Engineering with Computers (1993) 9: 125–132

Engineering with Computers

Supporting Collaborative Engineering Design

J. Favela, A. Wong and A. Chakravarthy†

Room 1–250, Intelligent Engineering Systems Laboratory, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge MA 02139, Massachusetts, USA

Abstract. *The complexity of current engineering design demands the collaboration of specialists. Collaboration involves both communication between and coordination among members of a design team. Currently, design collaboration is carried out through the use of schedules, specifications and drawings which only capture the end results of the design process. They fail to record important design information such as the reasoning behind design decisions. In this paper, we describe a tool, CADS, that supports collaboration in engineering design. CADS is based on an extension to the* Axiomatic Design *methodology which not only captures the design elements but also the rationale used to conceive them. CADS serves as a shared database of design information with facilities that support communication and coordination in engineering design.*

Keywords. Axiomatic design methodology; Collaborative engineering; Design rationale

1. Introduction

Most large engineering systems are designed collaboratively through the interaction of specialists. Take for example, the Architecture–Engineering–Construction (AEC) domain. In the design of an office building, an architect interacts with the users to find out what their requirements are. Then, the architect proceeds to make an initial layout of the building. This plan is then examined by a structural engineer who checks whether the design is acceptable structurally. If some part of the design is unacceptable, the structural engineer refers the plan back to the architect with recommendations. Collaboration thus involves both communication between and coordination among the specialists. In this paper, we describe a system named Collaborative Axiomatic Design Support tool (CADS) which provides a medium for communication and coordination between designers.

In current design practice, the main form of communication is through specifications, plans and drawings. These only capture the end results of the design process. Important design knowledge such as assumptions and arguments for and against alternatives, usually classified under *design rationale*, are not recorded. This lack of documentation often leads to errors in design when downstream decisions are made based on conflicting assumptions. The lack of quick access to information also results in coordination problems, especially when the design process is not explicitly specified. Conflicts in designs by different members of a team often happen, resulting in inefficiencies in the design process and loss of productivity [1].

An explicit design methodology helps coordinate design tasks by providing an explicit protocol for collaborative design. One such methodology is that of *Axiomatic Design* which is used as a model for the design process in CADS [2]. As described in Section 2, Axiomatic Design induces a hierarchical design style. The designer is also required to specify clearly the role of each level of the hierarchy. By following this methodology, collaborators generate a hierarchical product decomposition and associated functional requirements. However, design rationale is not recorded even when this methodology is used. In Section 3, we propose a design language, *ADL*, that is capable of recording such information as it is generated during the design process [3]. This language extends the Axiomatic Design methodology with an ontology similar to that found in DRL [4].

In Section 4, we describe the tool, CADS, which uses the ADL language to support collaborative design. We show an example of how the tool is used in an axiomatic design process for recording alternatives that were considered and the rationale behind

† Currently at the Media Laboratory, MIT.

Correspondence and offprint requests to: J. Favela, Room 1–250, Intelligent Engineering Systems Laboratory, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA

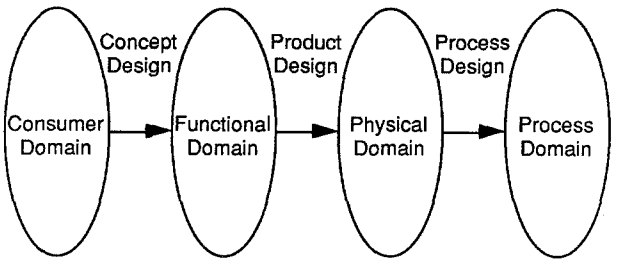

Fig. 1. The domains of the design process.

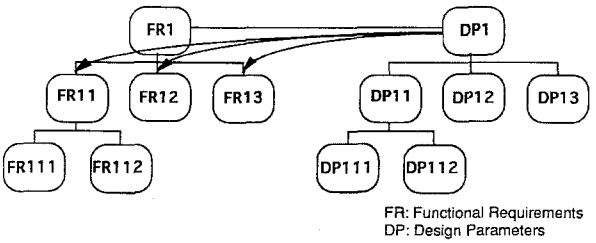

Fig. 2. Hierarchical decomposition of a product design.

them. This is followed by a description of the tool's features in the context of support for communication and coordination in Section 5. Finally, Section 6 contains a summary and discusses areas where research needs to be done.

Axiomatic Approach to Design

The axiomatic approach to design was proposed in the late 1970s by Nam Suh at MIT with the purpose of developing a scientific approach to the problem of design, which had been then largely viewed as a non-rational endeavor [2]. Since it was first introduced, the axiomatic approach to design has been successfully applied in the design of software [5], manufacturing processes [6], construction facilities [7], organizations [8] and other areas.

The axiomatic approach to design has its foundation on two axioms that were postulated by generalizing characteristics common to good designs. The first axiom, known as the *independence axiom*, states that the *functional requirements* that dictate the need for a design should be kept independent of each other. The second axiom, called the *information axiom*, characterizes an optimal design as one that minimizes the information content of a design. In this paper, we will only be concerned with the implications of the first axiom since it is more directly related to the design process.

The design methodology that follows from the independence axiom is based on step-by-step mappings between four independent domains that characterize the elements of a design (see Fig. 1). These domains are:

the *consumer domain*, where user needs are identified; the *functional domain* where these needs are stated in the form of required functionalities of a product; the *physical domain* where design parameters that satisfy the functional requirements are defined, and the *process domain* where process variables that define how the product will be implemented reside. The *concept design* is established by the mapping between the consumer and the functional domains, the mapping between the functional and the physical domains defines the *product design*, and the mapping between the physical and the process domain corresponds to the *process design*.

Figure 2 illustrates the methodology for a product design. The design process starts at a conceptual level with the definition of a few independent functional requirements and their corresponding design parameters. The design continues with the decomposition of functional requirements by answering the question: What are the requirements of each of the top-level design parameters? Different design parameters might generate a number of different sub-requirements. However, at each level of the decomposition, the designers should guarantee that the functional requirements are independent of each other, that is, the solution of each functional requirement should not affect other requirements at the same level of the hierarchy. Once the new requirements are identified, the mapping across domains continues by defining the design parameters that satisfy the more specific functional requirements.

The implications of the design axioms in the product development process have been formalized in the form of theorems and corollaries, and can be referred to in reference [2]. As will be seen in the example below, these implications of the axioms can be used to argue and decide between design alternatives.

3. Axiomatic Design Language (ADL)

This section describes the Axiomatic Design Language (ADL) which can be used to capture information generated during an axiomatic design process, specifically the product design stage. It provides information structures to model the hierarchical decomposition of the design into *functional requirements* and *design parameters*, *design alternatives/versions*, and the *cross-linking* between the two types of hierarchies. It also includes elements to allow the *capture of design*

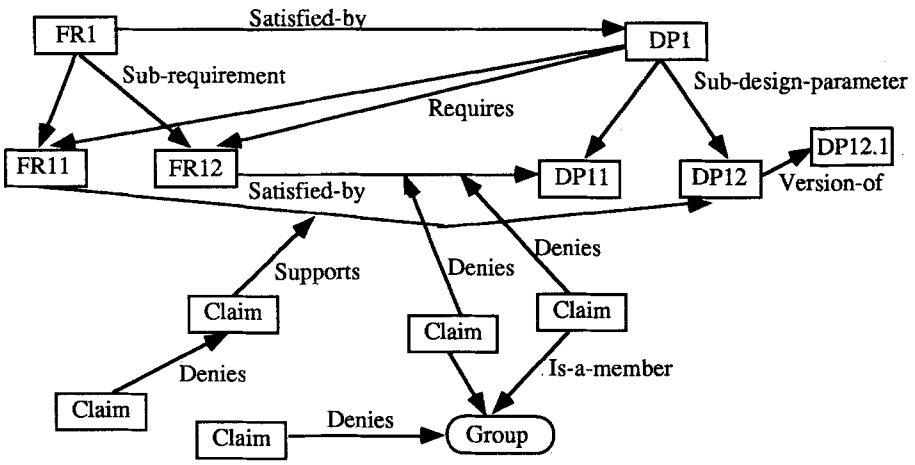

Fig. 3. ADL elements and relationships.

rationale and associates it with the elements of axiomatic design.

3.1. Elements of the Axiomatic Design Language

The constructs defined in ADL that are necessary to capture the axiomatic design process and the design rationale used are (see Fig. 3):

- *Functional Requirement.* A *Functional Requirement* is associated wih one or more design alternatives (also known as Design Parameters) which satisfy the functional requirement by the *Satisfied-by* relationship. Each functional requirement can be decomposed into a number of sub-functional requirements (*sub-requirements*) based on the design chosen. The cross-linking between a design and sub-requirements is depicted by the *Requires* relationships.
- *Design Parameter.* This is used to represent an attribute of a design. It can be decomposed into a number of sub-design parameters related by the *Sub-design-parameter* relationships.
- *Relationships.* The different types of relationships are: *Satisfied-by*, *Requires*, *Denies*, *Supports*, *Sub-requirement*, *Sub-design-parameter*, *Is-a-member* and *Version-of*. The *Version-of* relationship is used to associate different alternatives of Design Parameters which satisfy a particular functional requirement. Each of the different versions of Design Parameters results in a new Design parameter sub-tree and a new Functional Requirement sub-tree. The *Is-a-member* relationship is used to specify a membership in a *Group* element (see below).
- Claim. This is used to represent an argument made in the design process. Claims can be attached to relationships and other nodes of ADL through the *supports* or *denies* relationships.
- Group. This is used to group together a set of elements, usually so that some relation can be associated with the group.

Each of the above constructs also includes the following attributes: description, author and creation time.

4. Collaborative Axiomatic Design Support Tool

CADS has been designed as a tool to support collaborative engineering product design, specifically one following the Axiomatic Design Methodology. It contains a high-level graphical interface for using the Axiomatic Design Language which provides an explicit protocol for negotiation and decision making. The information that is represented in ADL acts as a shared database of information which is easily accessible to members of a design team. The system only guides the design process and imposes no restrictions on decision making. The current design is an asynchronous system based on shared files. Eventually, we are looking at implementing the tool on top of a distributed object oriented database, allowing real-time collaboration over a network [9].

4.1. An Example: Design of a Building Partition

This section describes a scenario that shows how the CADS tool can be used to support collaborative engineering design in the Architecture–Engineering–Construction (AEC) domain. It deals with the conceptual design of a partition to divide a space in a building that is located in an earthquake-sensitive area. The participants involved are an architect and a structural engineer.

The architect initiates a new design by selecting the

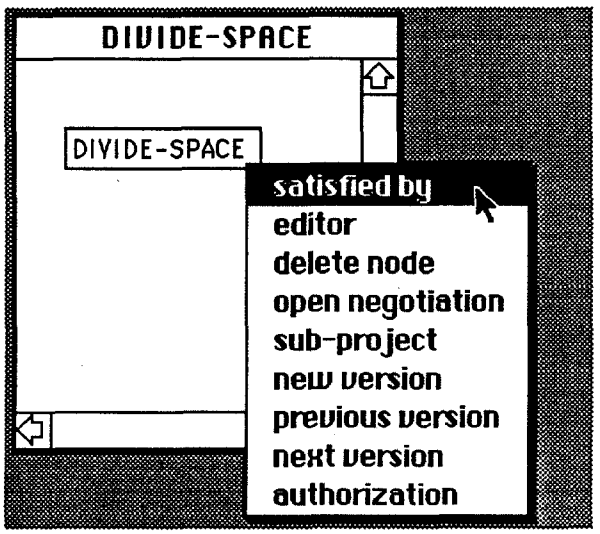

Fig. 4. A design parameter that satisfies the divide space functional requirement is created.

command *New* under the *File* menu, and is then presented with an empty window that has two subviews: one corresponds to the functional domain and the other to the physical domain. The project starts with the definition of the primary functional requirement which in this case is to *divide space*. This functional requirement is represented as a box that will eventually be the root of the functional requirement hierarchy. A pop-up menu with potential operations can now be accessed by pressing the mouse button while the cursor is on top of the node that has been just created. This is illustrated in Fig. 4.

At this point the user selects the *Satisfied by* command, which will display an electronic form that queries the user for information about the design parameter that corresponds to that requirement (Fig. 5). A *non-load bearing wall* is then posed as the design

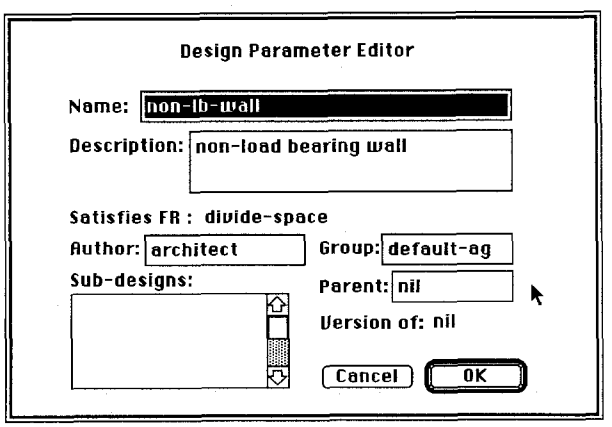

Fig. 5. The design parameter editor.

parameter that satisfies the need to *divide space*. After the transaction is committed by clicking on the *OK* button, a new node that represents the new design parameter is displayed in the physical domain subview.

The functional domain is now decomposed by defining the functional requirements that satisfy the top-level design parameter. This is done by invoking the *requires* command from the pop-up menu that appears when the architect selects the node that represents the design parameter. In this case, the user creates two sub-requirements which are that the wall *must be supported* and *must cover the dividing space*. Based on these sub-requirements, the non-load bearing wall can be decomposed into two sub-designs (i.e., the wall should be *attached rigidly to the columns* and *its size should be equal to the separation between columns and the height of the floor*).

However, when the structural engineer opens the project database, he perceives a problem with the design from the perspective of earthquake safety. This is because a rigid attachment of a wall to the structural frame *would interfere with the free movement and performance of the structural frame*. Besides, the *wall will likely crack* even if the structural frame is not damaged. Hence, the design is unsuitable and the engineer adds a claim to the *attach to columns* design parameter. To do this, he opens the negotiation window from that relationship by selecting the *open negotiation* command. The negotiation window will display a node labeled *FR12–DP12* that represents the relationship between the source and destination nodes. In this case, it represents the fact that the *must cover space* functional requirement (source node) is *satisfied by* the design parameter *attach to columns* (destination node). A claim is added by clicking on the relationship to be argued upon, represented by the node labeled *FR12–DP12*, and selecting the appropriate option (i.e., *supports* or *denies*). In our example, as illustrated in Fig. 6, the structural engineer chooses to *deny* the validity of the relation. An argumentation box that points to the issue being denied is provided where the user can state his argument. Claims can be emphasized by adding images or sound annotations, as in this case, where the engineer dramatizes his claim by including an image that shows the potentially disastrous consequences of the current design.

From the more formal perspective of axiomatic design, the problem with the current design is that it is coupled to the structural system. Following the recommendation of Corollary 7 of axiomatic design, which suggests pursuing an uncoupled design rather than a coupled one [2], the engineer decides to create a new version of the design by adding a new functional requirement, namely *allow the free movement of the*

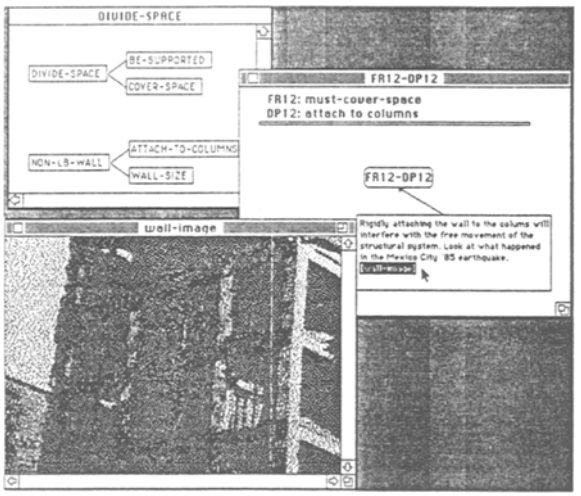

Fig. 6. A claim is added to the satisfied-by relationship between the must-cover-space functional requirement and the attach-to-columns design parameter.

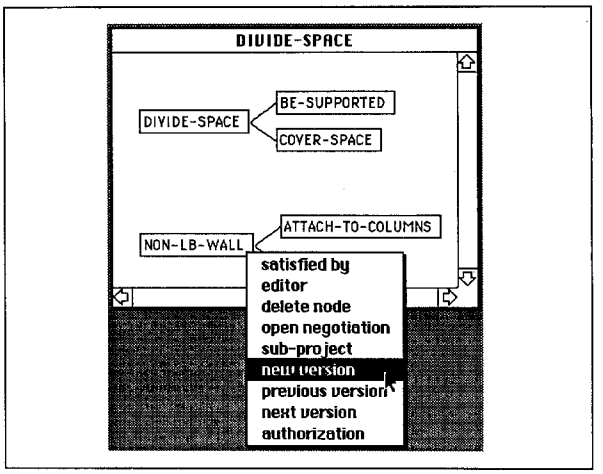

Fig. 7. A new version of the design is created from the non-load bearing wall design parameter.

structural system. Before he does that, and to document his rationale, he adds a claim to the *requires* relationship between the *non-load-bearing wall* design parameter and its associated sub-requirements. The axiomatic design theorems and corollaries are provided in the form of a library of claims in CADS. An advantage of using these claims is that all the collaborators can understand the consequences of them (the desire for an alternative design, in this case) without further explanations (see Section 5).

In order to create a new version of the design, the structural engineer refers to the last level of the hierarchy that will not be modified in the new design, in this case the *non-load-bearing wall* design parameter, and from it the *new version* command is called as shown in Fig. 7. When the command is executed, the nodes below this layer will be removed from both hierarchies. Since the new design will only add a new sub-requirement to the original version, the user can return to the previous design, using the *previous version* command from the top-level design parameter, select the original sub-requirements, copy them into the clipboard, return to the new design and paste them as sub-requirements of *divide-space.*

With the new version of sub-requirements proposed by the engineer, the architect can now access the project, get a summary of the changes made since he last worked on the project and review the rationale behind the decisions that were made. As part of the notification mechanism provided in CADS, for every relationship that is changed, its destination node is highlighted.

The architect then proposes a new design which satisfies the new functional requirements: a wall *attached to supporting beams* with *spaces between the wall panels* to allow for wall-panel movements. The design continues with the definition of new sub-requirements for the design (i.e., *the supporting beam, the connections, and the wall panels must have enough strength*). The structural engineer is notified of these requirements and *designs the components* accordingly (see Fig. 8). Similarly, there is a sub-requirement that the *noise transmission* through the space between the wall panels must also be *minimized.* For this requirement, *flexible joints filled with elastic material* are proposed as the design solution. The functional and design parameter hierarchies of the final design are shown in Fig. 8.

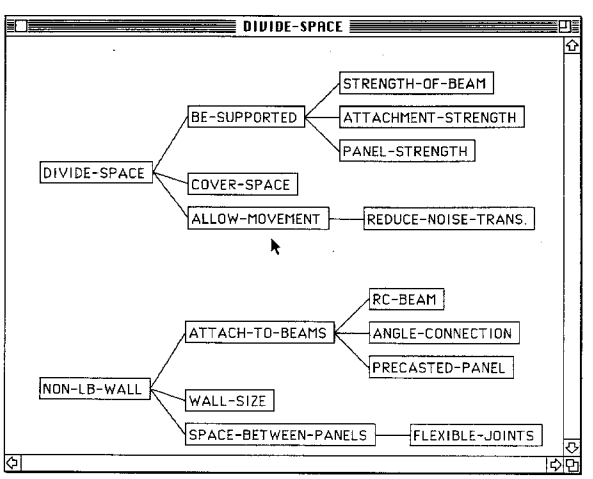

Fig. 8. The final design.

5. Support for Collaboration in CADS

Support in CADS for Collaborative Engineering Design is discussed in more detail in this section. These supporting facilities are described under the two general categories of capabilities of CSCW tools: Communication and Coordination [10].

5.1. Support for Communication

CADS uses ADL to define a common language that allows the communication of design information and arguments generated during the axiomatic design process. The ADL language is expressive enough to represent the important elements and relationships between them.

The Axiomatic Design Methodology also provides various theorems and corollaries as criteria for creating better designs [2]. CADS provides these as a library of claims which can be used during the argumentation process. The use of these claims not only decreases the input workload of users, but leads to better communication for users who are fluent with the axiomatic design methodology. Furthermore, they should result in a better design.

Besides, nodes can also be linked with design objects in databases or with multimedia data such as images to emphasize a claim during argumentation. Currently, the 'association' is provided through an image filename attribute and a design object part identifier/CAD file name attribute in a node. CADS's browsing facility provides faster information access compared with traditional methods such as drawings.

5.2. Support for Coordination

Coordination is concerned with the scheduling and management of individual work of the members of the design team. CADS, by embedding the Axiomatic Design Methodology, defines a formal and explicit protocol for the design process. The zig-zagging between the functional and physical domains in the decomposition of the hierarchies suggests a 'plan' of what to do next among members. For designers who are familiar with Axiomatic design methodology, this process is natural and has been found to be a successful way of making sure all requirements are identified and satisfied [2]. However, CADS only guides the design process using the axiomatic methodology and does not force the user into a standard process when making arguments. This is important especially when 'brainstorming' or arguing about subsequent steps in the design.

The axiomatic design methodology classifies designs into three types based on the relationships between the *Functional requirements* and *Design parameters* at each level of the hierarchies. These relationships across hierarchies can be displayed using a menu command. The relationships can also be depicted in the form of a design matrix, as shown in Fig. 9. Different approaches to task coordination are foreseen in each case:

- In a *functionally uncoupled design*, each functional requirement is linked to only one design parameter. In this case, each of the sub-designs can basically be decomposed (designed) independently by different members of a design team.
- In a *functionally decoupled design*, the functional requirements and design parameters are related such that a lower triangular matrix is obtained. Here, the design has to proceed from the least coupled sub-designs to the more coupled sub-designs.
- In a *functionally coupled design*, the functional requirements and design parameters are related in such a way that a matrix with elements on both sides of the diagonal is obtained. In this case, the design has to be done cooperatively and synchronously.

For good designs, the Axiomatic Design Methodology recommends that designers strive for functionally uncoupled designs, and the functionally coupled design is deemed to be unacceptable [2].

CADS provides a scheduling mechanism for design sub-tasks based on the principles described above. A *functional requirement* node and its *design parameter* node in the *uncoupled* case (or design parameters in the *decoupled* case) can be assigned to different team members or sub-groups as new sub-projects to be carried out independently. A menu option can be invoked whereby an electronic message is sent to the responsible team member(s) and the required authorization (see below) for the nodes is assigned to the person(s).

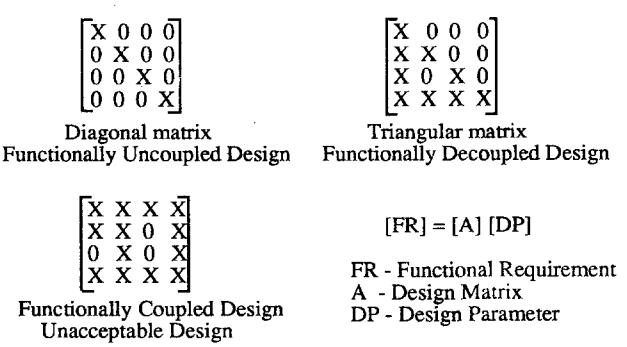

Fig. 9. Varieties of the Design Matrix, *A*.

Another form of coordination involves the use of notification mechanisms which keep other members of a team up to date with changes. Nodes and relationships are highlighted if there had been changes since the last session of the user. A textual chronological summary can also be obtained. In the future version where a shared database is to be used, the shared database will be updated and information propagated to multiple CADS clients whenever a transaction is committed. We are currently also looking into advanced transaction management strategies which are more suited and less restrictive for concurrent groupwork. These include the ideas of *group* and *shared* transactions [11].

Support for Authorization/Security is also an important issue in collaborative design. In CADS, an access list is kept in each element of ADL. This list contains elements which are triplets: ⟨*user identified, access right, access right*⟩ where access right can either be read, write, or none. The first access right denotes whether a user can add claims to the ADL element. The second access right refers to the access of design operations on the element itself. Users can also group a team of designers and their access rights and associate them with the access list. For example, when a sub-project is assigned as described above, read and write authorization might be given to a member only and none for others. In this case, the top-level nodes of the sub-project will be shown as insensitive nodes (on which menu operations cannot be invoked) to other users while the sub-tree is not shown. When the design is finished, the user can then set read authorization for the rest of the group.

6. Conclusions

This paper addresses the problem of collaboration in engineering design environments. In particular, we suggest solutions to the issues of coordination and communication through the use of the Axiomatic Design Language. The methodology embedded in the language is currently being used in diverse areas of design and provides an explicit protocol for collaboration. The axiomatic design methodology, when extended through ADL to record design intent, becomes a powerful vehicle for communication among members of a design team.

We have also described a tool to support collaboration in engineering design which is based on ADL. CADS provides various facilities for communication and coordination which include tools for editing and browsing design hierarchies, a library of claims, management of alternatives, support for multi-media annotations and notification mechanisms, among others. The fact that this tool implements an already accepted work process should facilitate its acceptance by current users of axiomatic design, and help the methodology gain support among those who consider the maintenance of the hierarchies and alternatives too cumbersome for the design of large systems.

Various systems for general decision making have been described elsewhere [12, 13]. Our approach, however, focuses on engineering design activities and therefore provides better support for these activities. The integration in CADS of an existing design methodology and its decomposition process, which is an important component of most design activities, is not directly supported in the general decision-making tools.

However, several issues still need to be addressed before the CADS tool can be used in large, real-world scenarios. We need to integrate this tool with other design databases, such as those that contain product models of the design parameters [14]. One crucial question that remains to be solved is whether the extra work required to maintain the documentation of the design rationale can be sustained in large projects. We are also working on ways to support the retrieval of previous designs that are similar to the one being considered [15]. This will encourage the documentation of design rationale and will help to preserve organizational memory.

Acknowledgements

We would like to thank Jerry Connor and John Williams for introducing us to the axiomatic design methodology, Sriram Duvvuru for providing insights into the problem of engineering collaboration, and Kenji Imai and Hiroyuki Takagi for their helpful comments regarding practical issues facing the AEC industry. Lastly, we would like to thank the Intelligent Engineering Systems Laboratory for providing us with an environment that encourages collaboration.

References

1. Howard, H.C.; Levitt, R.E.; Paulson, B.C.; Pohl, J.G.; Tatum, C.B. (1989) Computer integration: reducing fragmentation in AEC industry, Journal of Computing in Civil Engineering, 3, 18–32
2. Suh, N.P. (1990) The Principles of Design, Oxford University Press, New York
3. Wong, A.; Chakravarthy, A.S.; Favela, J. (1992) Documentation and communication in collaborative engineering design, presented at the AAAI Workshop on Design Rationale Use and Capture, San Jose, California, July 12–16

4. Lee, J. (1989) Decision representation language (DRL) and its support environment, AI Working Paper No. 325, AI Lab, MIT, Cambridge, Massachusetts
5. Kim, S.J.; Suh, N.P., Kim, S.G. (1991) Design of software systems based on axiomatic design, Robotics and Computer Integrated Manufacturing, 3, 243–255
6. Black, J.T. (1991) The design of manufacturing systems: axiomatic approach, MSF/MIT Special Workshop on Design, Cambridge, Massachusetts, January 22–23
7. Albano, L. (1992) An axiomatic approach to performance-based design, PhD Thesis, Department of Civil Engineering, MIT, Cambridge, Massachusetts
8. Carlson, B. (1991) Organizational design, science and intuition, MSF/MIT Special Workshop on Design, Cambridge, Massachusetts, January 22–23
9. Ahmed, S.; Wong, A.; Sriram, D.; Logcher, R. (1992) Object-oriented database management systems for engineering: a comparison, Journal of Object Oriented Programming, June, 27–44
10. Wexelblat, A. (1991) Building collaborative interfaces, Tutorial notes, ACM Conference on Computer Human Interactions, New Orleans, April
11. Ahmed, S. (1991) Transaction and version management in object-oriented database management systems for collaborative engineering applications, Master's Thesis, Department of Civil Engineering, MIT, Cambridge, Massachusetts
12. Conklin, J.; Begeman, M.L. (1988) gIBIS: a hypertext tool for exploratory policy discussion, ACM Transactions on Office Information Systems, 6, 303–331
13. Lee, J. (1990) SIBYL: a tool for managing group decision rationale, ACM Conference of Computer Supported Cooperative Work, Los Angeles, October 7–10, 79–92
14. Sriram, D.; Wong, A.; Logcher, R. (1991) Shared workspaces in computer-aided collaborative product development, First International Symposium on Building Systems Automation-Integration, University of Wisconsin, Madison-Wisconsin, June 2–8
15. Favela, J. (1992) Guided exploration of design information spaces, Thesis proposal submitted to the Department of Civil Engineering, MIT, Cambridge, Massachusetts

Announcement

Twelfth U.S. National Congress of Applied Mechanics
June 26–July 1, 1994
University of Washington, Seattle, WA 98195

Sessions are being planned for general lectures, symposia, and contributed papers covering all aspects of research which are of general interest to the Applied Mechanics Community. Contributed research papers will be selected from 300–500 word summaries which must be submitted for consideration by **October, 1993**. Inquiries regarding the Congress should be addressed to:

Professor Albert S. Kobayashi
Department of Mechanical Engineering
FU-10
University of Washington
Seattle, WA 98195, USA
Tel: (206) 543–5488
Fax: (206) 685–8047
e-mail: kobayashi@milton.u.washington.edu

The United States National Congress of Applied Mechanics is organized by the United States National Committee on Theoretical and Applied Mechanics under the general sponsorship of the National Academy of Sciences and the National Academy of Engineering. Cooperation Societies are: American Institute of Aeronautics and Astronautics, American Institute of Chemical Engineers, American Mathematical Society, American Physical Society of Civil Engineers, American Society of Mechanical Engineers, American Society for Testing and Materials, Society of Engineering Science, Society for Experimental Mechanics, Society for Industrial and Applied Mathematics, Society for Naval Architects and Marine Engineers, Society of Rheology.