ABSTRACT

Approaches to engineering design and manufacturing such as integrated design and manufacture and just in time fabrication depend on interaction with and among component supply companies that most often use very diverse technologies. The Internet Engineering Design Agents (i-EDA) software system uses a distributed, component-based, agent methodology that is realized following a strong black box approach to modeling. An individual Design Agent (DA) is a virtual product capable of encapsulating both descriptive and model-based information about the product it represents. Hierarchically recursive agents for sub-systems and/or components are linked via a communications network to form larger integrated model systems. A two-dimensional bridge system structural model is used as an example to illustrate the distributed assembly of structural models from components registered as DAs on a communications network.

Modular Distributed Modeling (MDM) of engineering structures performs static deflection analysis using traditional, fixed causality, structural stiffness models. This paper presents the methodology required to assemble traditional structural stiffness models provided by Internet agents representing structural components. The methodology discussed assembles these component models into the structural stiffness model of an assembly distributed by an agent on the communications network. Using this modular distributed modeling method; models of complex assemblies can be built while hiding the topology and characteristics of their structural subassemblies.

The automated, modular assembly of structural stiffness models will be derived for discrete physical connections. Discrete connections are important to the assembly of components such as truss and shaft structures where the relationship between component displacements involve discrete, matching degrees of freedom on components to be assembled. Specific examples of discrete assembly of truss bridge component models will be presented. Internet connection permitting, real-time, automated assembly of models and deflection analysis will be performed.

INTRODUCTION

Modular Distributed Modeling (MDM) is a novel Internet-based architecture enabling cooperative engineering design, distributed black-box modeling of engineered products, and open and competitive e-commerce in producer/assembler chains (Radcliffe and Sticklen, 2001). 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 of 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. This time 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 engineering modeling and 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 component parts and subassemblies to construct a new or improved product, 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 in the contemporary press include 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 parts/subassemblies have been available in-house and integrated internally 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 external functional capabilities of a device and
- the internal device design providing these capabilities.

An architecture for engineering models making 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 design 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 modeling and simulation approach in which all companies would hold tightly at their location their own designs, but open those design functional response models to interaction with other models via the Internet.

Analysis speed is often the deciding factor between two simulation approaches which both yield accurate results. In localized, in-house, modeling and simulation approaches, speed is primarily determined by the number of arithmetic operations required of an analytical procedure. In the internet world where network data transfer is orders of magnitude slower than transfer within a single computer, speed must be evaluated with a different metric – the internet traffic required of the approach.

The Internet Engineering Design Agent (i-EDA) software system (Figure 1) has the four (4) features important to the success of distributed modeling and simulation approach. The i-EDA system uses the Modular Distributed Model software agent architecture that:

a) allows hierarchical assembly of component functional capabilities from the functional capabilities of components,

b) uses standardized canonical forms for the external representation of assembly and component functional capability,

c) limits internet traffic by avoiding iterative access to component capability models, and

d) provides access to component capabilities through centralized resources for agent locations and query format.

Figure 1 shows a “User Client” that uses an “Agent Registry” to find the locations of “Assembly” and “Component” Agents on the network. The Client uses standardized queries whose topic and format are obtained from the “Query Ontology” which publishes an organized list of these queries to the internet. All product agents, whether assemblies or components, respond to queries with functional

![Figure 1: The Internet Engineering Design Agent Software System: i-EDA](image-url)
responses defining their engineering capabilities in the standard (canonical) format defined by the system’s ontology. As shown, the Assembly agent joins the functional capability responses of its components to form its own functional capability response.

This paper will detail the method required to develop the structural functional response of a structural assembly. To avoid network traffic, the assembly must accomplish this process without iteration. To allow hierarchical modeling of higher level assemblies, all product agents – components and assemblies - must respond with identically formatted functional responses. Our approach for building functional capability responses is modular distributed modeling.

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.

RELATED RESEARCH

There are many threads of research that 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. Collaborative CAD/CAM, CAD/CAM models may now be easily shared by individuals geographically distant from one another. With the trend towards globalization in manufacturing, there is a need for distributed information management systems to (a) eliminate design duplication, (b) reduce lead time for design and production, (c) provide more efficient product shipment, and (d) increase competitiveness. 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. One purpose of such connectivity is to support distributed, collaborative design.

In contrast, 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 multi-agent 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, Engelmore 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 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. Gu, B., Asada, H.H., and He, X.D., (2002) provide a more general discussion of the need to protect proprietary model information during an integrator’s engineering design analysis.

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 meet the challenge of hiding proprietary simulation models, and of the MDM architecture we have developed to meet those goals.

The reasons for and foundations of the Modular Modeling method the direct precursor to MDM were discussed in (Byam and Radcliffe 1999). Coverage included static Kx=F and dynamic (state-space) models. Extrapolation of that line covering linear statics and dynamics appeared in (Byam and Radcliffe 2000a), and in a companion work cases of non-linear models were covered (Byam and Radcliffe 2000b). More recent work setting out the design strategy for Modular Distributed Modeling agents is provided by Eskil et al (2003).

MODULAR MODELING

Modular modeling software agents on the Internet facilitate exchange of modeling information but require both standards for information interchange and protection of the proprietary data underlying that model information. We accomplish these requirements by to providing compact, simplified models in response to queries. These models are contraction mappings of the engineering modeling data that underlies the model. Algebraically coupled structural components are addressed in this work but similar formulations for structural dynamics are in development. This paper will discuss both a standard form for structural models and the contraction mapping of engineering proprietary model data into a contracted structural model that can be distributed without revealing the topology and components of the model from which it is extracted.

Work and Power conservation is the key concept in Modular Modeling (Byam and Radcliffe, 2000). Figure 2 shows three modular modeling components connected by a “join”. The “join” connects structural components through the application of two modular modeling conditions: equal output displacement and conservation of work. These conditions are enforced using the two port variables available at structural connections: force \( f_j \) and displacement \( u_i \).

The join displacement constraint specifies that all \( k \) output displacements for \( m^{th} \) join are identical.

\[
\begin{align*}
  u_{m,1} &= u_{m,2} = \cdots = u_{m,k} = u_m 
\end{align*}
\]  

where \( u_m \) is the output displacement from the \( m^{th} \) join and \( u_{m,k} \) is the \( k^{th} \) component’s output displacement connected at the \( m^{th} \) join. This condition requires conversion of component output displacements to “join” coordinates to maintain compatible units.

Conservation of the work done on the join requires that work is conserved in each connection. In such a “join” connection (Fig. 1), work is defined such that internal work is positive on each component while external work on the “join” is positive

\[
\begin{align*}
  \sum_j W_{mj} &= \sum_{j=1}^{k} (u_{mj}f_{mj}) = u_m f_m = W_m 
\end{align*}
\]  

where: \( u_{m,j} \) is the \( j^{th} \) port’s output displacement at the \( m^{th} \) join

\( f_{mj} \) is the \( j^{th} \) port’s input force at the \( m^{th} \) join

\( W_{mj} \) is work transferred into the \( j^{th} \) port at the \( m^{th} \) join

\( W_m \) is the external work applied on the \( m^{th} \) join

\( u_m \) is the \( m^{th} \) join’s output displacement, and
The signs chosen here result in positive work in identical directions for both components and assemblies of components. Identical work and displacement sign definitions are required for modular assembly that is independent of the internal structure of subassemblies.

Substituting the output displacement constraint (1) into the work conservation constraint (2) yields the input force condition for the \( m \)-th join,

\[
\sum_j W_{m,j} = \sum_{j=1}^{k} \left( u_{m,j} f_{m,j} \right) = \sum_{j=1}^{k} u_{m,j} f_{m,j} = u_m \sum_{j=1}^{k} f_{m,j} = u_m = W_m
\]

Factoring the constraint displacement \( u_m \) yields the required relationship between the forces involved in the join

\[
\sum_{j=1}^{k} f_{m,j} = f_m
\]

The relationships (1) and (4) form the set of constraints required to conserve work (2) on the “join.” These relationships allow the join to assemble the \( k \) components into a work conserving assembly.

**STRUCTURAL COMPONENT MODELS**

A modular modeling structural component (Figure 3) has \( p \) work ports with implicit, standardized, direction of positive work into the element and an internal algebraic representation that standardizes input-output variable causality. Standardization of positive work and input-output causality standardizes the modular modeling elements internal formulation.

The structural response model of a component is in the standard stiffness form.

\[
Ku = f
\]

where \( K \) is the component stiffness matrix, \( u \) is the component generalized displacement vector and \( f \) is the component input force vector. In general, the component stiffness matrix is singular and cannot be inverted. This situation occurs because component models have zero eigenvalues from “rigid-body modes” from components with no applied boundary conditions.

The subsystem model (Figure 4) 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. The subsystem has internal component ports 2 and 5 not connected externally. Finally, the assembly has port 1 and 6 that could be connected externally. Once assembled, a new algebraic equation set in the form (5) 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.

**ASSEMBLING STRUCTURAL MODELS**

The equations for each of the components are first assembled into the unconstrained system matrix.

\[
\begin{bmatrix}
K_1 & 0 \\
0 & K_2
\end{bmatrix}
\begin{bmatrix}
u_1 \\
u_2
\end{bmatrix}
=
\begin{bmatrix}
f_1 \\
f_2
\end{bmatrix}
\]

The subsystems components are uncoupled in this form. Coupling of the subsystems will use constraints (1). The expanded subsystem equations

\[
\begin{bmatrix}
k_{11} & k_{12} & k_{13} \\
k_{21} & k_{22} & k_{23} \\
k_{31} & k_{32} & k_{33}
\end{bmatrix}
\begin{bmatrix}
u_1 \\
u_2 \\
u_3
\end{bmatrix}
=
\begin{bmatrix}
f_1 \\
f_2 \\
0
\end{bmatrix}
\]

have the connection constraints applied to generate a set of subsystem equation written in terms of the subsystem’s external port variables only.

The application of the constraint equations to join the components occurs in two steps. First the force input constraint
The input constraints allow the removal of component inputs \( f_1 \) and \( f_2 \) from the subsystem equations through the algebraic addition of the two port equations comprising the connection.

**Application of displacement output constraint (1)** removes internal join displacements by replacing them with the join’s displacement output. In this case, \( u_5 = u_4 = u_7 \) so that

\[
\begin{bmatrix}
  k_{11} & k_{12} & k_{13} & 0 & 0 & 0 & u_1 \\
  0 & 0 & 0 & k_{45} & k_{46} & k_{47} & u_2 \\
  k_{31} & k_{32} & k_{33} & k_{44} & k_{45} & k_{46} & u_3 \\
  0 & 0 & k_{64} & k_{65} & k_{66} & k_{67} & u_4 \\
  0 & 0 & 0 & k_{64} & k_{65} & k_{66} & u_6 \\
\end{bmatrix}
\begin{bmatrix}
f_1 \\
f_2 \\
f_3 \\
f_4 \\
f_5 \\
f_6 \\
\end{bmatrix}
= \begin{bmatrix}
f_7 \\
f_5 \\
\end{bmatrix}
\]

The contracted model equations in the standard form (5) are obtained substituting (11) into the external port equations in (10),

\[
\mathbf{K_{ext}} \mathbf{u_{ext}} = \mathbf{f_{ext}}
\]

where \( \mathbf{K_{ext}} = [\mathbf{A} + \mathbf{BD}^{-1}\mathbf{C}] \)

In our example, the assembly equations have three internal port equations to be defined with homogeneous force input in the final assembly,

\[
f_2 = f_5 = f_7 = 0
\]

Each of the (3) conditions above allow a corresponding equation in the assembly model to be used to find displacement output variables \( u_2, u_5 \) and \( u_7 \) to be written in terms of the other assembly variables. Parsing the model equations (9)

\[
\begin{bmatrix}
k_{11} & 0 & k_{12} & k_{13} & 0 & 0 & u_1 \\
0 & 0 & 0 & k_{45} & k_{46} & k_{47} & u_2 \\
k_{31} & k_{32} & k_{33} & k_{44} & k_{45} & k_{46} & u_3 \\
0 & 0 & k_{64} & k_{65} & k_{66} & k_{67} & u_4 \\
0 & 0 & 0 & k_{64} & k_{65} & k_{66} & u_6 \\
\end{bmatrix}
\begin{bmatrix}
f_1 \\
f_2 \\
f_3 \\
f_4 \\
f_5 \\
f_6 \\
\end{bmatrix}
= \begin{bmatrix}
u_1 \\
u_2 \\
u_3 \\
u_4 \\
u_5 \\
u_6 \\
\end{bmatrix}
\]

and using the contraction condition (12) yields the contracted structural assembly model yields

\[
\mathbf{K_{ext}} \mathbf{u_{ext}} = \mathbf{f_{ext}}
\]

where: the condensed assembly stiffness \( \mathbf{K_{ext}} = \begin{bmatrix} \hat{k}_{11} & \hat{k}_{12} \\ \hat{k}_{21} & \hat{k}_{22} \end{bmatrix} \),

with \( \hat{k}_{11} = (k_{11}k_{32} + k_{31}k_{12})/k_{32} \)

\[
\hat{k}_{12} = \frac{1}{k_{32}} \left[ k_{12}k_{45}k_{46} - k_{12}k_{46}k_{45} + k_{13}k_{32}k_{46} + k_{13}k_{31}k_{45} + k_{13}k_{31}k_{46} + k_{13}k_{32}k_{45} \right]
\]

\[
\hat{k}_{21} = (k_{21}k_{32} + k_{22}k_{31})/k_{32}
\]

and

\[
\hat{k}_{22} = \frac{1}{k_{32}} \left[ k_{23}k_{45}k_{46} - k_{23}k_{46}k_{45} + k_{23}k_{33}k_{46} + k_{23}k_{33}k_{45} + k_{23}k_{31}k_{46} + k_{23}k_{32}k_{45} \right]
\]

**AN I-EDA EXAMPLE**

The functional response of a simple 3-bar truss (Fig. 5) can be computed from the functional responses of its component bars using the above methodology. Each of the bars has a functional response model at \( \theta = 0 \) of the form,

\[
k \begin{bmatrix}
  1 & 0 & -1 & 0 \\
  0 & 0 & 0 & 0 \\
-1 & 0 & 1 & 0 \\
 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
u_{1X} \\
u_{1Y} \\
u_{2X} \\
u_{2Y}
\end{bmatrix}
= \begin{bmatrix} f_{1X} \\
 f_{1Y} \\
 f_{2X} \\
 f_{2Y}
\end{bmatrix}
\]
where:  
\[ k = 1.47 \times 10^8 \text{ N/m for bar 1,} \]
\[ k = 1.47 \times 10^8 \text{ N/m for bar 2, and} \]
\[ k = 10.5 \times 10^8 \text{ N/m for bar 3,} \]

Each of the three bars in the assembly is 2 m long. Before assembly, each bar stiffness model is rotated by the appropriate angle (0°, 60°, and 120°, see Fig. 5) to form component stiffnesses. The component stiffnesses are then assembled using (1) and (4). To solve for the deflection of the assembly, boundary conditions \( u_{1x} = u_{1y} = u_{2y} = 0 \) are applied. For an applied external force \( f_{3y} = 5,000 \text{ N} \), the truss’s deflections are shown in Fig. 6.

In this example of the i-EDA software, the user interface (Fig. 6) is used to query agents provided by product manufacturers. This user interface software queries the “ACME Truss” internet agent located at the assembly manufacturer for the model stiffness for the Acme Truss part TAx2. Acme Truss’s truss agent assembles the stiffness of the TAx2 assembly through queries to agents representing each of its component bars and builds the stiffness functional response of the assembly. The resulting assembly stiffness functional response is sent by the truss agent to the user interface and, in this case, is displayed there. Examining the truss stiffness functional response does not allow the user to determine the component’s functional responses. In this example, the user interface software then enforces boundary conditions, applies loads and solves for truss deflection.

**CONCLUSION**

The modular modeling analytical procedure for structural models allows an internet agent to retrieve a structural model in the standard form (5), process it and produce a compact modular model in the standard form (5). The analytical modular model process implements the connecting “joins” between assembly components, contracts the model to remove internal ports not required at higher levels, and produces a modular structural model. This derived modular model has two important properties that allow for further hierarchical, modular, assembly models.

a) The model is of a standard, canonical, form, and
b) the model is a minimal the input-output model that does reveal internal topological detail.

Applications for modular structural modeling include all structural models that can be put into the standard, canonical, “stiffness matrix” form (5). These models include most linear finite element models as well as models of structures with discrete compliant components. Further work will develop methods for assembling algebraic structures with distributed connections where one-to-one port topography is not either possible or desirable.

Distribution of modular models over the internet has been implemented in the Internet-based Engineering Design Agents (i-EDA) test bed at Michigan State University. This system uses the modular structural model analysis to assemble structural truss systems with up to 9 subsystems and components using a single set of queries to independent model agents published on the internet. The ability to assemble models using minimum internet traffic while protecting internal engineering detail makes possible the world-wide system of standardized models of components necessary for networked, collaborative, engineering design.

**REFERENCES**


International Mechanical Engineering Congress and Exhibition, Orlando, FL.


Huang (1999). "Knowledge sharing and innovation in distributed design: implications of internet-based media on design collaboration."


8 Copyright © 2003 by ASME