooldridge (1997) defines an agent as "an encapsulated computer system that is situated in some environment and can act flexibly and autonomously in that environment to meet its design objectives." Jennings and Bossmann (2003) elaborate on this definition and states the distinguishing characteristics of agents: agents are problem-solving entities with clearly defined boundaries and interfaces. They have partial knowledge and control in their surroundings. They have objectives, and they exhibit autonomous behavior in pursuit of their goals. Objects, in contrast, are passive in nature. They are suitable for encapsulating information but they lack autonomous behavior.

PACT (Cutkosky et al., 1993) is one of the most well-known projects that advocate encapsulation of tool data and model representations rather than standardizing them. In PACT, each tool uses the most appropriate internal data structures and representation of models and communicates with languages of varying complexities. To support the complicated nature of communication between PACT agents, a *facilitator* mechanism is implemented. The facilitator provides an interface between a local network of agents and remote agents. Its responsibilities include routing and translating messages and monitoring the local problem-solving process. The collection of autonomous agents under facilitators is called a *federation architecture*.

MetaMorph (Maturana et al., 1999) is another project that uses the federation architecture to integrate distributed intelligent systems and concurrent engineering tools. In this architecture, the coordination of virtual groups of intelligent agents is realized by the *mediator* mechanism (Maturana & Norrie, 1996). The DIDE project (Shen & Barthes, 1996) proposes asynchronous cognitive agents for integrating design and engineering tools and human specialists. In DIDE, agents work independently during the design process, but their results are generally monitored by a human agent before they are broadcast to the community. WELD (Chan et al., 1998) makes use of the encapsulation strategy to enable complete distribution of users, tools, and services. This is accomplished by treating every component as potentially mobile. For uniform and reliable communication, WELD avoids any coupling between components. Its communication protocols are built on generic string-based messages for extendibility.

The approaches mentioned above primarily focus on collaboration in engineering design and fail to address simulation of distributed assemblies or protection of proprietary resources. Gu et al. (2002) provide a discussion of the need for protecting proprietary information in engineering design and analysis and propose an encapsulation-based method for simulation of coupled systems with minimum information disclosure. With this method the initial state and implementation details of component models are not required, but the protection of proprietary information is not guaranteed either. Gu et al. also draw attention to the need of integrating

powerful and adaptive methods for design parameterization in simulation architectures.

Silva and Katz (1995) and Hauck and Knoll (1998) proposed cryptographic techniques for transferring simulation models of components without revealing proprietary information. These cryptographic techniques are simulator dependent and require manufacturers to maintain a different database of appropriate simulation models for each technique. Moreover, they require considerable effort to update transferred representations when the core model is changed. Helaihi and Olukotun (1997), Dalpasso et al. (1999, 2002), and Fin et al. (2000) addressed these issues and proposed simulation to take place on the manufacturer site by use of *ad hoc* languages to model the functionality of the component. These approaches offer a solution to the protection of proprietary information at a cost of extra work and possible discrepancies between the design and component models. Another disadvantage is the increased complexity of assembling a design that incorporates components from different vendors.

The DOME project (Pahng et al., 1998; Abrahamson et al., 2000) deals with the protection of the proprietary data by encapsulating services in wrappers, thus creating a modeling infrastructure for individuals to share their simulation services related to their expertise. The ultimate goal is to allow individuals to design and understand complex systems by use of latest modeling technology offered by experts. The infrastructure serves as an interface for the modeling tool once it is published on the DOME server. This research aims more or less toward our goal: allowing individuals to design, simulate, and understand complex systems. In the DOME approach this is realized by making design and simulation tools available to individuals on the Internet, whereas our focus is on sharing device models. Abrahamson et al. (2000) point out the growing need for developing shared ontologies to be used with the DOME architecture.

Although the state-of-the-art approaches prove to be valuable search and decision tools, they provide very limited capability in automated design and analysis in the context of open e-commerce. Shakeri and Brown (2004) point out the need for resource sharing across disciplines and provide a new knowledge-based methodology for simulation of a design process. Spiller and Newton (1997) envision the future of the Engineering Design and Analysis community organized in an integrated and distributed environment. Interoperability between tools and design libraries will create an evolvable, customizable, and adaptable virtual design network. Such an organization would also enable querying products and serve as a virtual consultant to researchers and individuals. Regli (1997) emphasizes the importance of online smart catalogs supported with intelligent agents that can also filter relevant information. Such computer-interpretable information models augmented with the issues of security and trust can be integrated with existing tools and services to develop entirely automated and distributed design platforms. In contrast, as Regli draws attention to this, advances in distributed design bring about the problem of handling gigabytes of information flow over slow World Wide Web protocols.

Today, the prominent challenges in distributed agent-based design are the following:

1. difficulty of defining a complete product model because of the model complexity or size (Cera et al., 2004);
2. insufficiency of design tools that address design across different engineering domains (Shakeri & Brown, 2004);
3. unavailability of compatible models across different engineering domains (Cera et al., 2004), even between different manufacturers within the same engineering domain (Szykman et al., 2001);
4. unavailability of knowledge on design products for proprietary reasons (Cera et al., 2004);
5. keeping network traffic within manageable levels during online collaboration efforts (Regli, 1997); and
6. unavailability of tools that support both distributed design and simulation of distributed assemblies for design testing.

3. DISTRIBUTED COMPUTATIONAL MODELS: THE MDM PLATFORM

The Internet-Engineering Design Agents group of Michigan State University successfully developed an infrastructure capable of supporting a community of MDM agents (Eskil et al., 2003; Radcliffe & Sticklen, 2003; Reichenbach, 2003; Eskil, 2004). The ensemble of MDM agents are capable of supporting Internet-based cooperative engineering design and simulation, but with the constraint of device knowledge hiding. The conceptual building block to enable MDM is a fixed but autonomous MDM agent. An MDM community is

1. an Internet-based distribution of fixed MDM agents where
2. each MDM agent holds knowledge of a single manufactured device (or a family of similar manufactured devices) and
3. each MDM agent is an instance of one member of a typology of MDM agents.
4. Each query that an MDM agent receives is drawn from a known and well-defined typology of possible queries.
5. An MDM agent will respond to queries using a set answer syntax that is implied by the specific query.
6. Each query that an MDM agent will receive is focused either
   a. on a *quality* of the device held by the MDM agent or
   b. on a *functional response* of the device held by the MDM agent.
7. Each MDM agent contains the knowledge and computational resources to effectively answer those valid queries that it chooses to answer.

https://doi.org/10.1017/S0890060408000012

8. Each MDM agent composes answers to queries such that implementation details of the device it holds are *not* revealed.
9. Any MDM agent may hold a device that internally includes parts (or assemblies) held by other MDM agents.

Capabilities 1 and 2 express the need to "publish" to the Internet an MDM agent that acts, according to capability 8, as a buffer between the world and internally held device knowledge. The MDM agent will allow the world to know what its device is functionally capable of, and what the qualities of its device are, but that is all. Capability 3 is a statement that MDM agents are of known types. An example of an MDM agent type would be "mechanical structures."

Capabilities 4 and 5 express the need for structured communication. To conceive, design, and implement the infrastructure for the Internet community for cooperative design in an e-commerce setting, and equally importantly, to gain acceptance within the manufacturing and engineering communities, communication of MDM agents must be demonstrably fail-safe, and to those ends structured interagent communication is essential.

Capability 6 sets in broad terms the semantics of MDM agent communication. Subcapabilities 6a and 6b set the two broad areas of queries that *can* be accepted by and MDM agent. *Quality* refers to an inherent property of an MDM agent's device. Examples include weight and size. *Functional response* refers to the manner in which the MDM agent's device will interact with the world when supplied external inputs of specified type. Examples include static mechanical response to applied loads and dynamic mechanical response to applied forces. Capability 6a refers to one-shot queries and responses on a quality of the MDM agent. To achieve capability 6b in a feasible manner in the Internet environment, we will utilize the novel approach that was introduced by Byam and Radcliffe (1999, 2000) and detailed in Section 3.1.

Capability 7 sets the goal that MDM agents will be able to answer legal queries that it receives, where what is legal is covered in capability 6. There is an important but easily missed point here. The queries that an MDM agent will answer are a subset, and *possibly* a proper subset, of the queries that would be legal for a given MDM agent. The reason for this point is that although a manufacturer may wish to publish an MDM agent for a device offered for sale, the manufacturer may want to hold some information that would normally be available for the device. This flexibility provides members participating in an MDM agent network the ability to make information available to potential buyers in a selective manner.

Capability 9 sets the need for MDM agents to allow a recursive structure that mirrors engineering device/subdevice decompositions. For example, an automobile is composed of drive train system, suspension system, and so forth. A drive train system is further composed of transmission system, differential system, and so forth. Hierarchical decomposition is the means by which engineers (and engineering as a broad field) handle the complexity of modern manufactured devices. In design, engineers typically try to use off-the-shelf parts and subassemblies when designing new (or improved) devices. However, each of those off-the-shelf items may also be composed of other off-the-shelf items. Enabling engineers who are working within an MDM community to quickly incorporate into a new design off-the-shelf parts offered by other members of the MDM community, and to analyze a new design by running simulations is the bedrock motivation for our research.

The flow of information inside an MDM agent is demonstrated in Figure 1. As addressed by capability 9, an MDM agent (M) may be a component itself or composed of remote components assembled through the action of Assembler (A) and Join (J) constraints. The subcomponents likewise may be atomic—a part without subcomponents—or assembly. They can also be local (C)—a part whose representation is readily available on the local computer—or remote MDM agents (M). Conceptually, parts/assemblies from other MDM agents can be made a component of a given MDM agent (e.g., a fuel injector becomes a component of a Fuel System) but *only* device qualities and functional responses are known to other agents. It is *not* that one agent device model contains or has access to full descriptions of the devices held by other agents.

The sketch shown in Figure 2 depicts three MDM agents connected physically via the Internet. It is important to note that in response to queries, MDM agents would not reveal an internal virtual linkage; such virtual linkages can be used to express an internal structure that includes parts/assemblies of other MDM agents. Two additional MDM resources can be seen in Figure 2: *Query Ontology* and *Agent Registry*. Referring back to the list of MDM agent characteristics enumerated at the start of this section, specifically characteristic 4, the query ontology is an MDM network resource that makes available to any MDM agent the typology of legal queries. The Agent Registry meets characteristic 3, and is an MDM network resource that makes available to any MDM agent both the agent types and the list of all existing MDM agents. Of particular importance in Figure 2 are the two types of connections between MDM agents. Both the physical connection via the Internet and the linkage that represents device hierarchy relationships are sketched.

The MDM approach asserts that how a specific product is designed and its design details does not have any relevance to how the product integrates structurally and functionally in a larger design. Therefore, a distributed model can reside on any platform of its designer's convenience and communicate by string passing on a predetermined ontology that specifies the query and response formats while hiding the proprietary knowledge.

MDM offers a unique platform to easily locate and integrate a product as a part of an assembly while eliminating the proprietary data concerns. Most other techniques such as Universal Description, Discovery, and Integration Protocol

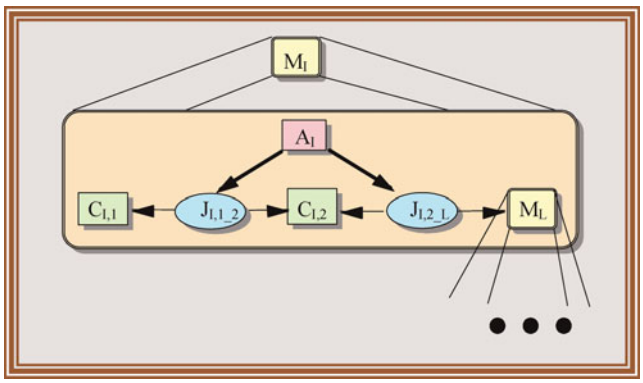

Fig. 1. A sketch of the information exchange topology. [A color version of this figure can be viewed online at www.journals.cambridge.org]

(www.UDDI.org) focus only on locating the right product or service and connecting its provider with the receiver. RD-MDM has the characteristic of not only searching but also integrating multiple components in an assembly and running simulations on the virtually assembled product. We will discuss these capabilities further in Sections 4 and 5.

Bandwidth is a major concern in distributed simulations, especially when the simulation is iterative in nature. Functional response models (FRMs; Byam & Radcliffe, 2000) overcome the bandwidth problem by reducing the required number of communications between each pair of components to *one* for the entire simulation. Instead of many thousands of iterations between a requesting MDM agent and the responding MDM agent, FRM technique enables *one response* to be made by the responder. The requesting MDM agent is then able to use the single functional response answer as the basis for local (to it) simulation that incorporates the device of the responding MDM agent into its own (the requesting MDM agent) device assembly.

3.1. Functional response modeling

When an MDM agent is queried, it may return a one-shot response (e.g., price, delivery time, color) or an FRM of the queried attribute. FRMs are critical in development of a methodology and computational mechanism that allows us to achieve *functional response* of the device held by the MDM agent. Some discussion here is necessary before we move on to the integration of the MDM architecture with the RD methodology.

We start with an observation: traditional methods of device simulation will *not* be effective over the Internet. Even very sophisticated simulation methods relying on object-oriented design currently still entail a large amount of communication between various components of a simulation object. In a tightly connected, high-speed environment, this may be effective. In the current Internet environment it will not be effective because of the enormous load on network traffic that would be entailed. Consider, for example, a state space simulation where multiple communications between derivative functions are required for each time step of the simulation. The situation for state space simulation is not unusual in terms of required communication load. Thus, a major

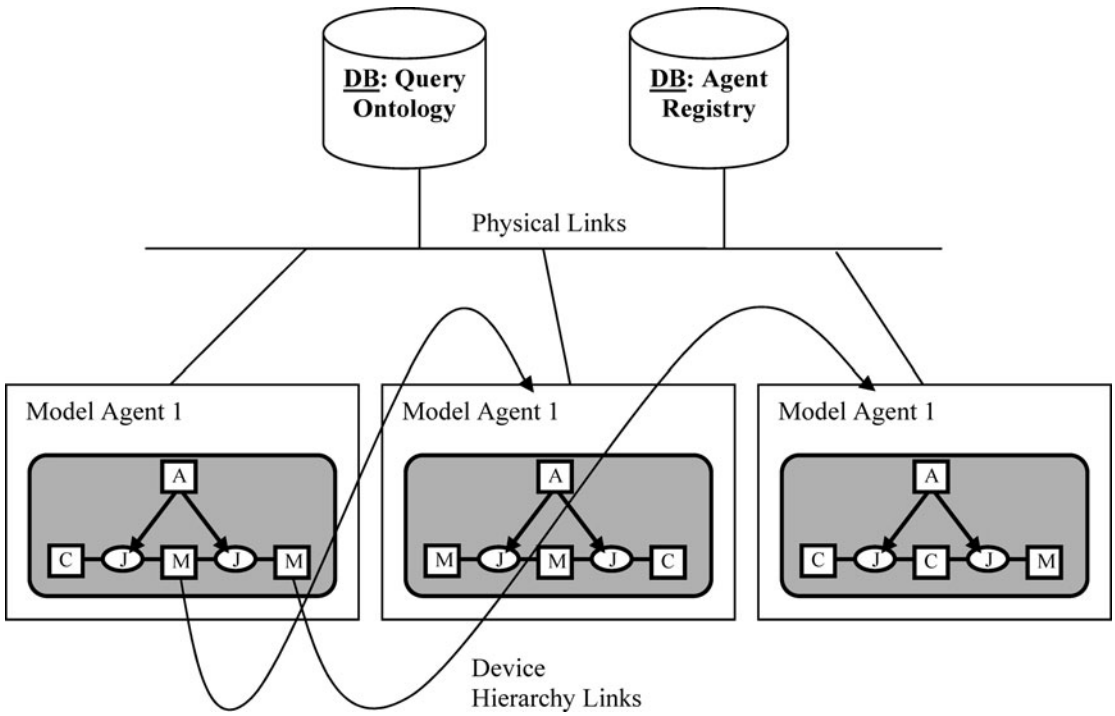

Fig. 2. A sketch of the modular distributed modeling agent community.

hurdle we had to overcome was to conceptualize and implement as proof of principle a new way to carry out simulation of engineered devices that would minimize Internet traffic.

We have surmounted that hurdle by conceptualizing a type of simulation output that for MDM agents is in response to questions asking for a *functional response* answer. Instead of many thousands of iterations between a requesting MDM agent, and the responding MDM agent, our output form enables *one response* to be made by the responder. The requesting MDM agent is then able to use the single functional response answer as the basis for local (to it) simulation that incorporates the device of the responding MDM agent into its own (the requesting MDM agent) device assembly. This is the core result that will enable MDM communities in an Internet environment.

Next, we will discuss one specific type of functional response: functional response answers to queries of mechanical structures in the domain of structural analysis. Although the example below is distinctly in the realm of mechanical engineering, it is centrally important to remember that our entire approach depends on the ability to both hide proprietary simulation models, while at the same time incorporating the device represented by its MDM agent into a higher level simulation model. The example sketched below is a proof of principle result that developing functional response methodology is feasible explicitly in the structural analysis domain.

Figure 3 shows a diagram of a contracted model for an Internet design agent model. The component's algebraic model is in the standard stiffness form. Modular modeling element graphical notation represents user-defined multiport, multiple degree of freedom subsystem models with a rectangle. The bold lines represent the power ports with implicit standardized direction of positive power into the element and enforce a standardized input–output port causality. Standardization of positive power direction and input–output causality standardizes the modular modeling elements' internal formulation, which is the essence of modular modeling.

In this example, the detailed physical response model of a component is in the standard stiffness form

$$\mathbf{K}\mathbf{y} = \mathbf{u}, \tag{1}$$

where **K** is the component stiffness matrix, **y** is the component generalized displacement vector, and **u** is the component

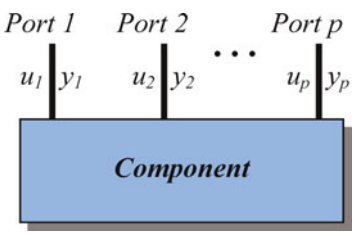

Fig. 3. Modular modeling element graphical notation. [A color version of this figure can be viewed online at www.journals.cambridge.org]

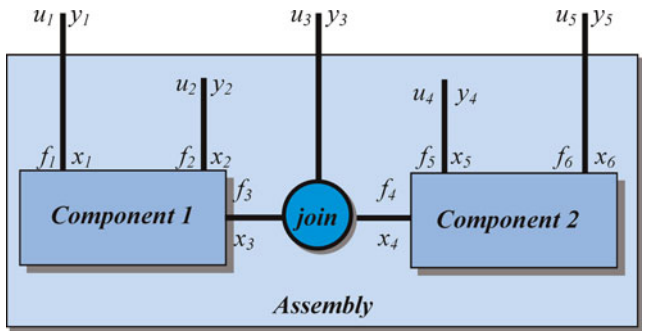

Fig. 4. The subsystem model with two components. [A color version of this figure can be viewed online at www.journals.cambridge.org]

input vector. In general, component stiffness matrix is singular and cannot be inverted. This situation occurs because component models have zero eigenvalues from "rigid-body modes" representing components with no applied boundary conditions. An example for this situation is an unconnected structural element, such as a beam that is free to translate in any direction.

Modular modeling connector constraints implement standard output and power constraints. The connector constraint forces two connected modular element ports' outputs, *i* and *j*, to be equal and their power flow to sum to zero to conserve power. The power flow constraint is translated to equal and opposite inputs at connected modular element ports because the product of power port variables is power.

The modular modeling connector constraint graphical notation represents connectors with a bold port line between modular elements as shown in Figure 4. By definition, connectors have the compatible input–output structure to modular elements. The bold lines have implicit standardized direction of positive power into connected modular element ports *i* and *j* and modular connector constraints. The modular connector has the flexibility to assemble by pairs any number of modular element power ports because modular modeling elements have an internal junction structure at each port. The only function of modular modeling connectors is to constrain connected modular element ports.

3.2. Deriving FRMs for assemblies using the Join procedure

The subsystem model depicted in Figure 4 is a demonstration of the possible situations that can arise when components are assembled into subsystems. It has two components connected via constraints on port variables on ports 3 and 4. It has internal component ports 2 and 5 that are not connected externally. Finally, the assembly has ports 1 and 6 that can be connected externally. Once assembled, a new algebraic equation set in the form of Eq. (1) is required so that this system can be used in higher level system models. Because the component models are often singular, the subsystem model will also be singular in general. Only when assembled with sufficient

boundary constraints do models become nonsingular and solvable.

For deriving the FRM for the assembly, the equations for each of the components are first assembled into the unconstrained system matrix, which is the unassembled dynamic inverse simulation model for the assembly.

$$\begin{bmatrix} \mathbf{K}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}, \tag{2}$$

or in the expanded form,

$$\begin{bmatrix} \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} & [\mathbf{0}] \\ [\mathbf{0}] & \begin{bmatrix} k_{44} & k_{45} & k_{46} \\ k_{54} & k_{55} & k_{56} \\ k_{64} & k_{65} & k_{66} \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \\ \begin{bmatrix} x_4 \\ x_5 \\ x_6 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \\ \begin{bmatrix} f_4 \\ f_5 \\ f_6 \end{bmatrix} \end{bmatrix}. \tag{3}$$

The subsystem components are uncoupled in this form. The modular matrix assembly equations transform the component input–output pairs ($\mathbf{f}_i$, $\mathbf{x}_i$) to assembly input–output pairs ($\mathbf{u}_i$, $\mathbf{y}_i$). The *Join* output constraint defines each component's output vectors ($\tilde{\mathbf{x}}$) in terms of the assembly output vector ($\mathbf{y}$).

$$\tilde{\mathbf{x}} = \mathbf{S}\mathbf{y}. \tag{4}$$

The power constraint on the assembly requires the sum of the work into all joined component ports $W_i = x_i f_i$ to equal the applied work $\tilde{W}_j = y_j u_j$. The causality in energy domains is defined such that this holds for every physical system in these domains.

$$\tilde{\mathbf{x}}^T \tilde{\mathbf{f}} = \mathbf{y}^T \mathbf{u}. \tag{5}$$

Applying the power constraint on (4) results in the modular *Join* input constraint between assembly's component input vectors $\tilde{\mathbf{f}}$ and assembly input vector $\mathbf{u}$ for all nonzero assembly outputs $\mathbf{y}$.

$$\mathbf{S}^T \tilde{\mathbf{f}} = \mathbf{u}. \tag{6}$$

The results we obtained in Eqs. (1)–(6) are all that is needed to derive the FRM for the assembly. Model assembly starts with the unconstrained grouping of all component models into an unconstrained assembly model as shown in Eq. (2),

$$\tilde{\mathbf{K}}\tilde{\mathbf{x}} = \begin{bmatrix} [\mathbf{K}_1] & [\mathbf{0}] & [\mathbf{0}] & [\mathbf{0}] \\ [\mathbf{0}] & [\mathbf{K}_2] & [\mathbf{0}] & [\mathbf{0}] \\ [\mathbf{0}] & [\mathbf{0}] & \ddots & [\mathbf{0}] \\ [\mathbf{0}] & [\mathbf{0}] & [\mathbf{0}] & [\mathbf{K}_n] \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_n \end{bmatrix}. \tag{7}$$

To obtain a concise modular model, we apply the constraints through matrix operations. The assembly output constraint is applied by substituting constraint (4) into (7):

$$\tilde{\mathbf{K}}\mathbf{S}\mathbf{y} = \tilde{\mathbf{f}}. \tag{8}$$

Multiplying both sides with $\mathbf{S}^T$ and using (6) yields the constrained assembly internal stiffness model

$$\hat{\mathbf{K}}\mathbf{y} = \mathbf{u}, \tag{9}$$

where $\hat{\mathbf{K}} = [\mathbf{S}^T\tilde{\mathbf{K}}\mathbf{S}]$.

This simple system is the assembled dynamic inverse simulation model. It has constants from the original system contributing to an algebraic combination of addition subtraction, and multiplication. The particular form of this function is dependent on the topology of the subsystem and is nonlinear in the parameters. The original engineering data from which this model is developed is well protected from reverse engineering—one principle objective of this modeling system.

4. DESIGN ARCHITECTURE: RD

Quite often when an engineer designs similar artifacts over and over again, he/she achieves a grasp of the routine nature of the process and starts discovering effective ways to decompose the design process into smaller design problems. Although all possible situations that might occur in a design process may not be known beforehand, it is nevertheless possible to acquire an understanding of design choices and plans that specify the order of making the choices. RD is a procedure that aims to capture this "expert" knowledge of the designer and realize it in computer environment.

Experience has shown that design of a complex system can be effectively fulfilled by breaking the functionality of the overall system into simpler and sometimes existing subcomponents and developing an assembly plan to bring these subcomponents together. Particularly in the design of domestic consumer products such as household goods, furniture, and automobiles, the focus is not to produce a product from scratch, but to modify certain aspects of an available design in an attempt to improve performance. Typically in such design tasks the design knowledge is available, but an understanding of how exactly the design knowledge should be applied is lacking (Hubka & Eder, 1996; Rodgers et al., 1999).

For instance, the first step in RD of an automobile would be the hierarchical decomposition of the task of overall design until manageable complexities are achieved in each subtask. After the design knowledge is incorporated in the subtasks, it is up to the strategies of RD process to carry out necessary modifications and most importantly the assembly process methods to bring together the individual components into a complete automobile.

RD should not be considered as a design tool that is applicable only to simple design problems. A complete RD

https://doi.org/10.1017/S0890060408000012

structure could represent any complex design for which the domain knowledge is available and could offer numerous solutions to a design problem. RD is not applicable only to novel design problems in which the required knowledge about the design strategies or potential design components (attributes of component types in case of configuration design) are unavailable, or obtained during the design process.

In a series of studies over the past decade, the Intelligent Systems Laboratory of Michigan State University developed several computer-based tools for engineering design and analysis. Among these tools, generic task RD (GT-RD) architecture was first suggested and implemented by Brown and Chandrasekaran (1989) and developed later on in the Intelligent Systems Laboratory (Kamel & Sticklen, 1994), which will be discussed in the next two sections.

4.1. RD using the GT approach

The introduction of the GT approach (Chandrasekaran, 1988) lead Brown and Chandrasekaran (1989) to define RD methodology and its representation language, Design Specialists and Plans Language (DSPL). The GT-RD architecture is capable of solving complex design problems (Sticklen et al., 1992; Chandrasekaran & Johnson, 1993; Kamel & Sticklen, 1994; Lenz et al., 1996) when an abstract design plan is obtained from a domain expert and captured in the GT-RD representational language.

DSPL deals with complex design cases by breaking the functionality of the overall system into simpler and existing subcomponents and capturing assembly plans to bring these subcomponents together. Currently, this method is applicable only when all subcomponent models are available locally. Worldwide exposure of all competing suppliers on the MDM platform would lead to numerous better parts, offered at lower prices. Therefore, a successful incorporation of GT-RD into MDM has the potential to realize lower cost and higher quality solutions to complex design problems.

4.2. Procedure of GT-RD

GT-RD decomposes the design problem into a hierarchy of cooperating specialists, each responsible for a specific aspect of the overall design. Typically, lower level specialists in this hierarchy represent actual components of the design and are responsible for parameterizing the design with a suitable component or parameterizing a generic component. As RD proceeds higher in the specialist hierarchy, conceptual aspects of the design problem become more and more pronounced.

Each specialist in the decomposition structure is furnished with a set of design plans. These plans are specified by the designer during the structuring of the task-specific RD system. To choose an appropriate design plan, a specialist invokes the plan selector, which in turn, refers to the sponsors of each plan, as depicted in Figure 5. A plan sponsor matches the status of the design with the conditions for applicability of the plan and reports the level of suitability of the plan it represents. A plan is initiated only when it is evaluated as suitable by its plan sponsor and chosen by the plan selector.

Design plans consist of ordered instructions for assigning values to parameters of generic components in accordance with the design goals of the specialist. To fulfill its task, an initiated design plan may invoke other specialists, execute design tasks, and check constraints. In case the design of a subsystem fails, design critique at that abstraction level tries to determine the reason for the failure and invokes the redesigner to take corrective actions, which may result in selection of a new plan (Brown & Chandrasekaran, 1989).

When a specialist successfully completes the part of the design it is responsible from, it hands in the design parameters to its parent specialist, where all subdesigns from one lower level are merged into a higher level design and checked for constraints. The design of the artifact becomes more complete as the design process progresses up in the design abstraction hierarchy.

5. THE INTEGRATED RD-MDM PLATFORM

Referring back to the challenges in distributed agent-based design that were enumerated in Section 2, the first challenge of defining an inclusive model for complex designs is generally dealt with breaking the functionality of the overall system into simpler and sometimes existing subsystems and devising an assembly plan to bring these subsystems together. GT-RD platform implicitly achieves this functionality provided that design strategies, models of potential design components, or physical attributes of components to be parameterized, and assembly plans are known beforehand. It also addresses to the second problem of design across different domains by offering GTs (Section 4.1), each of which captures the knowledge and control strategies that are characteristic to its type.

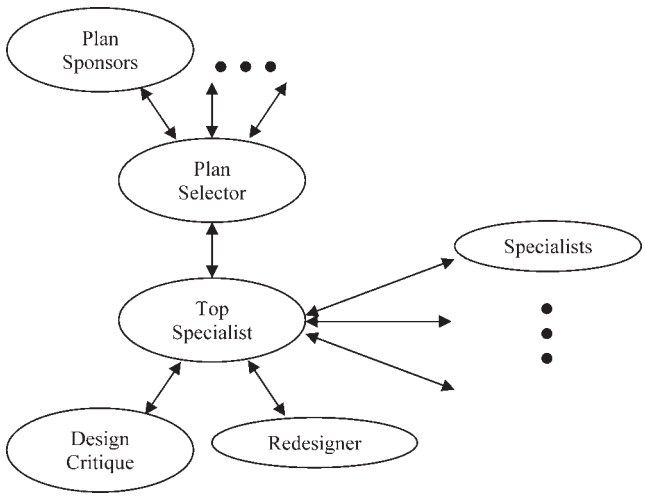

Fig. 5. The architecture of generic task routine design.

The MDM architecture deals with the third challenge, the unavailability of compatible models, by keeping agent communication as simple as possible with a well-defined typology of possible queries. The MDM approach asserts that the internal architecture of a model does not have any relevance to how the product integrates structurally and functionally in a larger design. MDM agents communicate only through message passing, and integrate in complex designs through FRMs (Section 3.1), which are nothing but representations of *n*-dimensional response surfaces. Because the actual models are not transferred between platforms, the problem of uniform model representation across platforms is alleviated.

The MDM approach also addresses challenges 4 and 5, which are protection of proprietary data and keeping the network traffic in manageable levels by use of the FRM technique. The FRM technique loses the internal topology of the assembled device, thus hiding the proprietary data (Section 3.2). Transfer of FRMs during simulation reduces the number of communications to *one* for each subcomponent, significantly reducing the network traffic.

An integrated RD-MDM platform therefore meets challenge 6, unavailability of distributed design and simulation tools, by incorporating the merits of both RD and MDM techniques. In the rest of this section we will extend the GT-RD framework with the selection of remotely represented MDM agents and elaborate the integration of MDM in the RD methodology for simulation based testing of distributed assemblies.

5.1. Selection of off-the-shelf components

GT-RD decomposes the overall problem into a hierarchy of cooperating specialists, each responsible for a specific part of the design (Fig. 5). Typically, higher level specialists represent the conceptual aspects of the design, whereas lower level specialists are responsible for selecting actual components. In the current RD platform, designers are allowed to use a pool of locally represented, generic components for parameterization of the design. We aim to enhance the RD platform by enabling the selection of alternatives among remotely distributed components, that is, off-the-shelf parts.

From the perspective of a designer, any remote design agent is nothing but a representation of the external behavior of a subsystem. This perspective is also valid for an RD process; when a lower level specialist selects suitable subsystem parameters (for component, material, process, or plan), the generated subsystem functions as a knowledge base that will have to be incorporated into the design. Thus, the RD system can be extended to implement a Remote Agent Selection Task (Fig. 6) to parameterize the design by choosing a suitable remote agent that belongs to a category that is specified by the designer.

In the integrated RD-MDM architecture (Fig. 6), each specialist is furnished with an *Assembler* or *Join* to incorporate the selected components in the design. Each specialist has access to computational resources to effectively respond to queries, depicted as *External Services*. An external service can be any local tool such as MathWorks MATLAB or a Dynamic Link Library. Off-the-shelf components are searched and contracted by the *Remote Agent Selection Task*, which carries out a multiattribute evaluation of the queried MDM agents. In the current architecture all connections are point-to-point and asynchronous.

With this improved RD tool, designers have the option to do multiattribute search and select among design parts that are available locally and remotely in the form of an MDM agent. If a commercially available MDM agent is an option, the designer must create a remote agent selection task for the related low-level specialist. This task specifies a category of MDM agents (via Agent Registry) and the queries of

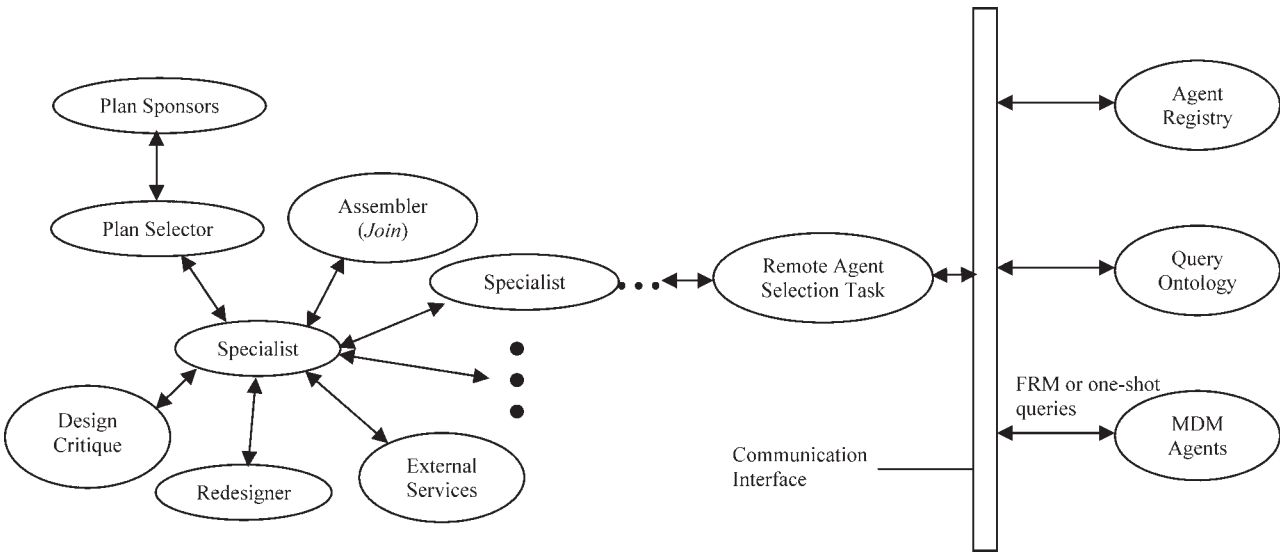

Fig. 6. The integrated routine design–modular distributed modeling architecture.

concern (via Query Ontology) for selecting the right part. Agent selection is made by matching the design requirements with the agent performance figures.

When the most suitable remote agent is selected by the RD process, it becomes a component of the design. The designer may prefer to keep a virtual link to the remote agent, or he/she may request and store an FRM of the remote agent. Keeping only a virtual link would ensure up to date responses from the remote agent, slightly slowing down the simulation for large number of components. When the FRM of the agent is stored on the local computer, analysis will be faster, but periodic updating of the FRM representation will become a necessity.

Consider an automobile manufacturing company that is capable of manufacturing all of the needed components except the drive train. An automobile designer of this company sets up the RD system as usual but creates a remote agent selection task in the plans of the drive train specialist. This situation is depicted in Figure 7. When the design process is started, remote agent selection task queries all MDM agents in the "Drive Train" category, selects a suitable drive train, and returns it to the drive train specialist. At this point, the design is parameterized with a remote component and the design can proceed to selection of other local or remote components and their integration in the overall design.

5.2. Joining selected components—Integrating components in an assembly

Selection of components that are represented remotely as MDM agents corresponds to selecting commercially available parts from a catalog for use in design. In a real-world design problem, the next step would be incorporating these parts in the overall design and analyzing their interactions. For this purpose we use the MDM *Join* operations (Section 3.2) and extend RD by implementing a new simulation tool in GT-RD.

In extended GT-RD, only the lower level specialists are responsible for selecting remote agents. Involving with a single component, these specialists do not require any *Join* operations. However, as the design proceeds up to higher level specialists, different local and remote models will start merging together. In any one of these specialists, the designer may need to define the *Join* operation that brings subassemblies together, optionally followed by a simulation.

MDM methodology can be incorporated in GT-RD as a recursive process that starts with merging the FRMs of locally represented components and/or distributed off-the-shelf parts and proceeds with merging the FRMs of subassemblies at higher abstraction levels (Fig. 6). The addition of a design testing capability at every abstraction level resolves the design failures at the lowest abstraction level they occur. In this scheme a subassembly that does not conform to the rest of the design will be discarded in order to generate a new and viable subassembly, before the design progresses to higher levels of abstraction.

The incorporation of MDM in GT-RD is accomplished by furnishing each parent specialist with an external service for simulation. To carry out a simulation, the responses of its every child have to be joined together. After the completion of each *Join* operation, the parent specialist becomes a representative of a joined, single component. This component is declared as a *local* MDM agent, which is *not* accessible from the outside world. Simulation of this assembly corresponds to querying the local agent with inputs and retrieving the outputs.

As discussed in Section 4.2 and depicted in Figure 6, every parent specialist follows a strategy as dictated by one of its plans. When the simulation of the outcome of a plan fails, redesigners are invoked and the design proceeds downward with the selection of a new plan. If the simulation is successful, the design will proceed upward, through parent specialists, their *Join* operations, and simulations. With each successful simulation, the designer is given the option of registering

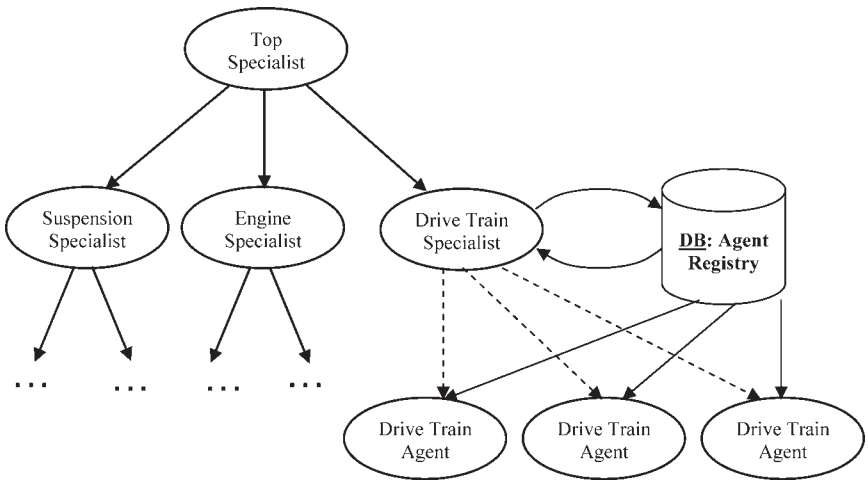

Fig. 7. Automated routine design of an automobile using modular distributed modeling agents.

https://doi.org/10.1017/S0890060408000012

the local agent that was created for simulation purposes. When registered, it becomes accessible to the MDM community. This functionality will most often be used for the output of the top-level specialist, that is, the product that is being designed.

6. AN EXAMPLE—FUEL CELL VEHICLE DESIGN

6.1. Setting up the routine designer and component selection process

We carried out our experiments using an MDM population of 4 electric motor, 6 transmission, and 12 fuel cell agents, communicating over 32 predefined queries. These agents were distributed over three servers, communicating with each other over TCP/IP. The Agent Registry and Query Ontology Agents were held on a separate server. A client computer was used to run the RD-MDM designer and communicate with the Agent Registry, Query Ontology, and MDM model servers.

In a real-world application the MDM agents will be published by their suppliers on the Agent Registry and the queries will be defined centrally by the Query Ontology. Therefore, a fuel cell vehicle designer's tasks will start with the setting up of the routine designer and not its components. The demonstration below will also assume that an MDM population and the necessary ontologies are readily available and proceed to the RD-MDM designer platform. The reader is referred to Eskil (2004) for generation of an MDM population.

As explained in Section 4.2, an RD problem solver is composed of specialists, their design plans, and plan tasks, each equipped with one or more steps of operations. To avoid repetition, we will only demonstrate the component selection process for the electric motor in this paper. The design of a fuel cell vehicle starts with the top specialist, which is called the "fuel cell vehicle specialist." This specialist has only one plan, whose items are calls to (Fig. 8)

1. *drive equipment specialist*,
2. *fuel equipment specialist*,
3. *acceleration simulation*, and
4. *fuel consumption simulation.*

When RD-MDM is run with a set of design requirements, the top specialist determines the design requirements for the drive equipment according to its design plan, and hands them to drive equipment specialist. As shown in the design decomposition, the drive equipment specialist is responsible for selection and (optionally) assembly of the drive train components. The drive equipment specialist transfers the related design requirements to electric motor and transmission specialists. Selection of the electric motor component depends on the acceleration requirement on the vehicle. After the selection of the electric motor, the electric motor specialist returns the name of the selected agent to the drive equipment specialist, who in turn, hands the control to the transmission specialist. The design proceeds with the fuel equipment specialist and its subspecialist, the hydrogen fuel cell specialist.

Our extension to RD for selection of off-the-shelf components (Section 5.1) enables the selection of remote MDM agents in the "Selection Steps." Each of these steps compares the remote agent responses with the selection criteria set by the designer. A selection step evaluates agents in the specified agent category and puts them in bins ranging from "perfect" to "neutral" and "rule out." This step offers the flexibility to do a multiattribute search and the designer is free to choose as many selection criteria as needed. For instance, the delivery time for the part, or the characteristics of previously determined design subassemblies could also be incorporated in this step.

At this point it is important to note that the outcome of our research is the infrastructure that is used to build the fuel cell vehicle routine designer, and not the designer itself. It is the human designer's responsibility to build such an RD structure with respect to techniques and knowledge that pertain to the domain. This distributed RD infrastructure gives designers the capability of applying the RD technique in various domains and across platforms, where the technique is applicable.

6.2. Assembling the selected components and simulation-based design testing

Let us step back and look at the plan items of the top specialist, that is, the fuel cell vehicle specialist. The drive equipment and fuel equipment specialists, as discussed above, will control the selection of suitable remote agents that will be incorporated as the electric motor, transmission, and fuel cell components in the fuel cell vehicle design. When the fuel equipment specialist completes its tasks and returns the agent names to the top specialist, the parameterization of the simplified fuel cell vehicle design example is complete. At this point, our augmentation of RD with simulation (Section 5.2) comes into play. RD-MDM uses the *Join* technique (Section 3.2) to assemble the fuel cell vehicle and analyze the assembly by means of simulations. The assembled fuel cell vehicle model is in the form

$$\begin{bmatrix} V_{\mathrm{FC}} \\ e_{\mathrm{ext}} \\ \tau_{\mathrm{ext}} \\ F_{\mathrm{ext}} \end{bmatrix} = \hat{\mathbf{K}} \begin{bmatrix} V_{\mathrm{FC}} \\ Q_{\mathrm{FC}} \\ \theta_{\mathrm{EM}} \\ x \end{bmatrix}, \tag{10}$$

where $\hat{\mathbf{K}}$ is the 4×4 FRM of the fuel cell vehicle; V_{FC} is the voltage request from the fuel cell stack; Q_{FC} is the charge generated by the fuel cell stack; θ_{EM} is the angular displacement of the electric motor shaft; x is the displacement of the vehicle; and e_{ext}, τ_{ext}, and F_{ext} are the external disturbances on the voltage (assumed zero), torque (brake on tire), and force (air drag on vehicle), respectively. Because of space considerations, we are unable to include the assembly of the fuel cell vehicle in this paper; refer to Eskil (2004) for a full formulation.

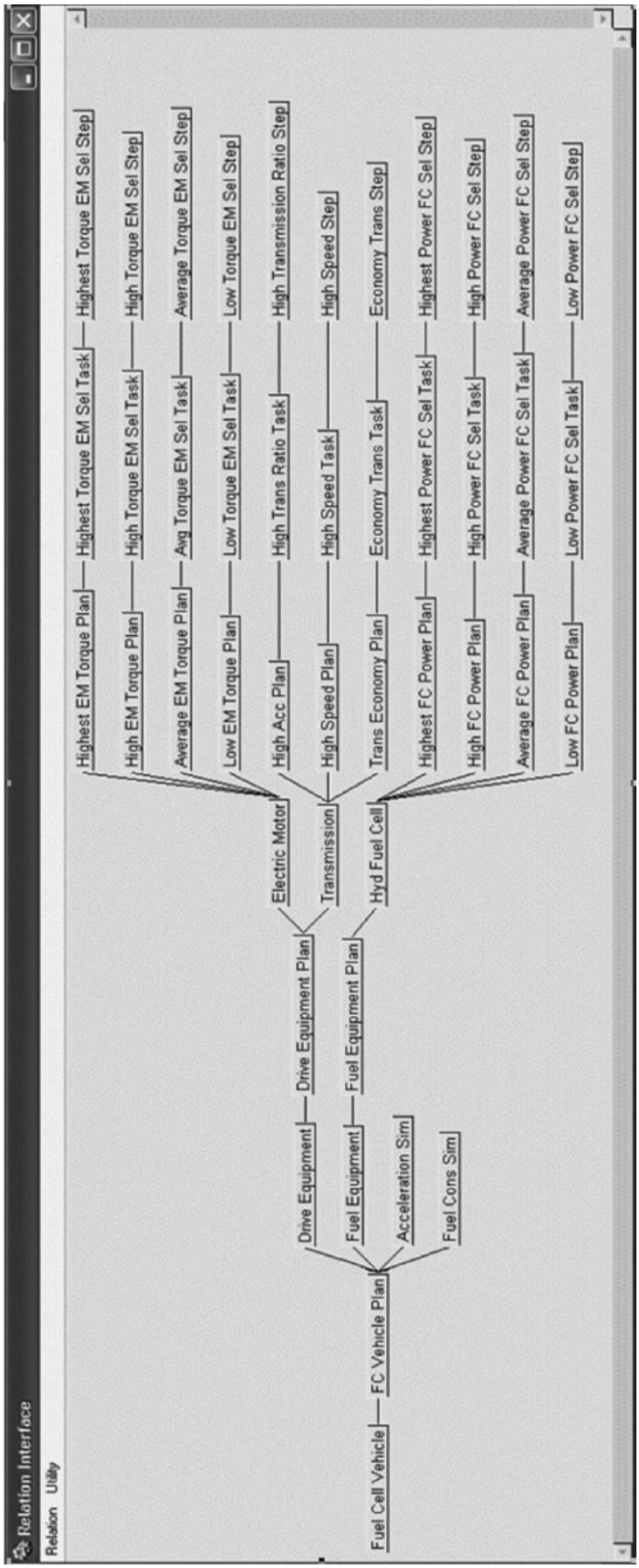

Fig. 8. Routine designer for fuel cell vehicles.

https://doi.org/10.1017/S0890060408000012

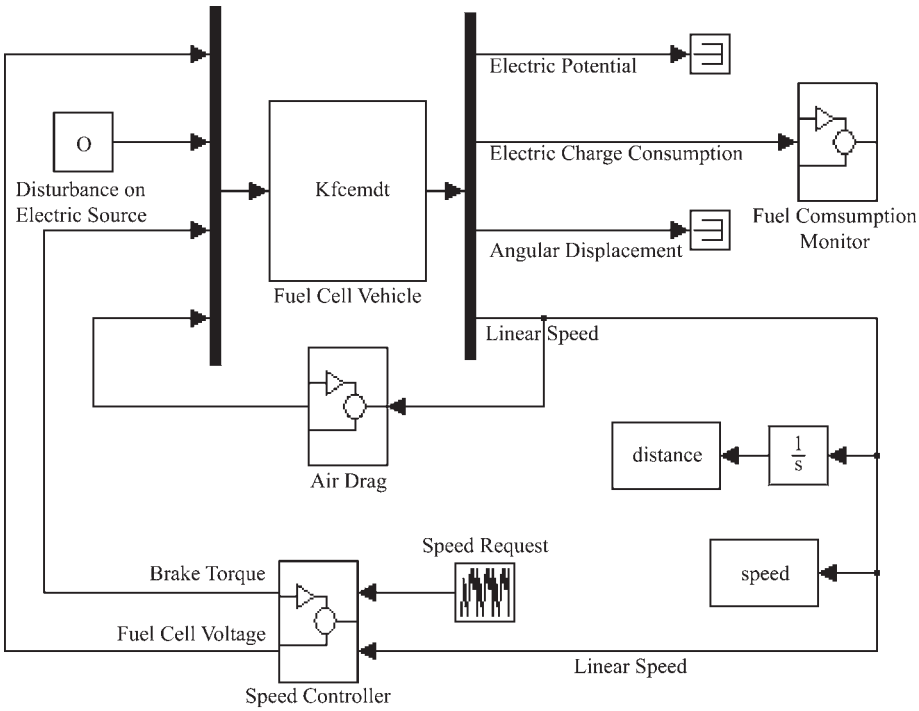

Fig. 9. The MATLAB Simulink model for fuel cell vehicle simulation.

The third item in the top specialist's plans, acceleration simulation, runs a simulation on the assembled component FRMs to determine the time to reach from 50 to 70 mph. Item 4 in the top specialist's plan, fuel consumption simulation, is very similar to acceleration simulation, the only difference being the simulation input for required speed. Acceleration simulation simulates the vehicle by a speed requirement of constant 160 mph, whereas fuel consumption simulation applies varying speed requests in compliance with hypothetical city driving conditions. Note that the GT-RD architecture also enables putting constraints on the outcomes of these simulations as well as other parameters of interest, which would initiate redesign with an automatic, rule-based evaluation of the failure, and selection of alternative design plans in case a constraint is not met.

The MATLAB Simulink model used for the acceleration and fuel consumption simulations is presented in Figure 9. The simulation model K_{fcemdt} for the fuel cell vehicle (fuel cell–electric motor–drive train assembly) is represented in a linear time invariant (LTI) block. As demonstrated in Section 3.2, the simulation model is the inverse of the assembled FRM $\hat{\mathbf{K}}$ [Eq. 10] of the fuel cell vehicle.

The first input to the LTI block is the voltage request from the fuel cell. This input is supplied by the speed controller, which compares the speed request with the vehicle speed and produces a feedback by use of a PI controller. The third input is the torque disturbance on the transmission and tire connection, and is utilized as the port for the brake torque. The fourth input is the external disturbance on the vehicle, that is, air drag. As demonstrated, the LTI block not only represents the overall system but also hides the details on the implementation and the components of the assembly.

When the RD-MDM fuel cell vehicle designer is run with design requirements (the vehicle accelerates from 50 to 70 mph in less than 10 s, and the vehicle's top speed is greater than 100 mph) a fuel cell vehicle is parameterized in accordance, assembled, and simulated with varying inputs. The plots of vehicle speed versus time for the automatically generated parameterization of the fuel cell vehicle are depicted in Figure 10. Acceleration simulation returned the time from 50 to 70 mph as 7.9 s and the result of the Max Speed simulation was 118.9 mph.

When the simulations are over, the Fuel Cell Routine Designer returns the selected components as well as the simulation results. As shown in Figure 11, although it was not guaranteed for all inputs, the routine designer had met the design requirements.

7. CONTRIBUTIONS AND CONCLUSIONS

This paper introduced RD-MDM, a conceptual framework and implementation that supports task directed, distributed RD including simulation-based design testing. In this research we leveraged the MDM methodology to simulate the interaction of design components in an assembly. The deliverable of our research is a distributed RD architecture that is capable of

1. automated multiattribute search for remote off-the-shelf design components represented as MDM agents,

2. design parameterization by choosing suitable components for the design,
3. integrating selected components in a distributed assembly and running simulations for design testing, and
4. making the final design available to prospective buyers without disclosing neither the internal connection of the utilized components nor the internal functional implementation.

We have achieved our goals on a prototype implementation by integrating the MDM methodology into the RD framework and extending both methodologies as necessary. The incorporation of simulation-based design testing in RD extends the RD methodology in two major dimensions:

1. simulation-based testing of distributed subassemblies at chosen abstraction levels of a design and
2. simulation-based testing of the overall design.

The first dimension enables determination of subassemblies that do not conform to the design requirements at an early stage in design. It also efficiently handles the incompatibility problem between subassemblies. In this scheme a subassembly that does not conform to the rest of the design or applicable design requirements will be discarded to generate a new and viable subassembly before the design progresses to higher levels of abstraction. The second dimension enables performance and failure tests on the complete design before publishing it as an MDM agent on the Internet, or before the manufacturing stage commences.

With RD-MDM, integrators can design and virtually assemble their end products by taking advantage of a global network of suppliers and evaluate design alternatives without the necessity of nondisclosure agreements. An integrator can also serve as a supplier to other integrators by making its RD-MDM-generated product model publicly available in the MDM community for evaluation as a part of higher level design.

An integrated RD-MDM framework creates a platform that realizes the potential of automated design that has been mitigated by lack of global access to design knowledge. It will be especially beneficial in decreasing the engineering design cycle time, increasing the commercial agility of the manufacturer, enabling custom-made designs while securing the proprietary design data, and keeping the network traffic within manageable levels. These benefits can be assessed in three major dimensions: cost, quality, and modularity.

7.1. Cost

Experience has shown that standardization approaches to model representation require massive conversions from report archives, CAD drawings, product catalogs, and so forth, to the standardized format, which is a significant burden on manufacturers. RD-MDM provides an alternative to the current standardization approaches and keeps the communication grammar to the bare essentials, that is, query–response pairs, while protecting the proprietary design models. With our approach, manufacturers are only required to gather the information they would like to provide and publish it on the global agent registry. The simplicity of the RD-MDM approach keeps the implementation and maintenance costs of RD-MDM distributed design framework at a minimum.

In addition to elimination of the costs associated with the standardization efforts, the RD-MDM framework promises significant savings in the product development stages. First, RD-MDM enables system integrators to rapidly generate design alternatives by use of distributed computational models provided by the suppliers. This black-box modeling technique eliminates the need for nondisclosure agreements that

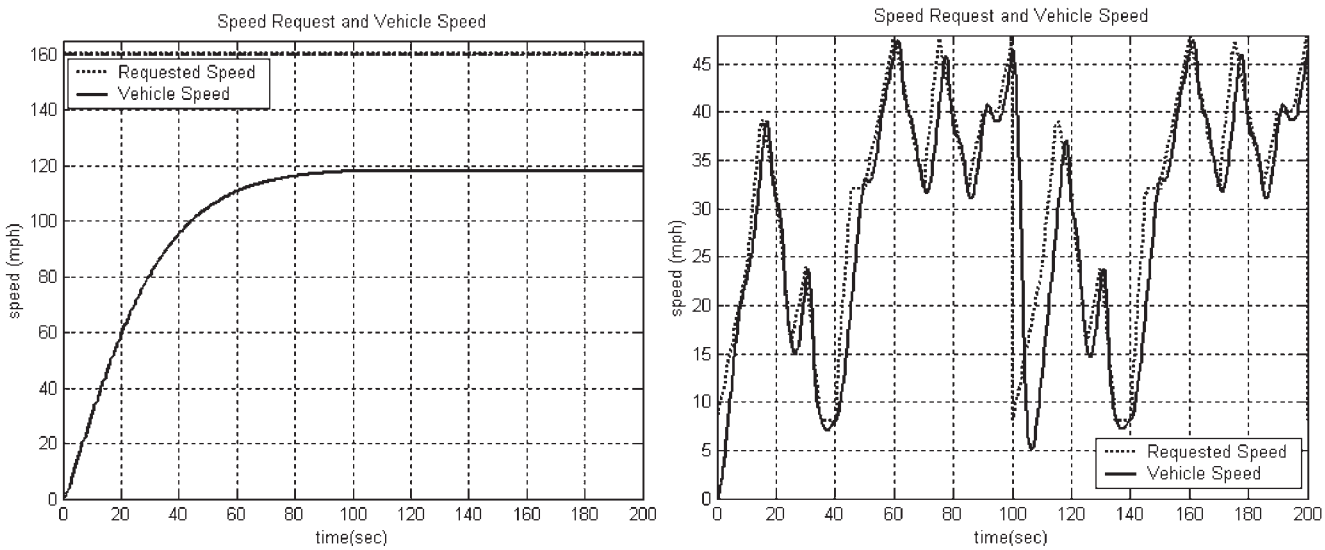

Fig. 10. Requested and achieved speeds in high-acceleration and city driving scenarios.

can take in the order of months to put in place, decreasing the engineering cycle time. Second, the RD-MDM approach simultaneously deals with the engineering design and project procurement practices such as procurement planning, supply market analysis, and solicitation (supplier selection). Provided that a reasonable number of manufacturers choose to promote their products by publishing them as MDM agents, such integration of engineering design and project procurement practices could produce significant savings in procurement-related transactions. Third, the capability to rapidly generate design alternatives and testing them by simulations eases the implementation of engineering changes during the manufacturing stage, lowering the associated costs.

7.2. Quality

For assemblers, sourcing higher quality components is a requisite for ensuring the quality of the assembled products. RD-MDM facilitates a multiattribute search over a global network of design parts to automatically locate the most suitable components.

The interaction between design components also plays a crucial role in the overall quality of the assembly. RD-MDM addresses this concern by automatically integrating the selected components in an assembly and running simulations. These simulations can also be run beyond the intended operation range of the product for design failure testing provided that component device models are developed to implement failure.

7.3. Modularity

In this research, we are asserting that how a specific product achieves its functional properties internally does not have any relevance to its structural and functional integration in a larger design. Therefore, we propose strict encapsulation of design models as component agents, allowing communication only on a predetermined ontology that specifies the query and response formats while hiding the internal and proprietary structure of models. As a result, we can envision a large pool of freely available distributed model agents, each of which represents a solution to an engineering design problem and can be utilized as a plug-in solution to a subproblem of a more complex problem. Integration of the encapsulated design models in larger assemblies is recursive in the hierarchy of low- to high-complexity design solutions.

8. LIMITATIONS AND FUTURE WORK

As discussed above and in Section 5.1, RD-MDM searches and selects design parts within prespecified categories of parts. Our test bed consisted of only three such categories of agents and their legal queries. As the population of participating MDM agents grows, there will be a need for systematic growth for the specifications of agent and query types and relations among them. Ontology research is an active field that deals with this taxonomy of knowledge and formal description and relationship of concepts in the domain of knowledge (Gruber, 1992; McGuiness et al., 2000; Gennari et al., 2002). The focus of ontology design, generating hierarchically correct ontologies, introducing new classes, avoiding cycles among classes, and handling inheritance, will prove to be important in a real-world application of RD-MDM and will need to be considered in the continuation of our research.

MDM methodology has so far been applied only on modeling of linear systems. For this reason, when a subcomponent with nonlinear characteristics is present in an assembly, a single FRM cannot be derived to represent the overall system. Although we can still design and simulate a nonlinear system by treating all nonlinear subcomponents as separate but coupled entities, there are important drawbacks of not integrating the assembly in a single FRM, such as reduced protection of proprietary data and cross-platform compatibility. This point underlines the synergistic relationship between RD problem solving and the MDM methodology; the integrated approach developed in the current research will benefit as

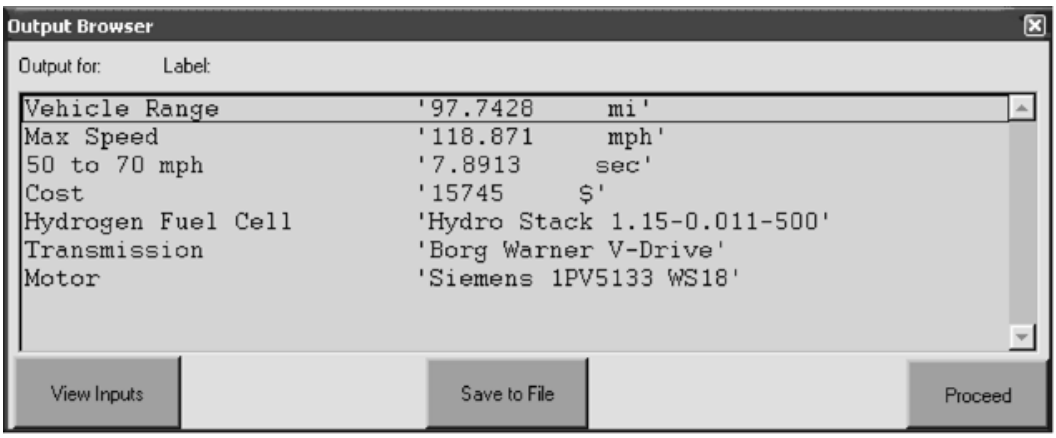

Fig. 11. Fuel cell vehicle design and simulation results.

improvements are made in either of these methodologies. The best way to deal with this problem is a future research direction of ours, that is, extending the MDM methodology to nonlinear systems.

Trust is also another issue to be dealt with in the current platform. Among the five types of trust defined by McKnight et al. (2002) we feel that the most crucial in the current state of the industry is the institution-based trust, that is, the truster puts more confidence in some institutions than others. The concept of trust must be implemented in the RD-MDM platform before it can be an effective design and analysis tool.

Last but not the least, our integration methodology must be extended to convey the assembler's requirements to suppliers so that new and innovative designs may evolve. In the past, suppliers and assemblers often interacted solely by exchange of geometric CAD data. However, as companies outsource more and more parts of their product, they gain leverage to dictate the requirements and specifications of the components they are seeking (Szykman et al., 2001). This reality can be exploited in RD-MDM by providing autonomy to the MDM agents that goes beyond choosing the right computational resource to effectively respond to queries or automatically replacing outdated or obsolete components. In this scheme, the client provides its requirements to an MDM agent of a particular type. The agent being queried on the other hand, goes through an automatic (and sometimes distributed) RD process to produce a design that is to the satisfaction of the client. It may also send such a redesign request to its remote components. This can be visualized as a process that initiates a dynamic evolution on some part of the MDM community. This scheme will furnish the RD-MDM structure with the currently lacking cooperative problem solving capability.

ACKNOWLEDGMENTS

This work was carried out in the Intelligent Systems Laboratory, Michigan State University. Parts of this work were completed in the Artificial Intelligence Laboratory, Işık University, and the Vision and Pattern Analysis Laboratory, Sabanci University, with the support of EU SPICE Project FP6-2004-ACC-SSA-2016684.

REFERENCES

Abrahamson, S., Wallace, D., Senin, N., & Sferro, P. (2000). Integrated design in a service marketplace. *Computer-Aided Design 2(2)*, 97–107.

Alexander, C. (1964). *Notes on the Synthesis of Form*. Cambridge, MA: Harvard University Press.

Ames, B.B. (2000). Digital design grows up. *Design News*, *19*, 92–94.

Augenbroe, G. (1995). An overview of the COMBINE project. *Proc. 1st European Conf. Product and Process Modelling in the Building Industry*, pp. 547–554, Dresden, Germany.

Ball, N.R., Matthews, P.C., & Wallace, K.M. (1998). Managing conceptual design objects: an alternative to geometry. *Artificial Intelligence in Design '98: Proc. 5th Int. Conf. Artificial Intelligence in Design, AID 98* (Gero, J.S., & Sudweeks, F., Eds.), pp. 67–86. New York: Kluwer Academic.

Brown, D.C., & Chandrasekaran, B. (1989). *Design Problem Solving: Knowledge Structures and Control Strategies*. San Mateo CA: Morgan Kaufmann.

Byam, B.P., & Radcliffe, C.J. (1999). Modular modeling of engineering systems using fixed input–output structure. *Symp. Systematic Modeling*, Orlando, FL.

Byam, B.P., & Radcliffe, C.J. (2000). Direct insertion realization of linear modular models of engineering systems using fixed input–output structure. *26th Design Automation Conf.*, Baltimore, MD.

Cave, P.R., & Noble, C.E.I. (1986). Engineering design data management. *Engineering Management: Theory and Applications (EMTA '86)*, pp. 301–307, Swansea.

Cera, C.D., Kim, T., Han, J., & Regli, W.C. (2004). Role-based viewing envelopes for information protection in collaborative modeling. *Computer-Aided Design 36(9)*, 873–886.

Chan, F.L., Spiller, M.D., & Newton, A.R. (1998). WELD—an environment for Web-based electronic design. *Design Automation Conf.*, San Francisco, CA, 146–151.

Chandrasekaran, B. (1988). Generic tasks as building blocks for knowledge-based systems: the diagnosis and routine design examples. *Knowledge Engineering Review 3(3)*, 183–210.

Chandrasekaran, B., & Johnson, T.R. (1993). Generic task and task structures: history, critique and new directions. In *Second Generation Expert Systems* (David, J.-M., Krivine, J.-P., & Simmons, R., Eds.). Berlin: Springer–Verlag.

Court, A.W., Culley, S.J., & McMahon, C.A. (1997). The influence of information technology in new product development: observations of an empirical study of the access of engineering design information. *International Journal of Information Management 17(5)*, 359–375.

Cutkosky, M.R., Engelmore, R.S., Fikes, R.E., Genesereth, M.R., Gruber, T.R., Mark, W.S., Tenenbaum, J.M., & Weber, J.C. (1993). PACT: an experiment in integrating concurrent engineering systems. *IEEE Computer 26(1)*, 28–37.

Dalpasso, M., Bogliolo, A., & Benini, L. (1999). Specification and validation of distributed IP-based designs with JavaCAD. *Proc. IEEE Design, Automation and Test in Europe*, pp. 684–688.

Dalpasso, M., Bogliolo, A., & Benini, L. (2002). Virtual simulation of distributed IP-based designs. *IEEE Design & Test of Computers 19(5)*, 92–104.

Dong, A., & Agogino, A.M. (1996). Text analysis for constructing design representations. *Artificial Intelligence in Design '96: Proc. 3rd Int. Conf. Artificial Intelligence in Design, AID 96* (Gero, J.S., & Sudweeks, F., Eds.), pp. 21–38. New York: Kluwer Academic.

Eskil, M.T. (2004). *Distributed routine design over the Internet with cooperating MDM agents*. PhD Thesis. Michigan State University, Computer Science and Engineering Department.

Eskil, M.T., Sticklen, J., & Radcliffe, C.J. (2003). Modular distributed modeling. *4th Int. Collaborative Technology Symp*. pp. D3–202, Orlando, FL.

Fin, A., & Fummi, F. (2000). A Web-CAD methodology for IP-core analysis and simulation, *IEEE and ACM Proc. Design Automation Conf.*, pp. 597–600, Los Angeles.

Fruchter, R., Clayton, M.J., Krawinkler, H., Kunz, J., & Teicholz, P. (1995). Interdisciplinary communication medium for collaborative conceptual building design. *Advances in Engineering Software 25(2–3)*, 89–101.

Fruchter, R., Reiner, K., Lifer, L., & Toye, G. (1996). Visionmanager: a computer environment for design evolution capture. *Artificial Intelligence in Design '96: Proc. 3rd Int. Conf. Artificial Intelligence in Design, AID 96* (Gero, J.S., & Sudweeks, F., Eds.), pp. 505–524. New York: Kluwer Academic.

Gennari, J., Musen, M.A., Fergerson, R.W., Grosso, W.E., Crubezy, M., Eriksson, H., Noy, N.F., & Tu, S.W. (2002). *The Evolution of Protege: An Environment for Knowledge-Based Systems Development*. Palo Alto, CA: Stanford University.

Gruber, T.R. (1992). *Ontolingua: A Mechanism to Support Portable Ontologies*. Palo Alto, CA: Stanford University, Knowledge Systems Laboratory.

Gu, B., Asada, H.H., & He, X.D. (2002). Software development of co-simulation of algebraically coupled dynamic subsystems without disclosure of proprietary subsystem models. *ASME Int. Mechanical Engineering Congr. Exhibition*, Paper No. 39287.

Hauck, S., & Knoll, S. (1998). Data security for Web-based CAD. *ACM/IEEE Design Automation Conf.*, pp. 788–793, San Francisco, CA.

Helaihi, R., & Olukotun, K. (1997). Java as a specification language for hardware–software systems. *Proc. 1997 IEEE/ACM Int. Conf. Computer-Aided Design*, pp. 690–697, San Jose, CA.

Hubka, V., & Eder, W.E. (1996). *Design Science*. New York: Springer–Verlag.

Jennings, N.R., & Bussmann, S. (2003). Agent-based control systems. *IEEE Control Systems 23(3)*, 61–74.

Kamel, A., & Sticklen, J. (1994). Multiple design: an extension of routine design for generating multiple design alternatives. *Artificial Intelligence in Design, '94*, pp. 275–292, Dordrecht.

Keskinocak, P., Goodwin, R., Wu, F., Akkiraju, R., & Sesh, M. (2001). *Decision Support for Managing an Electronic Supply Chain*, Vol. 1, pp. 15–31. New York: Kluwer Academic.
Kopena, J.B., & Regli, W.C. (2003). Functional modeling of engineering designs for the semantic Web. *IEEE Data Engineering Bulletin 26(4)*, 55–62.
Lenz, T.J., McDowell, J.K., Hawley, M.C., Kamel, A., & Sticklen, J. (1996). The evolution of a decision support architecture for polymer composites design. *IEEE Expert 11(5)*, 77–83.
MacGregor, S.P., & Thomson, A.I. (2001). A case study on distributed, collaborative design: investigating communication and information flow. *6th Int. Conf. Computer Supported Cooperative Work in Design*, pp. 249–254, London, Ontario, Canada.
Marsh, J.R. (1997). *The capture and utilisation of experience in engineering design*. PhD Thesis. University of Cambridge.
Maturana, F., & Norrie, D.H. (1996). Multi-agent mediator architecture for distributed manufacturing. *Journal of Intelligent Manufacturing 7*, 257–270.
Maturana, F., Shen, W., & Norrie, D.H. (1999). MetaMorph: an adaptive agent-based architecture for intelligent manufacturing. *International Journal of Production Research 37(10)*, 2159–2174.
McGuiness, D.L., Fikes, R., Rice, J., & Wilder, S. (2000). The chimaera ontology environment. *17th National Conf. Artificial Intelligence (AAAI 2000)*, Austin, TX.
McKnight, D.H., Choudhury, V., & Kacmar, C. (2002). Developing and validating trust measures for e-commerce: an integrative topology. *Information Systems Research 13(3)*, 334–361.
Nowack, M.L. (1997). *Design guideline support for manufacturability*. PhD Thesis. University of Cambridge.
Pahng, F., Senin, N., & Wallace, D. (1998). Distribution modeling and evaluation of product design problems. *Computer-Aided Design 30(6)*, 411–423.
Radcliffe, C.J., & Sticklen, J. (2003). Modular distributed models of engineering structures. *Int. Mechanical Engineering Congr. Exhibition*, Paper No. 41171, Washington, DC.
Regli, W.C. (1997). Internet-enabled computer-aided design. *IEEE Internet Computing 1(1)*, 39–51.
Reichenbach, D. (2003). *Modeling of dynamic system using Internet engineering design agents*. Master's Thesis. Michigan State University.
Rodgers, P.A. (1997). *The Capture and Retrieval of Design Information: An Investigation of the Information Needs of British Telecom Designers*. Cambridge: University of Cambridge, Engineering Design Centre.
Rodgers, P.A., Huxor, A.P., & Caldwell, N.H.M. (1999). Design support using distributed Web-based AI tools. *Research in Engineering Design 11(1)*, 31–44.
Rosenman, M.A., & Gero, J.S. (1996). Modelling multiple views of design objects in a collaborative CAD environment. *Computer-Aided Design 28(3)*, 193–295.
Schlenoff, C., Denno, P., Ivester, R., Libes, D., & Szykman, S. (2000). An analysis and approach to using existing ontological systems for applications in manufacturing. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 14*, 257–270.
Schott, H., Buttnet, K., & Birkhofer, H. (1997). Information resource management for design—illustrated by Hypermedial Guidelines. *Int. Conf. Engineering Design*, pp. 179–184, Tampere, Finland.
Shakeri, C., & Brown, D.C. (2004). Constructing design methodologies using multiagent systems. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 18*, 115–134.
Shen, W., & Barthes, J.P.A. (1996). An experimental environment for exchanging design knowledge by cognitive agents. In *Knowledge Intensive CAD* (Mantyla, M., Finger, S., & Tomiyama, T., Eds.), Vol. 2, pp. 19–38. London: Chapman & Hall.
Silva, M.J., & Katz, R.H. (1995). The case for design using the World Wide Web. *Proc. ACM/IEEE*, pp. 579–585.
Sinz, C., Kaiser, A., & Küchlin, W. (2003). Formal methods for the validation of automotive product configuration data. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 17(1)*, 75–97.
Smith, L., & Reinertson, D.G. (1991). *Developing Products in Half the Time*. New York: Van Nostrand Reinhold.
Spiller, M.D., & Newton, A.R. (1997). EDA and the network. *IEEE Int. Conf. Computer-Aided Design*, pp. 470–476.
Sticklen, J., Kamel, A., Hawley, M., & Delong, J. (1992). An artificial intelligence-based design tool for thin film composite materials. *Applied Artificial Intelligence 6(6)*, 303–313.
Szykman, S., Sriram, R.D., & Regli, W.C. (2001). The role of knowledge in next-generation product development systems. *Journal of Computing and Information Science in Engineering 1(1)*, 3–11.
Wallace, D., Yang, E., & Senin, N. (2001). *Integrated Simulation and Design Synthesis*. Cambridge, MA: Massachusetts Institute of Technology, Center for Innovation in Product Development.
Wooldridge, M. (1997). Agent-based software engineering. *IEE Proceedings on Software Engineering 144*, 26–37.
Zhang, L.J., Chao, T., Chang, H., & Chung, J.Y. (2003). *XML-Based Advanced UDDI Search Mechanism for B2B Integration*, Vol. 3, pp. 25–42. New York: Kluwer Academic.

Taner Eskil is an Assistant Professor in the Department of Computer Science and Engineering, Işık University, Turkey. Dr. Eskil obtained his BS in mechanical engineering and holds an MS in systems and control engineering. He received his PhD in 2005 from Michigan State University for his work in routine design and distributed simulation of dynamic systems. His research interests include knowledge-based systems and Internet agent support for electronic commerce.

Jon Sticklen is an Associate Professor in the Department of Computer Science and Engineering, College of Engineering, Michigan State University. His research in knowledge-based systems has emphasized an articulation between applications development and needs-driven theory advancement, particularly in task-specific architectures and function-based reasoning. Applications areas have included troubleshooting for high-performance aircraft, landscape level ecological modeling, and automated support for materials systems design and manufacturing planning of integrated structures made from polymer composites. More recently, Dr. Sticklen's major focus has been on pedagogical issues, and in particular, on the pedagogy of first-year engineering.

Clark Radcliffe has been at Michigan State University since 1980. He received his PhD from the University of California in 1980 for his work in the dynamics and control of circular saw vibration. He has served as Associate Technical Editor of two professional journals, Chair of the ASME Dynamic Systems and Control Division, as well as serving on the Systems and Control Operating Board of ASME. Dr. Radcliffe's research and teaching interests are in the area of dynamic systems and control with specific interests in the interfaces between mechanical, electrical, and computer systems.