Proceedings of the TMCE 2008, April 21–25, 2008, Izmir, Turkey, Edited by I. Horváth and Z. Rusák

INTEGRATING VENDORS INTO COOPERATIVE DESIGN PRACTICES

Mustafa Taner Eskil
Computer Science Department
Işık University
eskil@isikun.edu.tr

Jon Sticklen
Computer Science and Engineering Department
Michigan State University
sticklen@msu.edu

ABSTRACT

In this paper we describe a new approach to cooperative design using distributed, off-the-shelf design components. Our ultimate goal is to enable assemblers to rapidly design their products and perform simulations using parts that are offered by a global network of suppliers. The obvious way to realize this goal would be to transfer desired component models to the client computer. However in order to protect proprietary data, manufacturers are reluctant to share their design models without non-disclosure agreements, which can take in the order of months to put in place. Due to bandwidth limitations, it is also impractical to keep the models at the manufacturer site and do simulations by simple message passing. To deal with these impediments in e-commerce we leverage the Modular Distributed Modeling (MDM) methodology, which enables transfer of component models while hiding proprietary implementation details. We augment MDM methodology with Routine Design (RD) methods to realize a platform (RD-MDM) that enables automatic selection of secured off-the-shelf design components over the Internet, integration of these components in an assembly, running simulations for design testing, and publishing the approved product model as a secured MDM agent. We demonstrate the capabilities of the RD-MDM platform on a fuel cell –battery hybrid vehicle design example.

KEYWORDS

Cooperative Design, Simulation, Proprietary Data, Modular Distributed Modeling, Routine Design

1. INTRODUCTION

A manufacturing company must make use of all resources that are available internal and external to the company to thrive in today's marketplace. Internet and e-commerce affect engineering design profoundly by allowing companies to be more externally focused. However, the availability of vast number of suppliers on the Internet renders searching for candidates and locating the best candidate beyond the reach of a human designer. The changes in the acquisition process of enterprises need to be reflected in a new type of design and simulation environment, one that facilitates automated searching and locating of best products, integration of selected products in an assembly, and simulation of the overall design over the Internet.

The target of our research is to enable system integrators to rapidly design their products and perform simulation based design testing using secure computational models that are distributed over the Internet. There are two key challenges in the current wired world for achieving our target. First, in order to protect their proprietary data, manufacturers are reluctant to share design models with window shoppers without non-disclosure agreements. The second problem stems from the unavailability of automated tools that are capable of both distributed design and simulation-based design testing over the Internet. Most current engineering design and analysis tools are either limited to a local computer, need a vigorous standardization of distributed resources, or designed to operate on an exclusive virtual design network.

Protecting proprietary design models and openly sharing model functional capability has not been possible with traditional model-based approaches.

We are attacking the problem of sharing design models without revealing proprietary data by utilizing the Modular Distributed Modeling (MDM) methodology. 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 (Byam and Radcliffe 1999; Eskil, Sticklen et al. 2003).

To address the second challenge, unavailability of automated design and simulation tools, we extend the Routine Design (RD) methodology (Brown and Chandrasekaran 1989) in two dimensions. First, we extend the methodology by enabling design parameterization using distributed components. Second, we add the capability of design testing through simulation at all abstraction levels of the design. The underlying reason for our focus on the RD methodology is the routine nature of most real-world design problems.

The thrust in our work is to integrate MDM, and in particular its capability to provide simulation, into the routine design framework. The synergy of RD and MDM methodologies facilitates automated design parameterization with off-the-shelf components distributed over the Internet, virtual assembly of selected components, and simulation of the distributed assembly in an open, competitive e-commerce. With this approach, vendors will be able to make their core models available to the public without disclosing proprietary information. Designers on the other hand will be able to incorporate these models into their designs and simulate them as integrated components of the assembly.

In the rest of this paper we give a brief introduction to our research field, discuss the integrated RD-MDM system, present the current capabilities and limitations of the RD-MDM platform on a fuel cell – battery hybrid vehicle design example, and conclude with a summary and the status of our research. The next section describes existing distributed problem solving systems. Sections 3 and 4 outline the MDM platform and the RD methodologies. Our implementation is presented in Section 5. A hybrid vehicle design example is presented in Section 6, which is followed by results and conclusions.

2. DISTRIBUTED PROBLEM SOLVING APPROACHES

Solving complex problems as a whole proves to be intractable in many cases. Engineers' approach to such problems is to decompose them into several subtasks that may fall into the realm of different engineering domains. Early distributed problem solving approaches assumed that the expertise from these engineering domains could be gathered and represented on a network of closely bound computers, in compliance with a particular architecture.

For many real-world design problems however, gathering and organizing the widespread expert knowledge turns out to be infeasible. An example to such problems is automobile design, where expertise in mechanics (e.g. drive-train), vibrations (suspension), materials (tires, brake pads), electricity (accumulators, electric motors) and electronics (on-board computers) are required. As correctly identified by Alexander (1964) decades ago, the design information is widespread, unorganized, and in general beyond the reach of a single designer. In a more recent work MacGregor (MacGregor and Thomson 2001) also emphasized the lack of common terminology between teams of expertise and unawareness of existence of knowledge.

PACT is one of the most well-known projects that advocate encapsulation of tool data to solve the common terminology problem. 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 (Cutkosky, Engelmore et al. 1993) is implemented. The facilitator provides an interface between a local connection of agents and remote agents. The collection of autonomous agents under facilitators is called *federation architecture*.

Distributed Object-Based Modeling Environment (DOME) aims to create a modeling infrastructure for individuals to share their simulation services related to their expertise (Wallace 2001). 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. The DOME approach capitalizes on sharing design and simulation tools rather than component models.

The approaches mentioned above did not address to simulation of distributed assemblies while protecting the proprietary resources. Researchers proposed cryptographic techniques (Silva and Katz 1995; Hauck and Knoll 1998) and simulation to take place on the manufacturer site (Fin and Fummi 2000). However, these approaches are either platform dependent, hard to maintain, or do not support design by multiple components from different vendors.

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 et al. (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. On the other hand, as Regli draws attention to, advances in distributed design brings about the problem of handling Gigabytes of information flow over slow WWW protocols.

3. FUNCTIONAL RESPONSE MODELING

A major impediment in simulations of distributed assemblies is the extent of the Internet traffic that entails iterative communications. This challenge can be met by conceptualizing an output form that both hides proprietary data and enables *a functional response* to be made by the responder per simulation. The requesting Modular Distributed Modeling (MDM) agent is then able to use this single Functional Response Model (FRM) as the basis for local (to it) simulation that incorporates the device into its own device assembly. This is the core concept that will make MDM communities possible in the Internet environment (Byam and Radcliffe 2000; Eskil, Sticklen et al. 2003).

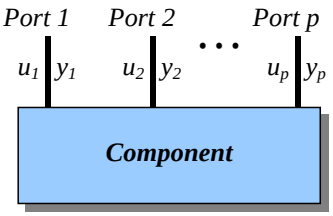

Figure 1 – Modular Modeling Element Graphical Notation

Power flows into Port i if u_i and y_i are both positive.

Figure 1 shows a diagram of a contracted model for an Internet design agent model. Modular modeling element graphical notation represents user-defined multi-port multi-DOF subsystem models with a rectangle. The bold lines represent the power ports with implicit standardized direction of positive power into the element and standardized input-output port causality. The direction of positive power 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:

$$\mathbf{Ky} = \mathbf{u} \tag{1}$$

where $\mathbf{K}$ is the component stiffness matrix, $\mathbf{y}$ is the component generalized displacement vector and $\mathbf{u}$ is the component 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.

As equation (1) implies, FRM is currently applicable only to linear time-invariant components. This constraint brings important limitations as most real-world design components exhibit nonlinear characteristics. In this paper we propose a method to implement nonlinear characteristics in an FRM application. Modeling nonlinear systems using the FRM technique is being studied in the Dynamic Systems Laboratory of Michigan State University.

3.1. Deriving FRMs for Assemblies

The Subsystem Model (Figure 2) depicts a possible situation that can arise when components are assembled into subsystems. The subsystem model has two components connected via constraints 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 (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 non-singular and solvable.

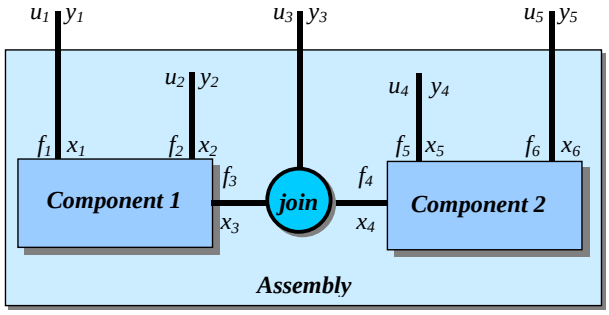

Figure 2 – Subsystem Model with Two Components

Each component is depicted with external, internal and connected ports.

For deriving a single FRM for the assembly, the equations for each of the components are first assembled into an unconstrained system matrix.

$$\begin{bmatrix} \mathbf{K}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_2 \end{bmatrix} \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix} \tag{2}$$

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

$$\mathbf{x} = \mathbf{S}\mathbf{y} \tag{3}$$

The power constraint on the assembly requires the sum of the work into all joined component ports to equal the applied work at any assembly connection. The causality in energy domains is defined such that this holds for every physical system in these domains. Therefore, the external work done on a physical assembly by port inputs **u** must equal the external work done on the assembly's components by port inputs **f**.

$$\mathbf{x}^T\mathbf{f} = \mathbf{y}^T\mathbf{u} \tag{4}$$

Applying the input-output constraint (Eq. 3) on the power constraint (Eq. 4) we find the input constraint between assembly's component input vectors **f** and assembly input vector **u** for all non-zero assembly outputs **y**.

$$\mathbf{S}^T\mathbf{f} = \mathbf{u} \tag{5}$$

These results are all we need to derive an assembly FRM. We start model assembly as in Eq. 2, with the unconstrained grouping of all component models:

$$\mathbf{K}\mathbf{x} = \begin{bmatrix} [\mathbf{K}_1] & [\mathbf{0}] & [\mathbf{0}] & [\mathbf{0}] \\ [\mathbf{0}] & [\mathbf{K}_2] & [\mathbf{0}] & [\mathbf{0}] \\ [\mathbf{0}] & [\mathbf{0}] & \ddots & [\mathbf{0}] \\ [\mathbf{0}] & [\mathbf{0}] & [\mathbf{0}] & [\mathbf{K}_n] \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_n \end{bmatrix} \tag{6}$$

To obtain a concise modular model, we will now apply the constraints. The assembly output constraint is applied by substituting constraint Eq. 3 into Eq. 6:

$$\mathbf{K}\mathbf{S}\mathbf{y} = \mathbf{f} \tag{7}$$

Multiplying both sides with $\mathbf{S}^T$ and using Eq. 5 yields the constrained assembly internal stiffness model

$$\hat{\mathbf{K}}\mathbf{y} = \mathbf{u} \tag{8}$$

where

$$\hat{\mathbf{K}} = \mathbf{S}^T\mathbf{K}\mathbf{S} \tag{9}$$

This simple system shown in Eq. 9 is the dynamic inverse simulation model or functional response model (FRM) of the assembly. 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 non-linear 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.

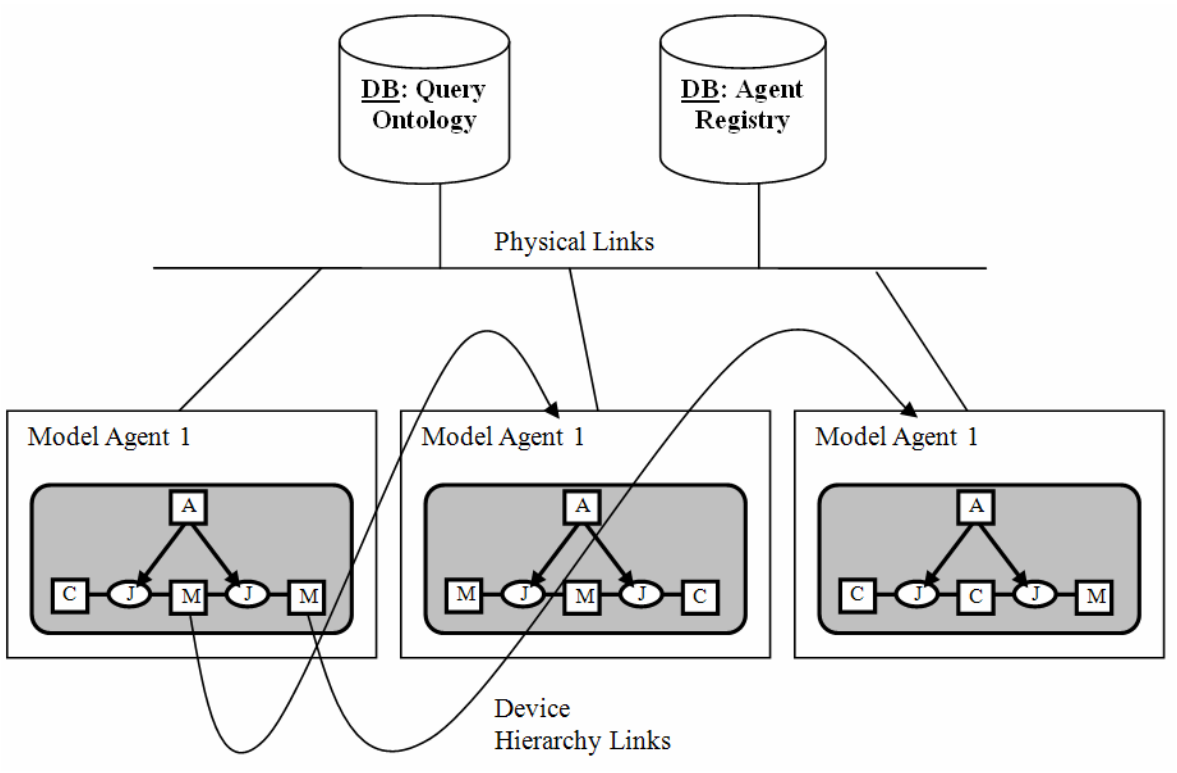

Figure 3 – Sketch of MDM agent community

Each agent is composed of an Assembler, Component(s) and Model(s). Components and Models are assembled with Join operations.

3.2. The MDM Platform

Bandwidth is a major concern in distributed simulations, especially when the simulation is iterative in nature. In our work FRM serves as a simulation model that fully describes the functional behavior of a component. This property of the FRM technique enables us to overcome the bandwidth problem by reducing the required number of communications for each component to *one* for the entire simulation. FRMs serve for two main goals in our research:

1. The external behavior of the manufacturing part is represented without revealing any proprietary information.
2. The transfer of FRMs in design and/or analysis speeds up the design process while reducing the network traffic.

Leveraging the FRM technique, we developed an infrastructure capable of supporting a community of Modular Distributed Modeling (MDM) agents. 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 modular distributed modeling is the MDM agent, a fixed but autonomous agent such that:

1. Each MDM agent holds knowledge of a single manufactured device.
2. Each MDM agent composes answers to queries such that implementation details of the device it holds are not revealed.
3. Any MDM agent may hold a device that internally includes parts (or assemblies) held by other MDM agents.

The sketch shown in Figure 3 depicts three MDM agents connected physically via the Internet. It is important to note that in response to queries, MDM agents would not reveal the proprietary internal virtual linkage; such virtual linkages can be used to express an internal structure that includes parts/assemblies of other MDM agents. Query Ontology is an MDM network resource that makes available to any MDM agent the typology of legal queries. Agent Registry makes available to any

MDM agent both the typology of agent types and the list of all existing MDM agents by agent type. In other words, Agent Registry serves as a database of agent categories and the registered agents under each category whereas Query Ontology provides a dictionary of legal queries for a chosen agent category.

With the MDM approach we assert that how a specific product is designed and its design details do not have any relevance to how it integrates structurally and functionally in a larger design. Therefore a product model can reside on any platform of its designer's convenience and communicate through one-shot queries and responses or FRMs that hide the proprietary knowledge.

4. ROUTINE DESIGN

When we engineers design similar artifacts over and over again, we often achieve a grasp of the routine nature of the process and start discovering effective ways to decompose the design process into smaller design problems. Although we may not know beforehand all possible situations that would occur in a design process, we acquire an understanding of design choices and plans that specify the order of making the choices. Routine Design (RD) is a procedure that aims to capture this "expert" knowledge from the designer and realize it in computer environment.

In a series of studies over the past decade, the Intelligent Systems Laboratory (ISL) of Michigan State University developed several computer based tools for engineering design and analysis. Among these tools, Generic Task Routine Design (GT-RD) architecture was first suggested and implemented by Brown (Brown and Chandrasekaran 1989) and developed later on in the ISL (Kamel and Sticklen 1994).

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. As RD proceeds higher in the specialist hierarchy, conceptual aspects of the design problem become more and more pronounced.

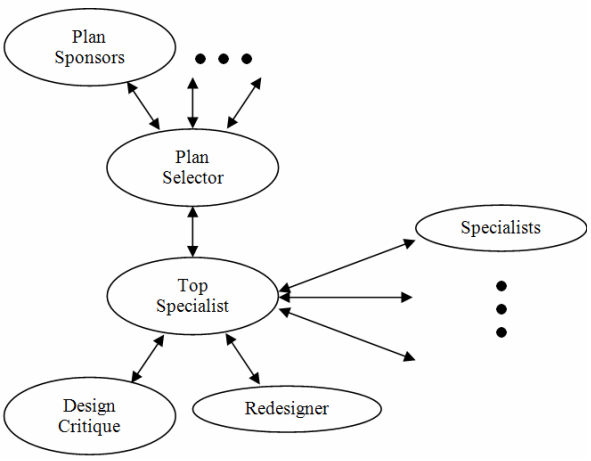

Figure 4 – The Architecture of GT-RD

The structure of the Top Specialist and all other specialists are inherently the same.

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 routine design 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 4. A plan sponsor matches the status of the design with the conditions for applicability of the plan and reports the level of appropriateness 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.

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 sub-designs 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.

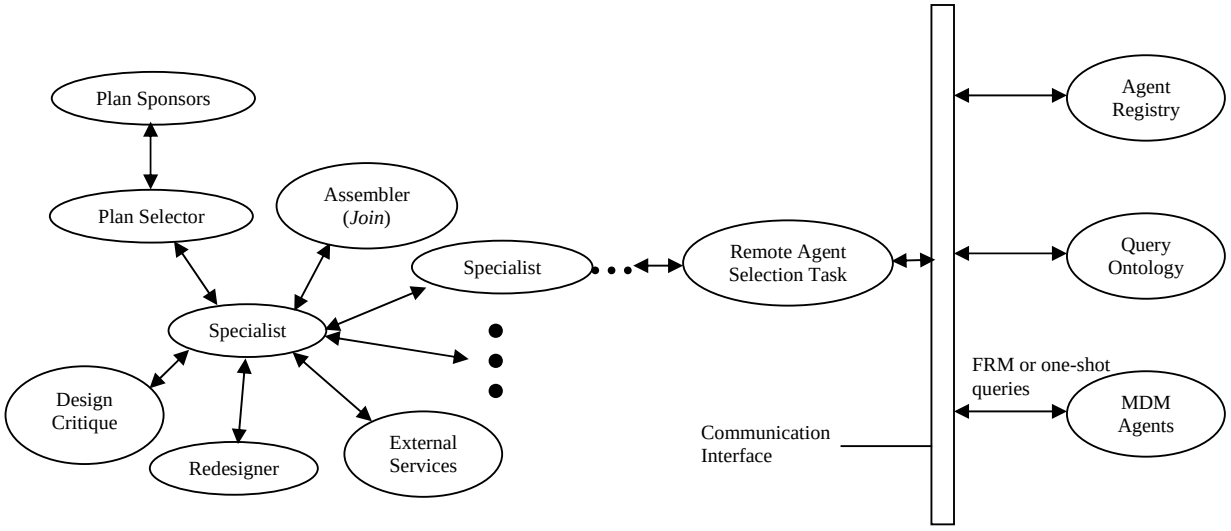

Figure 5 – The integrated RD-MDM architecture

5. THE RD-MDM PLATFORM

Typically in GT-RD decomposition, 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 enhanced the RD platform by enabling the selection of alternatives among remotely distributed components, i.e. 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, RD system can be extended to implement a Remote Agent Selection Task to parameterize the design by choosing a suitable remote agent that belongs to a specific category.

Figure 5 depicts the integrated RD-MDM architecture. In this architecture, each specialist is furnished with an Assembler 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 multi-attribute 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 multi-attribute 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 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 a Functional Response Model (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 in this company sets up the routine design structure creating a remote agent selection task in the plans of the drive train specialist. This situation is depicted in Figure 6. 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. The queries made for finding the write part are in general "one-shot" queries, such as 'price', 'delivery time', 'maximum power' or 'maximum torque'. When a drive train agent model is selected, the design is parameterized with a remote component and it can proceed to selection of other local or remote components and their integration in the overall design.

Selection of a component that is represented remotely as an MDM agent corresponds to selecting a commercially available part from a catalogue 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 FRM assembly operations (Section 3.2) and extend RD by implementing simulation capability.

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 assembly 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 an assembly that brings subassemblies together, optionally followed by a simulation.

MDM methodology is 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 (Figure 5). 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. In order to carry out a simulation, the responses of its every child have to be joined together. After the generation of each assembly, the parent specialist becomes a representative of an assembled, single component. This component is declared as a local MDM agent, which is not accessible from the outside world. Simulation of the assembly corresponds to querying this local agent with inputs and retrieving its outputs.

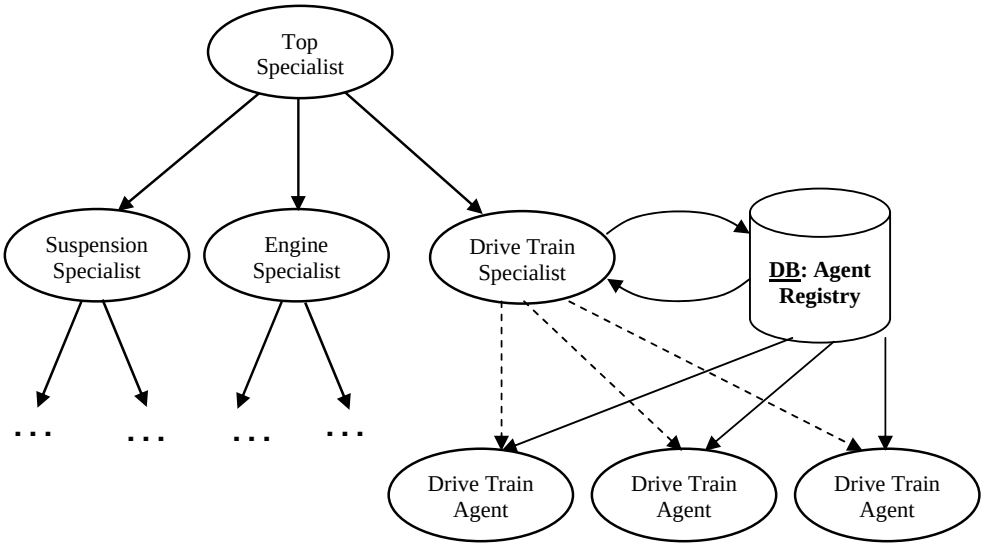

Figure 6 – Automated routine design of an automobile using MDM agents

Solid and dotted lines represent physical and virtual linkages, respectively.

As discussed in Section 4 and depicted in Figure 4, 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 upwards, through parent specialists, their assembly operations and simulations. With each successful simulation, the designer is given the option of registering 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, i.e. the product that is being designed.

6. THE HYBRID VEHICLE EXAMPLE

We chose to perform the distributed routine design of a hybrid vehicle to demonstrate the capabilities of the proposed architecture. We have several reasons for selecting this design problem. First, a hybrid vehicle power train is an inherently complicated and relatively new design problem, which gives us a chance to demonstrate the capabilities of our design platform on a cutting-edge application. Second, the drive train is composed of multiple components (e.g. fuel cell, battery, electric motor, transmission), allowing us to generate a reasonably sized population and ontology of MDM agents to demonstrate the automated agent selection process. Finally, design of a hybrid vehicle power train is a challenge in deriving FRMs and assembling mechanical and electrical components.

6.1. Assembling the Electric Motor and the Drive Train

To assemble the electric motor and the drive train components, we need to first define an unassembled dynamic modular model $\mathbf{K}$ as we have shown in Sec. 3.1. $\mathbf{K}$ is a diagonal matrix of modular models of components, such that:

$$\mathbf{Kx} = \mathbf{f} \tag{10}$$

The unassembled dynamic modular model of the electric motor and the drive train is composed of the electric motor, transmission (including axle and tires) and the vehicle mass models. Each of these models are represented as transfer functions ($\mathbf{K}_{EM}$, $\mathbf{K}_{TR}$, $\mathbf{K}_{VE}$) in Eq. 11.

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{EM} & 0 & 0 \\ 0 & \mathbf{K}_{TR} & 0 \\ 0 & 0 & \mathbf{K}_{VE} \end{bmatrix}$$

$$\mathbf{K} = \begin{bmatrix} L_{EM}s^2 + R_{EM}s & K_{EM2}s & 0 & & \\ K_{EM1}s & \left(J_{EM}s^2 + c_{EM}s\right) & 0 & & \\ 0 & 0 & \left(J_{TR1}s^2 + c_{TR1}s + k_{TR}R_1^{\,2}\right) & \cdots & \\ 0 & 0 & -k_{TR}R_1R_2 & & \\ 0 & 0 & 0 & & \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \cdots & -k_{TR}R_1R_2 & 0 \\ & \left(J_{TR2}s^2 + c_{TR2}s + k_{TR}R_2^{\,2}\right) & 0 \\ & 0 & m_{VE}s^2 \end{bmatrix} \tag{11}$$

The component input vector $\mathbf{f}$ is composed of input vectors of component models,

$$\mathbf{f} = \begin{bmatrix} \mathbf{f}_{EM} \\ \mathbf{f}_{TR} \\ \mathbf{f}_{VE} \end{bmatrix} = \begin{bmatrix} e_{EM} \\ \tau_{EM} \\ \tau_1 \\ \tau_2 \\ F \end{bmatrix} \tag{12}$$

where e_{EM}, τ_{EM} are the voltage and torque applied to the electric motor, τ_1, τ_2 are the torques applied at the two power ports of the transmission, and F is the force of inertia on the assembly.

The component output vector $\mathbf{x}$ is composed of output vectors of component models:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{EM} \\ \mathbf{x}_{TR} \\ \mathbf{x}_{VE} \end{bmatrix} = \begin{bmatrix} Q_{EM} \\ \theta_{EM} \\ \theta_1 \\ \theta_2 \\ x \end{bmatrix} \tag{13}$$

where Q_{EM} and θ_{EM} are the current and angular displacement at the electric motor, θ_1 and θ_2 are the angular displacements at the two ports of the transmission, and x is the linear displacement of the assembly.

Next, we will set the constraints on the power ports that define the linear mapping between 5 internal variables of $\mathbf{x}$ (Eq. 13) and the external outputs in the assembly $\mathbf{y}$. Number of output ports depends on

the number of constraints as dictated by the assembly output constraint $\mathbf{S}$. These constraints are:

1. The angular displacement output of the electric motor is joined with the first angular displacement output of the transmission:

$$\theta_{EM} = \theta_1 \tag{14}$$

2. The second angular displacement output of the transmission is joined with the displacement of the vehicle:

$$\theta_2 = \frac{x}{r_{TIRE}} \tag{15}$$

Using Eq. 3, the output constraint equation becomes:

$$\mathbf{x} = \mathbf{S}\mathbf{y}$$

$$\begin{bmatrix} Q_{EM} \\ \theta_{EM} \\ \theta_1 \\ \theta_2 \\ x \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1/r_{TIRE} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Q_{EM} \\ \theta_{EM} \\ x \end{bmatrix} \tag{16}$$

and the output vector $\mathbf{y}$ for the assembly matrix is:

$$\mathbf{y} = \begin{bmatrix} Q_{EM} \\ \theta_{EM} \\ x \end{bmatrix} \tag{17}$$

Using the assembly output constraint (Eq. 16) we find the assembly input vector $\mathbf{u}$ as:

$$\mathbf{u} = \mathbf{S}^T\mathbf{f} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/r_{TIRE} & 1 \end{bmatrix} \begin{bmatrix} e_{EM} \\ \tau_{EM} \\ \tau_1 \\ \tau_2 \\ F \end{bmatrix} = \begin{bmatrix} e_{EM} \\ \tau_{ext} \\ F_{ext} \end{bmatrix} \tag{18}$$

where τ_{ext} and F_{ext} are the external torque and force applied on the electric motor – transmission connection and the vehicle, respectively. An example of external torque is the brake torque that is applied for deceleration. F_{ext} input to the model could be utilized as an external force such as the gradient of the road or air drag. In our experiments we only take air drag into consideration as an external force. The external force due to the gradient of the road is $m_{VE} g \sin(\alpha)$ where g is the gravitational constant and α is the gradient angle, and implementing this force in the simulation simply corresponds to summing it with the air drag.

Using the unassembled dynamic modular model and the constraint matrix, we assemble the dynamic modular models of components as shown in Eq. 19.

Note in Eq. 19 that the assembled modular model $\hat{\mathbf{K}}$ is the inverse of the simulation model we will be using in our experiments. In the next section we will introduce a fuel cell stack and a battery into this assembly. Note that once the drive train modular model is obtained, assembly of fuel cell is a simpler process that requires an additional Join operation. However, in the regular operation of the RD-MDM platform, the FRMs of each component will be downloaded separately and sometimes assembled at once.

6.2. Modeling Hybrid Vehicles

We demonstrated how to obtain the FRM for the electric motor - drive train assembly in Sec 6.1. Modeling a fuel cell vehicle simply amounts to adding a fuel cell stack model into this assembly. For details of a fuel cell stack modular model and modeling and simulating fuel cell vehicles, reader is referred to (Eskil 2005, Eskil, Sticklen et al. 2008).

We are undertaking a more challenging problem of modeling a hybrid vehicle in this paper. It is important at this point that our formulation for FRMs

$$\hat{\mathbf{K}} = \mathbf{S}^T\mathbf{K}\mathbf{S} =$$

$$\begin{bmatrix} L_{EM}s^2 + R_{EM}s & K_{EM2}s & 0 \\ K_{EM1}s & (J_{TR1} + J_{EM})s^2 + (c_{TR1} + c_{EM})s + k_{TR}R_1^2 & -\frac{k_{TR}R_1R_2}{r_{TIRE}} \\ 0 & -\frac{k_{TR}R_1R_2}{r_{TIRE}} & \left(m_{VE} + \frac{J_{TR2}}{r_{TIRE}^2}\right)s^2 + \left(\frac{c_{TR2}}{r_{TIRE}^2}\right)s + \frac{k_{TR}R_2^2}{r_{TIRE}^2} \end{bmatrix} \tag{19}$$

works only for linear time-invariant systems. On the other hand, the simulation of a hybrid vehicle has 3 major nonlinearities associated with it. First, we cannot incorporate the depletion of the hydrogen tank or batteries in the overall FRM. Second, we cannot implement gearshift in the assembly model. And third, we cannot model switching between the fuel cell stack and the battery, which should be done according to road and driving conditions. We will deal with the first two of these problems by making the following assumptions:

1. It is assumed that the hydrogen flow and the battery charge are externally controlled (i.e., not embedded in the assembly FRM).
2. The transmission runs in a single mode and does not facilitate gearshift.

Due to these nonlinearities we pursue the design of a fuel cell – battery hybrid vehicle with a combination of assembled and stand-alone modular models. In other words, we utilize the modular model of power train assembly that was derived in Section 6.1 together with decision-making mechanisms to facilitate gearshift and switch battery on and off.

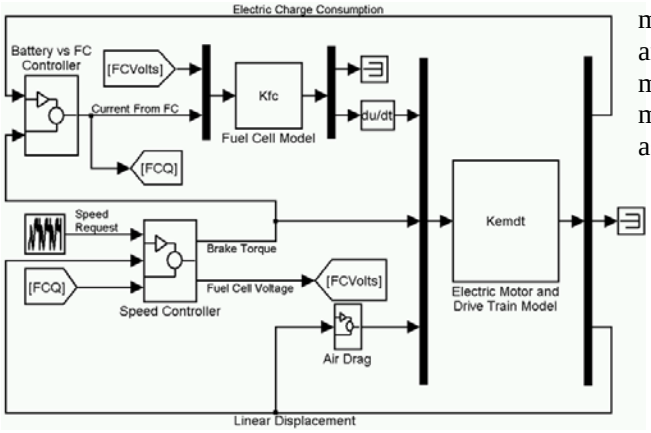

Figure 7 – The MATLAB Simulink model for fuel cell vehicle simulation

We connect the fuel cell and the battery in parallel as two sources of electrical potential. Sources connected in parallel must generate the same potential across the connection nodes to avoid internal and reverse electric flow. In this experiment, we use DC-DC converters coded in the simulation model to balance the potential provided by the fuel cell and the battery. Thus, when both sources are connected in the circuit, the current drawn from each source is exactly one half of the current demand of the electric motor. In real world applications, a decision-making mechanism evaluates the driving conditions for the power demand and connects the fuel cell, battery, or both to the circuit to power the electric motor. This mechanism also decides when the battery will be charged. The Battery vs. Fuel Cell Controller in Figure 7 does this operation by monitoring the position of gas and brake pedals and tracking the charge stored in the battery. One additional task of this module in our experiment is the computation of the hydrogen consumption. The decision-making mechanism in this block is as follows:

1. When the battery charge level is above a set limit and the electric current demand is greater than zero, draw half of the electric charge from battery.
2. When the vehicle is decelerating (brake torque is applied) and the battery is not full, charge the battery with the electric charge that is produced by the back EMF of the electric motor (regenerative breaking).
3. Otherwise, keep the battery idle.

As shown in Figure 7, we were able to develop a simulation model by leaving the battery and fuel cell modular models out of the assembly. However, there are drawbacks of not deriving an assembled modular model for the design, particularly when the design is meant to serve as a component in a higher-level assembly:

1. The recursive structure of modular distributed models is broken. Transfer of unassembled modular models as the product model raises issues with the protection of proprietary knowledge.
2. Modular models are concise descriptors that are simple matrices of transfer functions. A Simulink model is inherently harder to incorporate in an assembly. Moreover, transferring platform-specific models raises concerns about compatibility issues across platforms.

We ran the routine design of a hybrid vehicle with design requirements on acceleration, speed and vehicle range (fuel consumption):

1. Vehicle accelerates from 50 to 70 mph in less than 10 seconds.
2. Vehicle top speed is greater than 100 mph.
3. Requested vehicle range is greater than 120 miles.

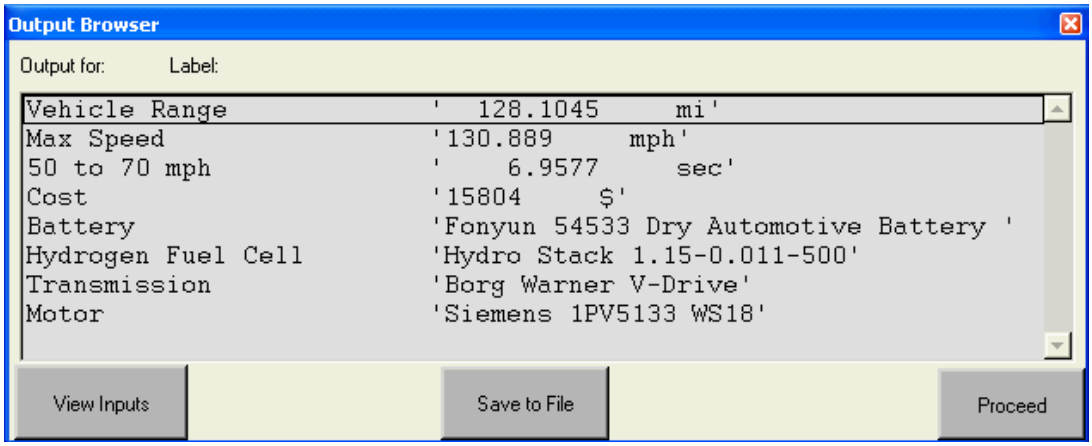

Figure 8 – Fuel cell – battery hybrid vehicle design and simulation results

When the routine designer is run, it follows the process decomposition as dictated by the conceptual design and its plans. This is done through the GT-RD architecture (Sec 4). The component selection process in RD-MDM (Sec 5.1) queries and chooses the suitable design components based on their prices, delivery time, power, efficiency, etc. Next, the component FRMs are requested and the received FRMs are integrated (Sec 5.2) to come up with the assembly simulation model. Lastly, simulations are made and the overall design is tested with respect to the design requests.

We carried out our experiments using an MDM population of 4 electric motor, 4 transmission and 12 fuel cell agents, communicating over 32 predefined queries. These agents were distributed over 3 servers, communicating 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.

When the simulations are over, RD-MDM returns the selected components as well as the simulation results. As shown in Figure 8, although it was not guaranteed for all inputs, routine designer had met the design requirements.

7. CONCLUSION

In this paper we describe a new approach to cooperative design using distributed, off-the-shelf design components. The proposed architecture, RD-MDM is a conceptual framework that supports task directed, distributed Routine Design (RD) including 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 deliverable of our research is a distributed RD platform that is capable of automated multi-attribute search for remotely represented off-the-shelf design components, design parameterization by choosing suitable components for the design, integrating these components in an assembly, running simulations for design testing, and publishing the approved design as an MDM agent.

RD-MDM enables design, virtual assembly and simulation of end products that integrate off-the-shelf components represented by remote supplier agents. 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 non-disclosure agreements. An integrator can also serve as a supplier to other integrators by making its RD-MDM generated product model available in the MDM community for evaluation as a part of higher-level design, without disclosing proprietary design details.

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. As the MDM community grows, RD-MDM will be more beneficial by 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 in manageable levels.

At the current state we are running experiments to test the platform in an online reverse auction setting. In our future papers we will discuss our results and compare the RD-MDM platform with other cooperative design platforms.

References

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

Brown, D. C. and B. Chandrasekaran (1989). “Design Problem Solving: Knowledge Structures and Control Strategies”. San Mateo, California, Morgan Kafumann Publishers, Inc.

Byam, B. P. and C. J. Radcliffe (1999). Modular “Modeling of Engineering Systems Using Fixed Input-Output Structure”. *Symposium of Systematic Modeling*, Orlando, FL, ASME International Mechanical Engineering Congress and Exposition

Byam, B. P. and C. J. Radcliffe (2000). Direct “Insertion Realization of Linear Modular Models of Engineering Systems Using Fixed Input-Output Structure”. *26th Design Automation Conference*, Baltimore, Maryland, ASME

Cutkosky, M. R., R. S. Engelmore, et al. (1993). "PACT: An Experiment in Integrating Concurrent Engineering Systems". *IEEE Computer* **26**(1): 28-37.

Eskil, M. T., J. Sticklen, et al. (2003). “Modular Distributed Modeling”. *4th International Collaborative Technology Symposium*, Orlando, Florida, Society for Modeling and Simulation International**:** D3-202.

Eskil, M. T. (2005). “Distributed Routine Design over the Internet with Collaborating MDM Agents”. PhD Dissertation, Michigan State University.

Eskil, T., Sticklen, J., Radcliffe, C., (2008). “The Routine Design–Modular Distributed Modeling Platform for Distributed Routine Design and Simulation-Based Testing of Distributed Assemblies”. *Artificial Intelligence in Engineering Design and Manufacturing*, vol. 22, no.1, pp. 1-18.

Fin, A. and F. Fummi (2000). “A Web-CAD Methodology for IP-Core Analysis and Simulation”. *IEEE and ACM Proceedings of Design Automation Conference*, Los Angeles, California**:** 597-600.

Hauck, S. and S. Knoll (1998). “Data Security for Web-Based CAD”. *ACM/IEEE Design Automation Conference*, San Francisco, California**:** 788-793.

Kamel, A. and J. Sticklen (1994). “Multiple Design: An Extension of Routine Design for Generating Multiple Design Alternatives”. *Artificial Intelligence in Design*, Netherlands, Kluwer Academic Publishers**:** 275-292.

MacGregor, S. P. and A. I. Thomson (2001). “A Case Study on Distributed, Collaborative Design: Investigating Communication and Information Flow”. *Sixth International Conference on Computer Supported Cooperative Work in Design*, London, Ontario, Canada**:** 249-254.

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”. Ms Thesis, Mechanical Engineering Department, Michigan State University, E. Lansing.

Shakeri, C. and Brown, D. C. (2004). "Constructing Design Methodologies Using Multiagent Systems". *Artificial Intelligence for Engineering Design*, Analysis and Manufacturing 18: 115-134.

Silva, M. J. and R. H. Katz (1995). “The Case for Design Using the World Wide Web”. *ACM/IEEE***:** 579-585.

Spiller, M. D. and A. R. Newton (1997). “EDA and the Network”. *IEEE International Conference on Computer-Aided Design***:** 470-476.

Wallace, D., Yang, E. and Senin, N. (2001). “Integrated Simulation and Design Synthesis”. Cambridge, MA, Center for Innovation in Product Development, Massachusetts Institute of Technology.