Computer supported collaborative engineering

D. Sriram
Intelligent Engineering Systems Laboratory
Department of Civil and Environmental Engineering
Massachusetts Institute of Technology
Cambridge MA 02139
USA

1 Introduction

Engineering projects, in general, involve a large number of components and the interaction of multiple technologies. The components included in the product are decided in an iterative design process. In each iteration, interfaces and interface conditions among these components are designed with slack to account for potential variations created when the components and interface values become better known. Iteration proceeds towards increasing detail; design personnel may change, and their numbers expand with increasing level of detail. This is true for both large systems and small systems.

The above multi-faceted nature of engineering problems demands considerable communication and coordination between various participants. Hence we view, *engineering as a collaborative process*. In other words, research in computer-aided engineering should strive to address various issues involved in the collaborative nature of engineering product development.

I will illustrate our view with an example. During a lecture on case-based design, I asked the students in my class to design an aircraft engine that uses an alternative source of energy (i.e., different than the sources used in current commercial aircraft). I also gave them a list of devices, such as windmills, electrical motors, etc. Many of them came up with designs that logically made sense (e.g., attach a windmill to the aircraft propeller), but will never work because the designs violated other concerns, such as the laws of thermodynamics (for an explanation, see Spalding and Cole's book on thermodynamics). Hence, engineering design involves an interplay between various disciplines. My experience in conducting protocol studies at several industrial sites further reinforced my view about the collaborative nature of engineering design.

I believe that computers could play a significant role in facilitating collaboration among engineers. One could envision a framework where several agents, where each agent could be viewed as a combination of a human and a computer, participate in a design process through a shared workspace. This could be viewed as a collaborative agent-based framework. In this paper I will discuss the various research issues that need be addressed by engineers, computer scientists, managers, psychologists to realize the collaborative agent-based framework.

2. Research Issues

A computer-aided cooperative product development environment would involve a close collaboration between computer scientists, engineers, cognitive scientists, and management

personnel. We believe that the following research areas will need to be addressed (see Figure 1).

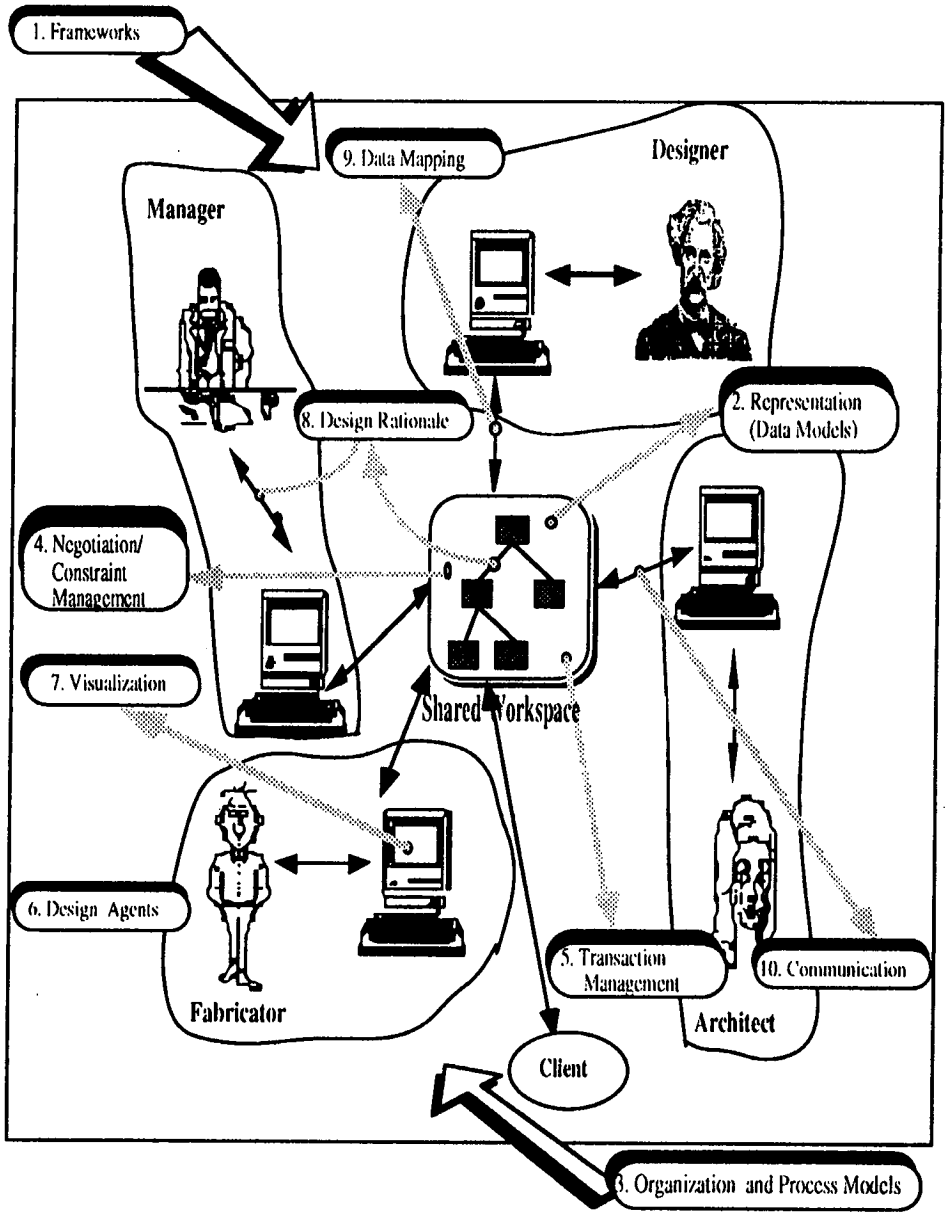

Figure 1: Research Issues for Computer-aided Collaborative Engineering

1. **Frameworks**, which deal with problem solving architectures.

2. **Representation Issues,** which deal with the development of product models needed for communicating information across disciplines.

3. **Organizational Issues**, which investigate strategies for organizing engineering activities for effective utilization of computer-aided tools.

4. **Negotiation/Constraint Management Techniques**, which deal with conflict detection and resolution between various agents.

5. **Transaction Management Issues**, which deal with the interaction issues between the agents and the central communication medium.

6. **Design Methods**, which deal with techniques utilized by individual agents.

7. Visualization Techniques, which include user interfaces and physical modeling techniques.

8. **Design Rationale Records**, which keep track of the justifications generated during design (or other engineering activities).

9. **Interfaces between Agents**, which support information transfer between various agents.

10. **Communication Protocols**, which facilitate the movement of objects between various applications.

3 Frameworks: Computer Architectures

Computer architectures deal with problem solving architectures. Most of the current work in this area utilizes a variation of the Blackboard architecture, developed in AI. Other types of architectures are reported in the proceedings of the workshops on distributed AI.

Over the past five years we have been working on a computer-based architecture called DICE (**D**istributed and **I**ntegrated environment for **C**omputer-aided **E**ngineering) which is aimed at addressing the coordination and communication problem in engineering. Essentially, DICE can be envisioned as a network of agents or Knowledge Modules (KMs) which communicate through a shared workspace; we call this shared workspace a Blackboard. In DICE, an agent is viewed as a combination of a user and a computer. Interface modules, provided at the agent level, will map the representations used by the agents to the shared workspace and vice versa. Agents (or KMs) in DICE are categorized into: Strategy, Specialist, Critic, and Quantitative KMs. The Strategy KMs help the Control Mechanism in the coordination and communication process. The Specialist KMs perform individual specialized tasks of the design and construction process. The Critic KMs check various aspects of the design process, while the Quantitative KMs are mostly algorithmic CAD tools. The shared workspace (Blackboard) is divided into three partitions: Solution, Coordination, and Negotiation partitions. A large part of the shared workspace is being implemented as an object-oriented database. The key components of this database are: Storage Manager, Object Manager, Transaction Manager, and Query Manager.

4 Product Modeling: SHARED Information Model

A basic issue in developing collaborative engineering systems is the representation of the product information which supports sharing. This product information should include not only the geometric data of the physical parts of the product and their relationships but also non--geometric information such as details on functionalities of the parts, constraints, and design intent. Requirements for such a design representation are:

* **Support for multiple levels of abstraction and different functional views.** This is needed to allow a top-down design process which involves the refinement of levels of functional abstraction into physical parts;

* **Support of multiple levels of geometric representation**. Geometric and topological information are an important part of design. However, at different stages in the design process, different levels of geometric representations might be required; and

* **Management of constraints.** Constraints between the different representations and abstractions during evolution of the design should be properly managed. Constraint management facilities help in maintaining the integrity and consistency of the database.

Besides, the representation should be reasonably general, hence allowing for the addition of new abstractions or physical components without requiring extensive changes. Furthermore, there is the requirement for mapping it onto a distributed database environment, which supports persistency and concurrent access in the shared workspace.

Our work aims at providing a framework for representing product information in a shared workspace which supports the requirements outlined above. The focus is on the development of general concepts such as geometric representations and abstractions of general properties such as concepts of "compositional" hierarchies of systems and components. These can be seen as primitives in our model, which is called SHARED.

SHARED is a tool kit/framework for developing environments which need to model, manipulate, and communicate design information between distributed cooperating applications, while supporting coordination between them. The SHARED tool kit is based on the SHARED information model which provides generic information structures needed for modeling product information through various design stages and from different functional views. SHARED also provides various services which are required by cooperating design applications. These include persistency, integrity checking, querying, transaction management, and notification services. Hence, the SHARED information model, together with the support services embedded in the SHARED framework presents a powerful system which should reduce the amount of work needed for developing environments for cooperative product development.

5 Organization and Process Models

Protocol studies on how engineers work together in groups (both large and small systems) could help us develop a language (with all the necessary vocabulary) that forms the basis for all collaborative engineering activities could be developed. Another important focus of these studies could be the investigation of strategies for organizing engineering activities for effective utilization of computer-aided tools.

We are conducting protocol studies on team design. These studies are being conducted over an extended period of time on designers working on real world problems, i.e., these designs will go through the engineering product life cycle. In another related work, we showed how an existing design can be reverse engineered through the use of collaborative specialists. A much larger effort at the MIT's Center for Coordination Science involves the development of a process handbook, which aims "to develop theoretical foundations, as well as a computer-based tool, that will aid in the design and analysis of both business processes and organizational structures."

6 Negotiation and Constraint Management

6.1 Negotiation

In large engineering projects, conflicts can occur either due to interface constraint violations or due to contradictory modifications of a single object. For example, a HVAC engineer may decide to place pipes at a certain location. However, an architect may also decide to place a beam at the same location. These conflicts can only be detected once the two designs have been generated and sufficient constraint propagation and/or modeling has been performed. Another type of constraint violation occurs when an engineer makes changes to a partial solution generated by another engineer. The two participants may or may not have similar roles in the system. For example, two architects may disagree on the location of the walkway, or the HVAC engineer might want to change the depth of a beam posted by the a structural engineer in order to put some pipes through it.

We have identified two types of techniques to address the negotiation problem: constraint relaxation and goal re-specification. The first attempt to negotiate involves traditional constraint relaxation techniques and implements a technique called compromise bargaining. Assuming that the conflict is due to constraint violations of certain design parameters, the system can act as a third party and offer compromise values to each party until an agreement is reached. In order to allow this scheme to function properly, each value posted on the Negotiation Blackboard has to be accompanied by a constraint. Each constraint must specify the range of possible values. These constraints can be either *soft* or *hard* constraints.

The second set of techniques involves the redefinition of design goals. The KMs (agents) are asked to negotiate on a more abstract plane. It is considered that the set of conflicting constraints are the concrete expression of an abstract hierarchy of goals. At the root of this hierarchy is the goal of designing and constructing the artifact for which the design team has been set up. Each participant develops his/her own hierarchy of personal goals. By helping the KMs find an agreement goal and developing a common set of more detailed goals, the system achieves integrative agreement.

6.2 Constraint Management

Constraints are continually being added, deleted and modified throughout the development of a new product. For example, the initial set of specifications may be augmented, changed and/or refined as the design progresses. The resulting constraint set may contain conflicting and/or unrealizable requirements. The management of these constraints throughout the evolving design is a non-trivial task. The constraints are often numerous, complex and contradictory (see Serrano's doctoral thesis at M.I.T. for more discussion about the role of constraints in engineering design). In complex engineering problem solving, where form, function and physics interact strongly, it is difficult to: 1) keep track of all relevant constraints and

parameters, and 2) understand the basic design relationships and tradeoffs. Effective tools for constraint management will facilitate good engineering; constraint management tools also aid in the negotiation process.

We are implementing a system -- called COPLAN -- which uses planning techniques to solve constraint satisfaction problems (CSPs). A planner is used as a top-level control process, guiding the search for a solution and producing an appropriate *solution plan* when the problem is solvable. The CSP is described by a *goal*. Usually the goal states which constraints should be satisfied but is more generally a list of assertions that should be true in the final world. The planner produces a non-linear *plan* at an abstract level where the different steps needed to achieve the goal are partially ordered. At the bottom level, numerical and symbolic methods are chosen in the order defined by the *plan*. The execution of a plan consists in executing the above procedure. This is very efficient in the case where one wants to vary a parameter over a certain range and to study its influence on other values for a given CSP.

7 Transaction Management

Collaborative engineering environments require a flexible framework for concurrency management of highly interleaved and interactive transactions. Traditional database management systems tend to control concurrency and maintain database consistency using the notions of *atomicity* and *serializability* of transactions. Serializability is founded on the assumption that individual concurrent transactions run oblivious to each other and do not interact in the middle of their execution. By enforcing serializability, the DBMS ensures a consistent database state irrespective of the nature of the transactions.

Though serializability is adequate (and even desirable) for financial applications, since CAD/CAE transactions are often of long duration, it is highly inappropriate for such work because it inhibits information sharing and may result in reduced concurrency and intolerably long waits. Collaborative engineering, is based on the notion that *units of work must interact,* so that the results are usable together. Besides, in engineering design the notion of data "correctness" is more relaxed and is application or context specific, i.e., it depends on the *nature of the design operations* . Serializability is thus too limiting a correctness criteria for such work, and hence more flexible concurrency control protocols need to be used. Special considerations are therefore necessary for the design of a more flexible and efficient transaction management system which allows a group of cooperating transactions to arrive at a complex design without being forced to wait over a long duration, and enables collaboration among design groups.

We have developed and prototyped a transaction framework for facilitating and coordinating collaborative engineering activities using object-oriented databases. The key features of our transaction management model include: transaction nesting and grouping with partitioned dataspaces for encapsulation of non-serialized data sharing, exploitation of application semantics to ensure data consistency, communication facilities, and version management for parallelism and documentation of design evolution.

8 Agents

Agents perform specific tasks. For example, a conceptual design agent generates a preliminary design. Most of the current work is focused on the development of generic design and planning shells, that form part of the design and planning agents. These shells tend to be domain independent and incorporate certain problem solving methodologies, e.g., case-based reasoning, hierarchical refinement, constraint propagation, qualitative reasoning, etc. Key issues in the development of design shells are:

1. **Problem Solving Support.** What kinds of problem solving techniques (e.g., hierarchical refinement, qualitative reasoning, first principles reasoning, etc.) need to be incorporated in the shell?

2. **Representation.** How does one represent structure, function, behavior, geometry, design rationale, etc.?

3. **Index and Retrieval..** What schemes are required for using function, structure, sub-graph matching, behavior, etc., for indexing and retrieving past designs?

4. **Data Quality.** How is the fuzzy engineering data represented and processed?

We are developing a shell -- called CONGEN -- for supporting various kinds of design activities. CONGEN supports the hierarchical refinement and constraint propagation problem solving techniques. A design rationale and intent model (DRIM), a comprehensive knowledge-base, and a case-based reasoner are being incorporated. Other agents that we have developed over the years are: 1) GHOST and BUILDER for scheduling construction projects; and 2) DATON for detailed design of steel structures.

9 Visualization of Engineering Information

Engineers make extensive use of diagrams (images) to convey their ideas. They also like to see scientific information (or data) to be conveyed by visual diagrams (images). Hence computer-based systems for engineering should have the ability to: 1) recognize and understand diagrams, and 2) generate diagrams. The study and development of the methodologies required to provide above capabilities in a computer program falls under the realm of *Visual Languages*. Visual languages can be classified into: 1) Visual Information Processing Languages (VIPL), and 2) Visual Programming Languages (VPL). In VIPL, the objects that are displayed on the screen (by the engineer) have inherent visual meaning, i.e., the object has some semantic meaning associated with it. The task here is to map these objects into their semantic content. An example of VIPL is spatial reasoning about engineering objects. On the other hand in VPL, the visual diagrams are generated on the screen from scientific (or otherwise) data and it is left to the engineer to extract the semantic meaning of these diagrams, e.g., current work on solid modeling. It is also important to realize that these visual languages should be portable. Hence, they should be developed in an environment that is portable across a variety of hardware, such as the X Window system, which is rapidly gaining acceptance as an industry standard.

Our work is focused on the following areas:

1. **Symbol to Structure Mapping.** Various design alternatives are mapped from a symbol space to a geometry space.

2. **Interpretation of Engineering Drawings.** The semantic content of engineering drawings is extracted in the form of design objects and relationships between these design objects, in an appropriate agent's space (or view).

3. **Geometric Modeling.** Geometrical information forms an important part of the information about a product. Solid modeling captures the "complete" representation of 3D solid objects. However, in design, objects usually evolve from sketchy form of lower dimensionality during the conceptual stage to complete 3D models at the detail design stage. It is important that this evolution of design objects be captured, not only for preserving possible geometric constraints of these early design decisions on later designs, but also to allow communication of concepts during the early design stages. The later is especially important for cooperative product development. The non-manifold geometric representation scheme provides a uniform paradigm for representation and manipulation of mixed-dimensional models. This makes it suitable for capturing the design evolution. We have implemented an object-oriented non-manifold geometric modeler -- GNOMES -- for capturing geometrical information at various stages of design. GNOMES is utilized by our SHARED information model.

10 Design Rationale

One problem with current design practice is the lack of information about engineering decisions. Hence, any future CAE tool should incorporate techniques to encode design rationale about both the overall process and the individual choice points.

The design rationale and intent model (SHARED-DRIMS), which is being developed as a part of our DICE initiative, provides a methodology for encoding design rationale and intent both at the individual design agent and the shared workspace levels. In particular, SHARED-DRIMS helps the engineer to document the following: intent evolution; artifact evolution; relationships between intents and between intents and artifacts; and negotiated rationale from multiple agents. The key advantage of our model is that it provides a complete picture of the design rationale to agents participating in the collaborative engineering activity.

11 Data Mapping and Query Processing

In a collaborative environment, agents may work with different representations. Hence, there is a need for translation of information (both syntactic and semantic mappings) between the shared workspace and local applications. Further, facilities are needed for status checking and monitoring; for communication between the shared workspace and users; and for coordination. We are working on tools for integrating heterogeneous applications. Our tools perform both syntactic and semantic translations between the shared workspace and local applications. In addition, facilitates for distributed query processing are being incorporated.

12 Communication Frameworks

Communication is an important prerequisite for the success of any cooperative work. It is required for the coordination, negotiation and cooperative development of engineering ideas. There are several ways to address this problem.

1. **Electronic Message System**. At the simplest level electronic mail could be used for communication.

2. **Electronic Video/Audio Conferencing System**. This would allow conferencing between users who are geographically separated. Users should be provided with a *shared window* which displays the information of a *shared workspace* on their individual displays. The *shared window* could allow users to edit and process the information of the shared workspace dynamically. An important use of the conference system will be in negotiation.

Research into how people communicate should help us address the above issues,. We have developed and demonstrated a framework for cooperative user interfaces.

13 Summary

In this paper, I have espoused a view that *engineering is a collaborative process*. The future of computers in engineering will depend on how effectively we can build tools for collaborative activities. I believe that the development of a collaborative agent-based architecture should be undertaken. This would involve a close cooperation between computer scientists (databases, AI, communications, visual information processing), psychologists (negotiation, conflict resolution), managers (organization), and engineers (domain).

Six years ago, we have embarked on the development of a set of tools for supporting collaborative engineering activities. The main contributions of this venture -- the M.I.T. DICE project -- are:

1. An object-oriented blackboard architecture (DICE) that supports persistent objects
(Publication: **Sriram, D., Logcher, R., Groleau. N., and Cherneff, J.** , DICE: *An Object Oriented Programming Environment for Cooperative Engineering Design*, Technical Report IESL-89-03, IESL, Dept. of Civil Engineering, M. I.T., 1989 [Also appeared in AI in Engineering Design, Tong, C. and Sriram, D., (editors), Academic Press, 1992]).

2. An object-oriented knowledge-based building tool (COSMOS), which integrates rule-based and object-oriented programming paradigms in a C++ environment
(Publication: **Sriram, D., et al.**, *An Object-Oriented Knowledge Based Building Tool for Engineering Applications*, Technical Report, IESL, 1991).

3. A domain independent C++-based shell for conceptual design (CONGEN)
(Publication: **Sriram, D., et al.**, *Engineering Cycle: A Case Study and Implications for CAE*, In Knowledge Aided Design, Green, M (editor), Academic Press, 1992.

4. The concept of shared workspaces for storing product information that is shared by various engineering disciplines

(Publication: **A. Wong and D. Sriram,** *SHARED: An Information Model for Collaborative Product Development,* Research in Engineering Design, 1993).

5. A transaction management framework that supports long duration interleaved CAD transactions
(Publication: **Ahmed, S., Sriram, D., and Logcher, R.,** *Transaction Management Issues in Collaborative Engineering,* ASCE Journal of Computing in Civil Engineering, January 1992 and Engineering with Computers, Fall 1992).

6. User interfaces for collaborative work
(Publication: **Wong, A., Sriram, D., and Logcher, R.,** *User Interfaces for Cooperative Product Development,* Proceedings of the Second National Symposium on Concurrent Engineering, West Virginia University, Feb., 1990).

7. Prototype implementation (MagpieBridge) in a commercial object-oriented database management system (GEMSTONE)
(Publication: **Sriram, D., Logcher, R., Wong, A., and Ahmed, S.,** *Computer-Aided Cooperative Product Development: A Case Study,* International Journal of Systems Automation: Research and Applications (SARA) 1, 91-114, 1991).

8. A constraint management system that utilizes AI planning techniques (COPLAN)
(Publication: **Fromont, B. and Sriram, D.,** *Constraint Satisfaction as a Planning Process,* AI in Engineering Design, Held at Carnegie Mellon University, July, Kluwer Publishers, July 1992.)

9. DATON, BUILDER, and GHOST, which are design/planning agents
(Publications: **Agbayani, N., Sriram, D., and Jayachandran, P.,** *An Object Oriented Framework for Steel Design: Implementation Issues,* Computing Systems in Engineering, 1992 [DATON].
Cherneff, J., Logcher, R., and Sriram, D., *Integrating CAD with Construction Schedule Generation,* ASCE Journal of Computing in Civil Engineering, January 1991 [BUILDER].
Navinchandra, D., Sriram, D., and Logcher, R., *GHOST: A Project Network Generator,* ASCE Journal of Computing, July 1988 [GHOST].)

14 Credits

The DICE project is headed by D. Sriram and Robert Logcher. Nicolas Groleau worked on the initial implementations of DICE. Albert Wong worked on the user interface module and the shared workspace concept. Shamim Ahmed was responsible for the transaction management framework. CONGEN was a joint effort between S. Gorti, A. Nishino, Kevin Cheong, and Parin Gandhi. Bruno Fromont and Fred Garcia are working on the constraint management system. Nestor Agbayani's SM thesis dealt with DATON. Jonathan Cherneff addressed the drawing interpretation problem in his doctoral dissertation. He was the prime architect of the BUILDER system. Also, discussions with him helped us to reformulate and rationalize our thoughts in a more coherent manner. Navinchandra was responsible for the GHOST system and the initial work on a case-based reasoner (CYCLOPS). Feniosky Pena is currently working on SHARED-DRIMS, the design rationale intent model. S. Gorti is responsible for the symbol to structure mapping of design alternatives. Query and storage manage facilities of DICE are being implemented by Murali Vemulpati. Domain aspects (in the AEC industry) of DICE were addressed by Miriam Gross and Erik Swenson.

Funding for the DICE project comes from the IESL affiliates program and a NSF PYI Award No. DDM-8957464, with matching grants from NTT Data, Japan and Digital Equipment Corporation, USA. Partial support for Albert Wong was provided by Gleddon Postgraduate Studentship from the University of Western Australia. Kevin Cheong was supported partially by a X-Window consortium grant. Bruno Fromont's fellowship came from Aerospatiale, France.