



Modular Distributed Modeling

Taner ESKIL¹, Jon Sticklen¹, and Clark Radcliffe²

Intelligent Systems Laboratory, 3115 Eng Bldg, College of Engineering
Michigan State University
East Lansing, MI 48824 USA

Keywords: Internet integration of engineering models, black-box use of Internet engineering models, cooperative design, e-commerce, fixed agent architecture

Abstract: *Modular distributed modeling* is introduced as a novel Internet-based architecture enabling cooperative engineering design, distributed black-box modeling of engineered artifacts, and open, competitive e-commerce. The motivation for modular distributed modeling is that today's engineering and manufacturing communities have been fundamentally altered by two factors: (a) the development of **non-captive** suppliers for large assembly-centered manufacturing firms and (b) the development of reliable, high speed electronic connectivity (the Internet). The first factor should enable a more competitive and efficient marketplace. The second factor should be enabling for rapid collaborative development of engineered devices (auto, aircraft,...). But to leverage these two historical developments, there is a need to develop device simulation models that hide knowledge of how device functionality is achieved. Traditional modeling has centered on bringing together all parts of a model to one location - all codes collocated on one large computer, or on a tightly bound, very high speed internal network of computational engines. The historical changes in the marketplace now need to be reflected in a new type of simulation environment, one that meets the challenge of information hiding. While in the current generation of simulation environments, the base currency is the number of computer cycles, in the new simulation environment the number of Internet packets is the base currency. We discuss a conceptual architecture that supports the modular distributed modeling, our current status to develop methods and tools supporting modular distributed modeling, and directions we are now pursuing.

INTRODUCTION

Modular distributed modeling (MDM) is a novel Internet-based architecture enabling cooperative engineering design, distributed black-box modeling of engineered artifacts, and open and competitive e-commerce in producer/assembler chains. The motivation for modular distributed modeling lies in the realization that today's engineering and manufacturing communities have been fundamentally altered by two important factors: (a) the development on **non-captive** suppliers for large manufacturing firms who largely

act as assemblers and (b) the development of reliable, high speed electronic connectivity (the Internet).

The automobile industry is prototypical of both the old and new manufacturing organizational structure. Formerly, large manufacturing/assembler firms typically had a captive supply chain of smaller companies. The smaller companies produced parts *only* for their captor firm. Today the situation has changed dramatically; the automotive companies act as assemblers of components produced by a global, non-captive, supplier network. In today's typical engineering and manufacturing environments, supplier companies are not captive to any single firm, but rather sell their goods as dictated by the marketplace. On the surface, this change should result in more competition and hence lower pricing and more innovative products. But in order to protect intellectual property, suppliers are very reluctant to share with "window shoppers" the designs of their products without strong and enforceable legal agreements. Putting in place such legal agreements (typically *proprietary information agreements*) requires both money and, more importantly time. It is not uncommon for such agreements to take between six months and a year to put into place. This delay lengthens time-to-market, and requires agreements be in place long before sales agreements are considered. The effect is that although on the surface assembler/supplier chains are open, in effect there is a web of legal agreements that bind specific assemblers to specific suppliers. The bottom line is that the market place is not as freely competitive as one might think, and innovative change from potentially new suppliers is diminished.

The second historical change in today's engineering and manufacturing communities is the availability of reliable, high speed electronic connectivity, i.e., the Internet. The Internet has enabled collaborative design teams to function irrespective of physical distance, rapid part ordering from electronic catalogs, use of distributed design and simulation tools, and a plethora of other beneficial functions. But the advent of the Internet has not as yet given rise to a simulation environment with an architecture that leverages the inherent advantages offered while observing basic constraints imposed by the current wired world.

The leverage offered for modeling and simulation by using the Internet lies in the inherent structure of the Internet. If a company's purpose is to use currently available parts and subassemblies to construct a new or improved artifact, then that assembler company would strongly profit from a wider number of potential suppliers. In a perfectly open marketplace, such broad, and indeed worldwide exposure of all competing suppliers would lead to better parts, offered at lower prices. However, the sticking point lies in selecting a superior part before large lots of the part are purchased. The way around this sticking point is to simulate possible parts from suppliers in the context of the device an assembler is creating. Large scale electronic simulation of entire devices is becoming more common. Notable examples include the often advertised Daimler Chrysler automobiles, and the Boeing wide body 777 aircraft. In both cases, whole device simulation was carried out prior to manufacture. As advanced and complex - and successful - as the whole-device simulations have been, they have been carried out in a traditional simulation environment: all codes including design details for all

1. Computer Science and Engineering Department
2. Mechanical Engineering Department

parts/subassemblies have been available and integrated directly into a simulation model for the whole.

Suppose instead that each part/subassembly of a device were modeled at a distinct location (the site of the company manufacturing the part or subassembly), and that each model were available *via* the Internet. Further suppose that we make a strong distinction between

- the functional capabilities of a device and
- the internal device implementation that gives rise to these capabilities.

An architecture for engineering models that made this distinction in a hard manner would cut through the time-to-manufacture barrier that is imposed currently by the necessity of enacting legal agreements protecting intellectual property because the implementation of the device is hidden from the outside world. Extrapolating, a strong knowledge hiding model architecture would enable “Internet window shopping” for the best alternative to meet functional requirements of a needed part of subassembly in a device under design. The end result for the engineering/manufacturing community would be a more open, more competitive marketplace.

The motivation for research reported here is to develop a conceptual Internet architecture and agent structure that supports such strong knowledge hiding. The next logical step would seem to be to develop a distributed simulation approach in which all companies would hold tightly at their location their own models, but open those models to *interaction* with other models via the Internet. This would 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 two simulation approaches which both yield accurate results, has up to now been the speed a simulation approach engenders. And since simulation approaches to date have all dealt the “all codes available” situation, this speed factor has been pushed down to a the speed with which a single computer (or a tightly coupled cluster) can complete a simulation. Put another way, 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.

But in relation to the Internet world 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: (a) increase Internet packet transmission times or (b) lower the number of Internet packets needed to transfer the needed information that drives the simulation. Path (a) is not feasible practically 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. Path (b) thus is the only feasible solution.

Path (b) as described in the preceding paragraph entails a strong change from the current simulation environments. Our approach is the subject of this paper: *modular distributed modeling*. Below we will discuss (a) relevant other work, (b) the conceptual architecture for MDM, (c) an example from the domain of statics simulation of mechanical structures, and (d) a conclusion in which we discuss our current status and short term goals, as well as the potential impact of this research.

RELATED RESEARCH

Our research is aimed at providing tools and techniques supporting collaborative design in an Internet environment. More explicitly, we target the problem of *model use* in a commercial, distributed environment with the strong information hiding constraint of holding tightly proprietary aspects of a model. In a nutshell, we seek to develop tools and techniques for allowing *sharing of capabilities* of engineered artifacts while manifestly *not sharing the physical implementation* used to achieve those capabilities.

There are many threads of research which overlap with our research. Over the last two decades, substantial improvement in communications technology, particularly that supporting the Internet, has enabled a quantum leap in capabilities for collaborative design tools. As noted by Tian for the area of collaborative CAD/CAM, CAD/CAM models may now be easily shared by individuals geographically distant from one another. Noting the trend toward globalization in manufacturing, Tian goes on to argue the need for *distributed information management systems* (DIMS) on the basis of (a) elimination of design duplication, (b) reduced lead time for design and production, (c) more efficient product shipment, and (d) a resulting increase in competitiveness. Tian concludes that manufacturing enterprises in the future will be information-oriented, knowledge-driven, and for the most part automated around an Internet environment that provides inter-enterprise collaboration. (Tian 2002) One purpose of such connectivity is to support distributed, collaborative design.

On the other hand, some researchers have pointed out difficulties that seem to be inherent in distributed design. Both Huang (Huang 1999) and MacGregor (MacGregor, Thomson et al 2000) have noted obstacles. In KITE (Knowledge Integration and Transfer for Engineering Design), MacGregor reported that distributed knowledge bases tend to be fragmented, that there is a lack of common terminology between teams of expertise, and that there is an unawareness of the *existence* of knowledge that may help to solve a problem. A lack of common terminology needed to access relevant knowledge in an Internet environment is a common reflection.

To help alleviate this problem, researchers seeking to develop models for distributed collaborative design have pursued two lines: strong structuring of the individual nodes of a collaborative design network inside an *agent* wrapper, and the use of a common ontology to provide a base for common vocabulary.

Research on agents has been prodigious over the last decade, so much so that the term *agent* is both overworked and does not have a definition that is universally agreed upon. Agent research draws strongly on the earlier thrust towards object oriented approaches to software systems development. In a strong sense, fixed agent architectures such as the one we are developing can be thought of as strongly object oriented systems. Many in the distributed design community have argued that object oriented design is well mapped conceptually to the distributed, collaborative environment of the Internet. Others have strongly argued that object oriented approaches bring to the design activity heightened productivity and flexibility of design, largely based on reuse properties of the object oriented approach. (Rumbaugh 1997; Dunne 1999)

Agent technology has been applied to a variety of aspects of engineering, such as supply chain management (Fox 1993), manufacturing planning, scheduling and control. Engineering design using agent technology has been addressed in a more limited number of projects (Park 1999). For example, LEGEND (Stephens 1993) sought to provide an infrastructure to enable design experiments prior to actual production. LEGEND, like almost all later research we have accessed, does not address any need to hold proprietary information.

A common goal for distributed, collaborative design is to enable a distributed design team to work with one another towards common design

goals although not geographically collocated. A thread along this line is to share computational tools for design. For example, DIDE (Shen 1996) is a multiagent environment whose objective is to incorporate engineering tools such as CAD/CAM or knowledge-based systems in a common and freely accessible system.

In contrast another thread of agent-based research seeks not to provide an Internet-accessed set of computational tools for a targeted design effort, but rather to develop a multiagent, distributed architecture in which computational tools are located locally, and in which the design task is distributed by distributing (as agents) the components of the device under design. An example of this type of research is the MetaMorph II project (Shen 2000).

The research projects described in the above paragraphs leverages the broad area of agent research to address the means to structure an ensemble of distributed, collaborative software units for purposes of design. But given a loose architecture for structuring such an ensemble the problem remains of providing some means of enabling communication among the *players*. Two approaches in general have been followed. In one approach to enabling communication among design agents, the tack has been to provide a common base language, and transducers which translate both from this language into the specific language of a given agent, and vice versa. This transducer approach is taken by Sun et al (Sun, Zhang et al 2001). Although this approach has some benefits, the main drawback is that for any new design agent to be added to a distributed cooperative ensemble, the construction of appropriate transducers would be necessary.

A different approach was taken in the SHADE project (McGuire 1993) and the PACT project (Cutkosky, Engelmores et al 1993). These studies seek to integrate frameworks of engineering tools that are developed for specific engineering disciplines. SHADE focused mainly on information sharing and PACT was developed based on the infrastructure developed in SHADE. In PACT, the engineering tools are designed as agents that facilitate collaborative distributed design. Most significantly for purposes here, transfer of knowledge among agents was enabled by a *common ontology* that is system wide.

The studies noted above fail to address to the role of proprietary resources. Hiding proprietary information is noted in IMAGE (Hale 1994): proprietary resources are defined as stand-alone in nature, with limited communications capabilities.

Although steady and significant progress has been made in developing and implementing architectures for distributed, collaborative design, no projects to date have focused frontally on the issue of providing a design and simulation environment in which agents represent physical devices, each agent has a local resource that is a simulation model of the device it represents, and each agent can be linked as a subagent (a component) of an encompassing agent. Although some research has included information hiding, the items hidden have not included the simulation model of the represented device. This leads us to a discussion of the explicit goals for an architecture which will met the challenge of hiding proprietary simulation models, and of the MDM architecture we have developed to meet those goals.

CONCEPTUAL ARCHITECTURE FOR MODULAR DISTRIBUTED MODELING

We seek to develop an infrastructure capable of supporting a community of MDM agents. The ensemble of MDM agents should be capable of supporting Internet based cooperative engineering design and simulation, but with the constraint of device knowledge hiding. The conceptual architecture for MDM can be described in terms of a specific list of capabilities we are developing. The conceptual building block to enable modular distributed modeling is the MDM agent: a fixed but autonomous agent. For MDM agents we desire the following features and capabilities.

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 devices³) 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
 - 6a. on a *quality* of the device held by the MDM agent or
 - 6b. on a *functional response* of the device held by the MDM agent.
7. Each MDM agent contains the means to effectively answer those valid queries put to it which it chooses to answer, where those means are broadly of two types as follows:
 - 7a. the necessary capacity to develop a flexible and situation specific plan for answering any given query and
 - 7b. knowledge and computational resources to execute its plan for a specific query.
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 “devices for mechanical structures.”

Capabilities 4 and 5 express the need for structured communication. In order to conceive, design, and implement the infrastructure for the Internet community for cooperative design in an e-commerce setting and equally important, to gain acceptance within the manufacturing and engineering communities, communication MDM agents must be demonstrably fail-safe, and to those ends structured inter-agent 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 an inherent property of an MDM agent’s device. Examples include weight and size qualities.

3. A complex device would tend to have its own MDM agent: 1:1 map between MDM agent and device. However more simple devices with great similarity to each other (but all vended by one company) would likely share a single brokering agent.

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. 6a is relatively straightforward, but to achieve 6b in a feasible manner in the Internet environment, we have developed a novel approach that will be described in more detail, and explicated further by example below in the concluding section.

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 is 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 tightly. This flexibility allows members participating in an MDM agent network the ability to make available information to potential buyers in a selective manner.

Capability 7a sets the need for an MDM agent to be more than simple Internet buffers between the world and internal device knowledge for the device held by the MDM agent. Depending on speed of desired response, desirable level of detail that is requested, and other factors, the MDM agent must determine how to use knowledge and computational resources at its disposal to respond to a given query. In short, an MDM agent must be at top level a planning agent whose task is to develop and execute a plan given set resources. Capability 7b is a statement that an MDM agent must include organization and structure for the resource base used to answer queries.

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, ... A drive train is further composed of transmission system, differential system, ... And so on. Hierarchical decomposition is the means by which engineers (and engineering as a broad field) handles 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. But each of those off-the-shelf items may itself 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 simulation techniques is the bedrock motivation for our research.

The sketch shown in Figure 1 depicts three MDM agents connected physically *via* the Internet. The sketch also drives home the point made at the end of the preceding paragraph: 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/assemblies⁴ of other MDM agents. The other salient point to note about the sketch of Figure 1 is that there are two additional MDM resources which have not been discussed to this point. The two new resources are the *Query Typology* and the *Agent Registry*. Referring back to the list of capabilities enumerated at the start of this section specifically Capability 4, the Query Typology is a MDM network resource that makes available to any MDM agent the typology of legal queries. The Agent Registry meets Capability 3, and is a MDM network resource that makes available to any MDM agent both the typology of agent types and the list of all existing MDM agents by agent type. Of particular importance in Figure 1 is two types of connections between MDM

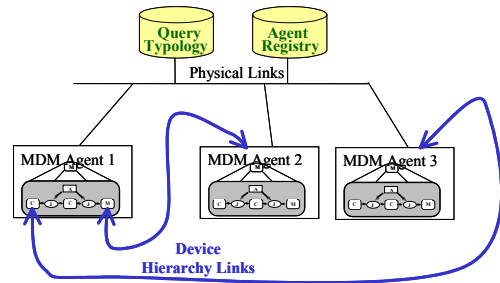


Figure 1: Sketch of MDM agent community. Note the Query Typology Server, and the Agent Registry

agents. Both the physical connection via the Internet is sketched, and the linkage shown in blue in Figure 1 that represents device hierarchy relationships.

In a forthcoming publication we will discuss all aspects of MDM. Each of the capabilities enumerated in the list at the start of this section carries with it important background for the MDM project. Space here however precludes a complete description.

We focus in the remainder of this paper on one aspect of our work leading to realization of an MDM agent community. This aspect is in some ways the most critical step in our work and the one that is most novel. The step is the development of a methodology and computational mechanism that allows us to achieve Capability 6b ... developing the means to produce a *functional response* of the device held by the MDM agent. Some discussion here is necessary before we move on to discuss background work in this area and an example that shows our current capability.

We start with an observation: traditional methods of device simulation will *not* be effective in an Internet, distributed environment. 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, very high speed environment, this may be effective. In the current Internet environment, or even in the "Internet 2" environment, it will not be effective because of the enormous load on network traffic that would be entailed. Consider for example 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 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.

In the next section we discuss one specific type of functional response: functional response answers to queries of mechanical structures in the domain of structural analysis. It is important to note that we have definitive

4. 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 it is important to remember that only device qualities and functional response 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.

proof of principle results for functional response only for mechanical structures in the domains of statics analysis and dynamics analysis. There are many more domains yet to be covered. But our current results encourage us to believe that the general goal of developing functional response methodology for engineered devices is achievable. 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 heart of our approach is the *functional response*. The example sketched below is a proof of principle result that developing functional response methodology is feasible explicitly in the structural analysis domain.

A FUNCTIONAL RESPONSE EXAMPLE – STRUCTURAL ANALYSIS

Structural Analysis is engineering assemblies is an important part of modern mechanical engineering design. This effort is commonly supported by large numerical software packages for Finite Element Analysis (FEA), Boundary Element Analysis (BEA) and various other methods. A common result of such analyses is a structural model that generates a full structural stiffness model based on some discretization of the device’s structural geometry. When these techniques are used to develop structural models of an assembly of substructures, common practice has been to assemble those substructure models locally, including all their internal geometry. When component suppliers provide substructures, it is exactly those internal geometry details that are the proprietary details of their designs that must be protected.

The fully detailed FEA model of structural component and its geometry may be large as well as proprietary. Automotive body FEA models typically often include $N = 20,000$ to $40,000$ degrees of freedom yielding a linear stiffness models \mathbf{K}_p with millions (N^2) of entries.

$$\mathbf{K}_p \mathbf{y} = \mathbf{u} \quad (\text{EQ 1})$$

Access to these large, detailed, proprietary models or their equivalent structural geometry is the current mode of distributed structural design. We propose to change that structural engineering practice.

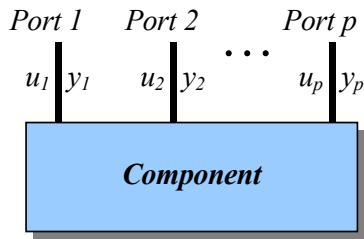


Figure 2: Modular Modeling Element Graphical Notation. Power flows into port i if u_i and y_i are both positive.

Figure 2 shows a diagram of a contracted, functional response, modular input-output model for an internet design agent model. The component’s algebraic model is in the standard stiffness form reduced to include only those degrees of freedom where the component interacts functionally with other structural components. This model is both smaller and non-proprietary. If the component interacts through $m = 20$ degrees of freedom, this functional response structural model has a linear stiffness \mathbf{K} with $N^2 = 400$

entries. This size reduction means far fewer terms are for the non-proprietary structure functional response than that required for the fully detailed, proprietary, structural response model above. Additionally, the reduced model does not reveal the internal topology of the component making reverse engineering of the component’s internal details more difficult.

The procedure for an agent to develop the model of an assembly from the models provided by the from models starts with the independent component structural response models provided by component agents,

$$\begin{aligned} \mathbf{K}_1 \mathbf{y}_1 &= \mathbf{u}_1 \\ \mathbf{K}_2 \mathbf{y}_2 &= \mathbf{u}_2 \end{aligned} \quad (\text{EQ 2})$$

where \mathbf{K}_1 and \mathbf{K}_2 are the modular component stiffness matrices, \mathbf{y}_1 and \mathbf{y}_2 are the component output displacement vectors and \mathbf{u}_1 and \mathbf{u}_2 are the component input vectors. In general, the component stiffness matrices are 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.

The components are grouped and assembled using modular modeling connector constraints (Byam & Radcliffe, 2000) that require both identical output response at connections and work conservation through the connection.

$$\hat{\mathbf{y}}_i = \hat{\mathbf{y}}_j = \mathbf{y}^c \quad (\text{EQ 3a})$$

$$\hat{\mathbf{W}}_i = \hat{\mathbf{W}}_j = \hat{\mathbf{W}}^c \quad (\text{EQ 3b})$$

where $\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j$ are the output displacements connected between the i^{th} and j^{th} components and $\hat{\mathbf{W}}_i, \hat{\mathbf{W}}_j$ are work done on the components by the connection.

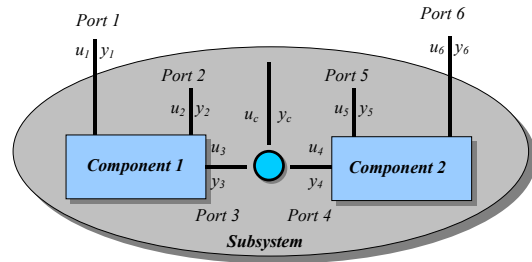


Figure 3: Subsystem Model With Two Components Each With External, Internal And Connected Ports

The Subsystem model (Figure 3) is a demonstration of the possible situations that can arise when components are assembled into subsystems. The subsystem model has two components connected via constraints on port variables on ports 3 and 4. In this case, external connection inputs u_c are assumed zero. The subsystem has internal component ports 2 and 5 not connected externally. Finally, the assembly has port 1 and 6 that can be connected externally. Once assembled, a new algebraic equation set in the form

lation of that line covering linear statics and dynamics problems appeared in (Byam and Radcliffe 2000a), and in a companion work cases of non-linear models were covered (Byam and Radcliffe 2000b).

Our short term goals focus on three areas:

1. the extension in mechanical engineering to cover questions of dynamics functional response,
2. adding the capability of MDM agents to do more robust response planning given available knowledge and computational resources, and given constraints for time bounds on response and fidelity of response, and
3. the development of more "real world" MDM agents by interaction with collaborating companies.

We expect to meet these goals in the next year to 18 months.

The move into our third generation implementation was motivated by Short Term Goal 2 above. Smalltalk is the base computational environment we are using. But on top of Smalltalk, we are using the MSU Intelligent Systems Laboratory suite of knowledge-based systems tool. This toolset is an embodiment of the "generic task" concept for knowledge-based systems which has been applied successfully to a number of diverse problem domains including modeling managed ecosystems, production agriculture, and most recently developing support for design and manufacturing in the polymer composite materials area, and in collaboration with the Composite Materials and Structures Center of Michigan State University. We expect to apply the lessons we have learned across these domain to the problem of flexible knowledge and computational resource use in the MDM agents project.

We expect the impact of successfully meeting our long terms goals in the MDM project to be felt in two areas. First, and most directly, we expect the impact could be profound for design and engineering commercial communities. As argued in the introductory section, a major bottleneck in the path to develop a more open and competitive marketplace for engineering and manufacturing communities is the necessity to protect intellectual property. The concept of modular distributed modeling was motivated by this problem, and successful long terms results from our research should mitigate the problem substantially, if not totally solve the problem.

Second, from a more focused computer science viewpoint, MDM in general, and the concept of functional response in particular, is based on the realization that the world of cooperative computing has changed because of the Internet. There are situations in Internet-based distributed computing in which the computational power of individual machines is still determinant. A specific example would be distributed computation with genetic algorithms. In this case a large number of machines are given identical software (some version of GA software) and at each new generation of the GA, problems are given out to the individual machines while some central machine waits for responses. In this type of distributed computing the power of individual machines is clearly determining for how fast a generation can be run through.

However in the more general case, cooperation among computer systems on the Internet does not entail running identical software on differing inputs. The more general case would have more similarities to the problem we are attacking in the MDM project. Such situations call for the strong realization that it is no longer raw computer speed that "makes a difference" but rather it is speed of Internet transmission that is the determining factor for system response, and in the end for system effectiveness.

REFERENCES

- Byam, B. P. and C. J. Radcliffe (2000a). Direct Insertion Realization of Linear Modular Models of Engineering Systems Using Fixed Input-Output Structure. ASME Design Engineering Technical Conferences, Baltimore, MD.
- Byam, B. P. and C. J. Radcliffe (2000b). Direct Insertion Realization of Non-linear Models of Engineering Systems Using Fixed Input-Output Structure. 2000 ASME International Mechanical Engineering Congress and Exhibition, Orlando, FL.
- Byam, B. P. a. R., C.J. (1999). Modular Modeling of Engineering Systems Using Fixed Input-Output Structure. Symposium of Systematic Modeling, Orlando, FL, ASME IMECE.
- Cutkosky, M. R., R. S. Engelmire, et al. (1993). "PACT: an experiment in integrating concurrent engineering systems." IEEE Computer **26**: 28-37.
- Dunne, P., Gray, A. (1999). "The impact of using class inheritance in a distributed information system." Advances in databases and information systems **1691**: 332-348.
- Fox, M. S., Chionglo, J.F., and Barbuceanu, M. (1993). The Integrated Supply Chain Management System, Dept. of Industrial Engineering, Univ. of Toronto.
- Hale, M. A., Craig, J.I. (1994). IMAGE: A Design Integration Framework Applied to the High Speed Civil Transport. 1st Industry/Acedemy Symposium on Research for Future Supersonic and Hypersonic Vehicles, NC.
- Huang (1999). "Knowledge sharing and innovation in distributed design: implications of internet-based media on design collaboration." International Journal of Design Computing: Special Issue on Design Computing on the Net (DCNet99).
- MacGregor, S. P., A. I. Thomson, et al. (2001). A case study on distributed, collaborative design: investigating communication and information flow. Sixth International Conference on CSCW in Design, London, Ontario.
- McGuire, J. G., Kuokka, D. R., Weber, J. C., Tenenbaum, J. M., Gruber, T. R., Olsen G. R. (1993). "SHADE: Technology for Knowledge-Based Collaborative Engineering." Concurrent Engineering: Research and Applications **1**: 1-17.
- Park, H. a. C., M. R. (1999). "Framework for modeling dependencies in collaborative engineering processes." Research in Engineering Design **11**(2): 84-102.
- Radcliffe, C. J. and J. Sticklen (2001). Method and System for Creating Designs Using Internet-based Agents. USA, Board of Trustees, Michigan State University.
- Radcliffe, C. J., Sticklen, J. and Gosciak, G. (2002). The Internet Engineering Design Agent System: i-EDA. Proceedings 2002 ASME Internat'l Mech. Engineering Congress and Exhibition, New Orleans, LA.
- Rumbaugh, J., Blaha, M., Premerlani, W. and Eddy, F. (1997). Object-Oriented Modeling and Design. Englewood Cliffs, NJ, Prentice-Hall.
- Shen, W., Maturana F. and Norrie, D.H. (2000). "An Agent-Based Architecture for Distributed Intelligent Design and Manufacturing." Journal of Intelligent Manufacturing - Special Issue on Distributed Manufacturing Systems **11**(3): 237-251.
- Shen, W. a. B., J.P. (1996). "An Experimental Multi-Agent Environment for Engineering Design." International Journal of Cooperative Information Systems **5**(2-3): 131-151.
- Stephens, E. (1993). LEGEND: Laboratory Environment for the Generation, Evaluation and Navigation of Design. Doctoral Dissertation, School of Aerospace Engineering, Georgia Institute of Technology.
- Sun, J., Y. F. Zhang, et al. (2001). "A Distributed Multi-Agent Environment for Product Design and Manufacturing Planning." International Journal of Production Research **39**(4): 625-645.