CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

Knowledge-Based Support for Concurrent, Multidisciplinary Design

Raymond E. Levitt, Clive L. Dym, and Yan Jin

CIFE Working Paper
Number 10

January, 1991

Stanford University

If you would like to contact the authors please write to:

c/o CIFE, Civil Engineering,
Stanford University,
Terman Engineering Center
Mail Code: 4020
Stanford, CA 95305-4020

Knowledge-Based Support for Concurrent, Multidisciplinary Design[1]

Raymond E. Levitt[2], Clive L. Dym[3], and Yan Jin[4]

1. ABSTRACT

Artificial intelligence (AI) applications to design have tended to focus on modeling and automating aspects of single discipline design tasks. Relatively little attention has thus far been devoted to representing the kinds of design "metaknowledge" needed to manage the important interface issues that arise in concurrent design, that is, multidisciplinary design decision-making. This paper provides a view of the process and management of concurrent design and evaluates the potential of two AI approaches—blackboard architectures and cooperative distributed problem-solving (CDPS)—to support the concurrent design of complex artifacts.

A discussion of the process of multidisciplinary design highlights elements of both sequential and concurrent design decision-making. We identify several kinds of design metaknowledge used by expert managers to: partition the design task for efficient execution by specialists; set appropriate levels of design conservatism for key subsystem specifications; evaluate, limit and selectively communicate design changes across discipline boundaries; and control the sequence and timing of the key (highly constrained and constraining) design decisions for a given type of artifact.

We explore the extent to which blackboard and CDPS architectures can provide valid models of and potential decision support for concurrent design by (1) representing design management metaknowledge, and (2) using it to enhance both horizontal (interdisciplinary) and vertical (project life cycle) integration among product design, manufacturing and operations specialists.

---

1 Working Paper. For review and comment. Please do not cite without permission of the authors.

2 Professor of Civil Engineering and Associate Director, Center for Integrated Facility Engineering, Stanford University, Stanford, CA.

3 Professor of Civil Engineering, University of Massachusetts, Amherst, MA.

4 Research Associate, Department of Naval Architecture, University of Tokyo, Tokyo, Japan.

2. INTRODUCTION

The design of artifacts and the processes required to implement them is an ubiquitous human problem-solving activity. A working definition of design is provided by [Dym 1990a]:

> Engineering design is the systematic, intelligent generation and evaluation of specifications for artifacts whose form and function achieve stated objectives and satisfy specified constraints.

Design tasks typically require several iterations of synthesis, analysis, and evaluation [Asimow 1962; Dym 1990b]:

- *Synthesis* is the assembly of primitive design elements or partial designs into a configuration that self-evidently satisfies a few key specifications and constraints. ("Performance specifications" can be viewed as constraints that are optimized rather than just being satisfied [Simon 1975], but we maintain the distinction herein.)

- *Analysis* involves computations or deductions based upon attributes of the current synthesis to assess the extent to which the synthesis satisfies other, more subtle or complex specifications and constraints.

- *Evaluation* is the process of comparing the analysis of the current synthesis against the specifications and constraints for the artifact to determine whether that synthesis is acceptable. An evaluation that a given synthesis is unacceptable will typically prompt a new synthesis aimed at improving the artifact's performance to satisfy the violated specifications or constraints.

The design of most complex artifacts requires combining the expertise of specialists in several discrete areas. The various kinds of expertise involved in generating the design of a complex artifact can be employed in either a "sequential" or a "concurrent" mode. In sequential design, the design task is broken down into a sequence of design subtasks for which the output of one designer's decision process serves as input to another. In concurrent design, distributed problem solving by specialists from different disciplines (e.g., process, structural, mechanical, electrical) or functions (planners, designers, manufacturers' constructors, and operators/users) occurs simultaneously, and design decisions must either be coordinated in real time or reviewed for consistency and modified as needed during periodic reviews. Routine or semi-custom design can

often be done sequentially because experience has produced a feasible sequence of design decisions that can be executed serially and without much backtracking. The design of complex or substantially innovative artifacts typically requires a significant degree of concurrent decision making by its designers.

The present capabilities of computer tools to support concurrent, multidisciplinary design are quite limited. Efforts to employ computers as design aids have to date been channeled primarily within individual disciplines or functions. For example, in building design, "islands of automation" have evolved to aid structural engineers in designing and analyzing structural systems or to assist mechanical engineers to compute the air conditioning requirements of a building. Moreover, computer tools used by the various specialty contractors to plan and schedule their parts of the construction process are completely isolated both from each other and from the tools used in design. Similarly, in manufacturing, most MRP or factory scheduling tools are isolated from the CAD/CAE tools used to design the products produced in the plant.

We note that some degree of concurrency is present in the design of most artifacts and that effective management of concurrent design for a class of artifacts involves specific, experience-based coordination and control metaknowledge. Therefore, knowledge-based architectures such as blackboards and CDPS systems that were developed to represent task coordination and control knowledge knowledge should be able to provide powerful models of and enhanced decision support for the management of concurrent design. Our paper explores this proposition.

3. MANAGING THE DESIGN OF COMPLEX ARTIFACTS

In this section we discuss the nature of design tasks—sequential vs. concurrent—and list some of the kinds of knowledge needed to coordinate design decision-making for complex artifacts.

3.1. SEQUENTIAL VS. CONCURRENT DESIGN

In sequential design, a tentative design synthesis is developed by one designer—often acknowledged as the "lead discipline" designer—which addresses some of the key performance specifications and constraints for the artifact. Examples of this lead designer role include the architect who develops the conceptual design of a building and the process designer who develops a process block diagram for a chemical plant. This conceptual design is then refined and evaluated iteratively in terms of a broader set of product and process specifications and constraints by other design specialists. For all but the most routine kinds of design, several

iterations involving considerable amounts of backtracking are typically necessary before a feasible design synthesis emerges from a sequential design process of this type. Moreover, time pressures typically permit the generation and evaluation of only one—or a very few—alternatives for major projects in this style of design.

Concurrent design occurs when the multiple design specialists required to design a complex artifact work in parallel. This approach has the potential to save time and thus permit the consideration of more alternative design syntheses. However, without adequate coordination, it runs the risk that the design details developed by the various specialist designers for a given artifact may be locally suboptimal, redundant, or mutually incompatible [Logcher 1979].

In the course of progressing from a high-level specification of an artifact to its detailed design and manufacturing, most real-world design processes involve both sequential and concurrent design elements. In many cases, the flux of sequential and concurrent design and manufacturing activities follow a predictable pattern that has been referred to as "matrix swing" [Levitt 1984].

In the early stages of conceptual design, high-level specialists from all involved design disciplines meet frequently to be sure that (1) key performance specifications and constraints for each discipline will be met and (2) assumptions of designers from a given discipline that may have implications for other disciplines are communicated and understood. This represents a form of tightly-coupled concurrent design. But extensive coordination meetings of experienced designers impose a significant overhead cost on design and must be used sparingly, particularly where the specialist designers are employed by separate organizations and are geographically dispersed.[5]

As a conceptual design becomes firm, key subsystems that fall primarily within the area of expertise of a single design specialist can frequently be demarcated, and the most important interface issues to be coordinated with other specialists can be identified. Once this occurs, design activities can proceed concurrently, in a much more loosely coupled fashion. During this stage of design few meetings are held: Most designers can proceed to develop partial design

---

[5] In the early stages of designing the space shuttle, NASA conducted huge meetings consisting of high-level design representatives from contractors for all of the shuttle's major subsystems to deal with this problem of "configuration management" or change control. One of the authors attended such a meeting as an observer and conservatively estimates the salary, travel and administrative costs of running each of these meetings at $250,000.

solutions covering the components, materials, dimensions and other attributes of the subsystems assigned to them, with only limited communication across discipline- or subsystem-based boundaries.

In many types of design, this activity can proceed to the point where about one-third of the total budgeted design hours have been expended, at which time additional, concerted review and coordination activities occur. Subsystem attributes become known in sufficient detail that more detailed and specific interface issues can be addressed. For example, the adequacy of a particular beam to support the designed mechanical equipment (along with other known loads) in a structure can now be analyzed. Or, the ability of the power supply and distribution system of a process plant to meet the demands of component devices (whose power consumption has now been determined more exactly) can be computed. Conflicts identified at this stage will typically require one or more design specialists to redesign some elements of the artifact, while others can proceed to flesh out design details without backtracking.

In routine design, the constraints imposed by other subsystems or specialties become embedded implicitly over time into standard component designs, design procedure manuals, or other kinds of personal and institutional memory. For the design of innovative or unique artifacts, this implicit knowledge about the constraints posed by other subsystems or specialties has not yet been learned by the design team. As a result, iterations at this stage of design are often sequential rather than concurrent; they involve frequent negotiations among specialists to determine who will relax conflicting constraints or performance objectives.

As final detailed descriptions of all subsystems are determined for a large and complex artifact such as an aircraft, software system or power plant, an overwhelming amount of information must be shared and coordinated to ensure continued fit—functional, spatial and temporal—between each of the artifact's subsystems. In traditional design practice, this often involves the circulation to all affected parties of paper drawings which are "red pencilled" to identify conflicts, errors or omissions. This process is largely sequential in nature, prone to errors, and extremely time-consuming.

The means that are used to coordinate such engineering project teams include a consistent repertoire of four coordination devices used by any organization: *rules and standards*, *direct supervision*, *hierarchical planning*, and *face-to-face meetings* [Logcher 1979; Thompson 1967]. These coordination techniques are used successively, in the order listed, as the interdependence among specialists' tasks and their degree of novelty or uncertainty increase. Since

interdependence and uncertainty are ever present in engineering, engineering organizations rely heavily on face-to-face meetings to share information and to coordinate decisions.

3.2. KINDS OF KNOWLEDGE USED IN MANAGING DESIGN

Expert designers and design managers accumulate several types of experiential knowledge for coordinating the design of classes of artifacts . They develop heuristics or rules of thumb to aid them in making the following kinds of decisions:

- *Defining efficient boundaries between the artifact's subsystems* in order to minimize the number and severity of interface issues that must be coordinated across design specialties. Of course, the definition of subsystem boundaries involves other considerations besides ease of coordination. It includes considerations of available fabrication methods and of subcontractors' or suppliers' capabilities to deliver particular combinations of components.[6]

- *Making good guesses about the required degree of conservatism in the initial performance specifications for each subsystem.* Experienced design managers use their experience to set functional performance requirements and cost contingencies for subsystems to achieve a good balance between the expected costs of excess conservatism (e.g., wasted materials, labor, space, power, etc.) and the potential costs of under-design (e.g., loss of design flexibility, time lost to redesign, or increased probability of failure of the completed artifact).

- *Evaluating and limiting proposed changes in subsystem specifications.* Assessing whether a proposed change is essential (the artifact will fail to meet an important specification without the change), desirable (the value of enhanced performance of the artifact will outweigh the technical and administrative costs of making the change), or undesirable (the costs of making the change will exceed its benefits) involves experience-based knowledge. The importance of this type of knowledge is enhanced by the fact that

---

[6] The latter may be viewed as a form of microeconomic institutional learning about how to package components rationally. Of course, as product designs, manufacturing technologies or factor prices change, these constraints on packaging of subsystems may become obsolete, and can create significant barriers to more rational reallocation of design and manufacturing responsibilities among project participants.

the impact of administrative changes varies substantially during design and manufacture. Further, participating designers vary in their ability to tolerate disruption and ambiguity.[7]

- *Determining which specialists are likely to be impacted by changes that do get approved*, and communicating information about such changes selectively to the affected parties.

- *Controlling the timing and sequencing of that small number of key design decisions* that usually turn out to be highly constrained or constraining for a particular class of artifact. These key decisions have the potential to affect many disciplines.

4. RESEARCH ON COMPUTER SYSTEMS TO AUTOMATE DESIGN

Since the early 1960s, a great deal of research and development effort has been conducted in universities and industry to automate certain aspects of design. Early work such as the ICES project at MIT was aimed at substituting computation for closed form mathematical analysis of structural, thermal and other loadings on engineered systems [Roos 1967]. As the need to manage large volumes of data arose in such computations, engineers helped to push the frontiers of database technology forward. Subsequently, computers were used to automate drafting, a trend which is still continuing.

Up to about 1983 most designers—even in the largest and most sophisticated firms—used batch programs to analyze syntheses that they developed manually. Their designs were then drawn by drafters either manually or, more recently, at CAD stations. Only in the last few years have links between engineering analysis, graphics and database technologies created the first cohort of CAD/CAE tools that provide meaningful support for design synthesis. Engineers can now design automobiles, aircraft, and buildings at graphics workstations, bypassing the drafters.

4.1. 3-D CAD FOR DOCUMENTING AND VERIFYING DESIGN SYNTHESES

As designers in more and more fields begin to develop syntheses on workstations, their partially-developed designs become machine-readable. This creates new opportunities to exercise

---

[7] A major constructor of petrochemical facilities has estimated that, in the period from 75% completion to 100% completion of construction, the indirect costs of design changes are about 150% of the identifiable direct costs. This contractor attempts to freeze designs at about the 75% point, claiming that subsequent changes can often be more efficiently implemented—and perhaps more easily resisted—when carried out as retrofits to the completed facility.

coordination and control of concurrent design in real-time. 3-D CAD models provide a machine-readable and electronically communicable alternative to the red pencil approach for consistency checking and communication across disciplines. If an integrated 3-D model with layers for each discipline's input is used to coordinate the design, the model is not only machine-readable—it is "machine-usable" in checking for spatial conflicts.

This form of automated verification of *spatial consistency* ("interference checking") among concurrent designers has proven to be extremely valuable. It has in some cases eliminated the need for *physical* modeling of artifacts, e.g., plastic scale models of refineries, full scale mock-ups of new aircraft , and so on. However, the current generation of computational design tools provides little or no assistance to design managers in verifying *functional consistency* among an artifact's subsystems. The software architectures discussed in this paper begin to address the need for intelligent functional coordination among subsystem designs developed by separate design teams.

Incompatible database architectures for CAD/CAE systems used by different design specialists pose one obstacle to computer-aided design integration. An administrative solution to this problem is to insist on the use of a single CAD/CAE package, provided that it can support all the kinds of analysis needed. Of course, this is most easily adopted by horizontally and vertically integrated firms.[8] Several parallel research efforts are attempting to address this barrier to design integration. One approach involves development of "intelligent" database management systems that can propagate and resolve constraints among related attributes of a design to specify additional attributes [Stonebraker 1986]. An alternative, more evolutionary line of attack on the "Tower of Babel" problem in design is the development of knowledge-based database interfaces that can mediate between several incompatible but related databases. One example of this approach is KADBASE [Howard 1987].

Note that to date, CAD systems have been useful first for *documenting* design and more recently for the *spatial coordination* of designs. They have not yet provided any significant support for *automated design synthesis.*

---

[8] Japanese engineering/construction firms, which are generally far more integrated than their US competitors, have adopted this strategy with considerable success in some market sectors.

4.2. AI TECHNIQUES FOR AUTOMATION OF DESIGN SYNTHESIS

AI contributions to design have come from the application of knowledge-based (expert) systems (KBESs) to design tasks [Dym 1990b]. KBES applications typically encapsulate and represent knowledge in a well-specified part of a single narrow domain (e.g., diagnosis of bacterial meningitis, evaluation of molybdenum ore deposits). This distinguishes the strong or knowledge-intensive methods that KBESs use for solving problems from weak or domain-independent methods such as mathematical optimization techniques. The latter rely on extensive search and computation in solving problems, but they are potentially applicable across a broader range of problem domains. It has been observed that the capability of a KBES degrades rapidly and ungracefully outside of its scope of intended application. In the words of one of the pioneers of this technology, the KBES falls off the edge of its "knowledge mesa" [Feigenbaum 1977]. Since KBES applications have usually represented and reasoned with knowledge that is highly specialized, it might seem like an impossible task to model the many kinds of expertise used by diverse engineering specialists within a single KBES.

Furthermore, design problems are inherently different from the types of problems for which AI techniques have thus far provided powerful solutions. In terms of Amarel's [1968] spectrum of tasks, from *selection* to *formation*, most KBES applications have been to *selection* problems such as diagnosis or interpretation for which the universe of potential solutions is relatively small and easy to define. In contrast, design synthesis is a *formation* task in which an infinitely large number of potential solutions must be synthesized from elemental features or components to meet a set of requirements or specifications for the completed artifact. As a result, design applications of AI have been slower in coming. Nevertheless, significant progress has been made.

AI techniques have been explored for automating some aspects of mechanical design synthesis [Brown 1989] as well as for structural systems [Maher 1984]. A number of these attempts at automating design have now progressed beyond the laboratory. The PRIDE design system for designing paper-handling subsystems for copiers paths, implemented in Xerox PARC's *LOOPS* language, is now in routine use at Xerox [Mittal 1986]. The IBDS system, implemented in the *Design*$^{++}$ system, is being routinely used to design industrial boilers by Tampella in Finland [Riitahuhta 1988].

Emerging commercial knowledge-based design systems such as Design Power's *Design*$^{++}$, ICAD's *ICAD* or Wisdom System's *Concept Modeler* can be used to generate design syntheses

for semi-custom products. These systems can be developed to run in a fully automated mode; or they can keep human designers in the loop via human interface tools such as product structure graphs and geometric CAD visualizations of the design, to which the human designer can react at each stage of design development. The applicability of these tools is limited to semi-custom products—those for which a "once through" sequential design approach can be defined. "Design management" in such tools is hard-wired into the rules used to determine component inclusion and refinement in a sequential manner. High-level control of the process must be provided by the human user intervening to carry out iterative sensitivity analyses in a "What if?" mode.

Research on the use of AI techniques to coordinate non-routine design decision-making across disciplines has been proposed and initiated in a few places (see [SIGMAN 1990; IJCAI 1989; Pohl 90; Sriram 1989]. However, the focus of this research has been primarily on using AI techniques to automate specific design tasks rather than on using AI techniques to provide decision support for the *management of design.*

In the next two sections we discuss two architectures, blackboard architectures and cooperative distributive problem solving (CDPS), which have been used to model concurrent tasks carried out by human and computer agents. Thus, instead of the now-traditional focus of KBES applications on automating individual tasks, we examine architectures that support the interactions of multiple agents because this is essential for the support of multidisciplinary design activities. Our discussions of blackboard and CDPS architectures will evaluate their potential to provide models of and decision support tools for managing concurrent design.

Two related discussions are worth noting. In a review of AI applications to planning and scheduling, Levitt [1987] concluded that AI techniques offered considerable value as decision support tools for human synthesis in planning, scheduling and controlling project activities, and relatively less value as substitutes for humans in synthesis tasks. We shall argue here that the same general conclusions hold at this time for multidisciplinary design of all but the most standard artifacts. Also, in a somewhat broader context, Dym [1991] argues that knowledge-based approaches provide an essential building block for identifying and integrating all the kinds of engineering knowledge needed to perform complex engineering analysis and design tasks.

5. BLACKBOARD ARCHITECTURES

Blackboard architectures for KBESs evolved out of the HEARSAY project on speech recognition at Carnegie Mellon University [Lesser 1975; Erman 1980] and the HASP project on

sonar data interpretation at Stanford University [Nii 1982]. Reasoning at multiple levels of abstraction, where each level of reasoning helps to resolve uncertainties at other levels, was needed for both speech understanding (phonemes, words, sentences) and sonar ship identification (vibrations, sources, platforms). The blackboard architecture is an attempt to organize such multilevel or multidisciplinary kinds of knowledge into a single KBES. As such, they might be viewed as good candidates to model the diverse kinds of knowledge required to execute and manage concurrent design. Recently, attempts have been made to use blackboard architectures developed for integrating knowledge at multiple levels of abstraction in domains such as concurrent engineering, which involves multiple sources of expertise at the same level of abstraction [Tommelein 1989a, 1989b].

The blackboard metaphor for integrated computer-based reasoning with multiple sources of expertise was abstracted from HEARSAY and adapted in HASP and other subsequent applications. It is now incorporated into a number of general-purpose blackboard programming environments. We set the stage for this section with a discussion of how the decision making of specialist designers is coordinated in human organizations. Then we show that blackboard architectures incorporate some of the same kinds of coordination devices to manage concurrent reasoning by their separate knowledge sources. We touch on some details of blackboard architectures by outlining how representation, reasoning, and control are implemented in the BB1 blackboard system [Hayes-Roth 1985]. We conclude with an evaluation of the suitability of blackboard systems for modeling concurrent engineering tasks.

The blackboard architecture described in this section is a KBES architecture which incorporates some of the same four means for coordination of its disparate knowledge sources as do human engineering organizations (cf. Section 3.1). We will see that a blackboard architecture:

- Models the specialization of expertise in concurrent design by organizing chunks of related knowledge about some aspect of the design domain (corresponding to organizational rules, standards and procedures for design decision-making) into discrete modules termed *domain knowledge sources*;[9]

[9] These knowledge sources are roughly, but not closely, analogous to rule sets in integrated rule/frame/OOP environments such as IntelliCorp's KEE or Neuron Data's Nexpert Object.

- Incorporates the information sharing—although not the negotiation—that occurs in face-to-face meetings. Knowledge sources communicate indirectly with one another by writing to, and reading from, a common data structure called a *blackboard*; and

- Approximates hierarchical planning and direct supervision by having one or more *control knowledge sources* that form a solution strategy against which it can evaluate and implement recommendations from domain knowledge source as problem solving proceeds.

Several general-purpose blackboard architectures with these three basic features—but varying in the details of their implementation—have been abstracted from applications that used the blackboard style of reasoning. The evolution of these blackboard architectures for problem solving through a series of applications is thoroughly reviewed by Nii [1986, 1989]. To ground our discussion of blackboard architectures, we will next briefly describe how representation, reasoning, and control are implemented in one blackboard system, the BB1 blackboard architecture.

5.1. Overview of the BB1 System

The *BB1* blackboard architecture is a general-purpose blackboard architecture originally developed to support opportunistic planning [Hayes-RothB 1985]. BB1 grew out of the Opportunistic Planning Model (OPM) system developed at Rand Corporation [Hayes-RothB 1979] and was strongly influenced by developments in the HEARSAY projects [Lesser 1975; Erman 1980] in terms of the concept of an abstract blackboard architecture with control being viewed as a problem-solving process run in the blackboard framework. BB1 was subsequently employed for several other classes of problems, including intensive care patient monitoring and control [Hayes-RothB 1989]); construction site layout [Tommelein 1989a]); case-based reasoning about structural design [Wang 1989]; and project planning [Darwiche 1989]). BB1 is implemented in Common Lisp and has been widely distributed and used in many research efforts besides those mentioned here. Our discussion of BB1 is adapted from [Tommelein 1989a] and [Dym 1990b].

5.2. The Blackboard Metaphor in BB1

The BB1 architecture emulates a "structured meeting" in which a number of participants—here called knowledge sources (KSs)—are faced with a problem that is described on a blackboard

(BB). None of the KSs can solve the entire problem on its own, in part because no single KS has enough knowledge, and in part because the KSs are distributed over different levels of abstraction of the problem domain. That is, the KSs vary not only in their knowledge but in the level of detail with which that knowledge is concerned. However, each KS may be able to contribute problem-solving steps which, when combined in a reasonable sequence, lead to a solution. By looking at the BB, KSs know when it is appropriate for them to focus their attention and when it is proper to propose an action. The KSs can communicate with each other only indirectly, by making changes on the blackboard. In each step toward the solution, one and only one KS gets to execute its proposed action by being allowed to make changes to the BB. In reaction to such changes on the BB, other KSs may now focus their attention or propose to take action.

The "meeting" has a moderator —here called the scheduler—which, at each cycle, evaluates each of the contributions or actions proposed by the various KSs and selects one of them for execution. Thus, as embodied in the scheduler, the problem-solving style of BB1 is *hierarchical*, in its distribution of KSs at different levels of abstraction and in its assignment of authority for control to the scheduler; *incremental*, as it only does one piece of the puzzle at a time; and *opportunistic*, in adjusting its strategy for picking the most appropriate KS to call on at any stage in the problem-solving process.

5.2. REPRESENTATION, REASONING AND CONTROL IN BB1

All concepts (objects, constraints, events, etc.) in a BB1 knowledge base are represented by frames that can have any kind of user-defined attributes or links to other objects and that can inherit attributes over specific links (not just abstraction links). Concepts in a BB1 knowledge base thus form a conceptual graph defined by all of the relationships existing between them, including—but not limited to—the abstraction relationships [Sowa 1984].

For clarity and flexibility, the semantic net or conceptual graph of concepts in a BB1 knowledge base is layered. Concepts specific to a particular application domain are grouped in what is termed a blackboard (BB), itself part of a knowledge base (KB). System-level concepts applicable to many domains are layered into higher-level blackboards that form part of the overall conceptual graph for a knowledge base. A BB1 knowledge base thus contains several blackboards, including system- and application-level BBs, a problem BB containing the description of the current problem, and a solution BB on which the evolving solution to the problem is stored.

Knowledge sources in BB1 are responsible for reasoning. KSs are similar in structure to situation-action rules and reside on their own blackboard. They contain the knowledge that BB1 will apply to make its inferences from the current state of the problem and solution BBs. These KSs are not designed to "chain" together as in traditional rule-based systems; rather, they are independent entities whose antecedents can become true on the basis of facts stated on any of the BBs and whose consequents—upon execution—post new facts onto the frames in any of the BBs. Thus, KSs need not be "aware" of each other's presence. KSs, as all other concepts in BB1's conceptual graph, are represented by means of frames.

Two types of knowledge sources are distinguished in BB1. *Domain knowledge sources* are application-dependent and are specific to the problem-solving method that is used. *Control knowledge sources* contain so-called metaknowledge which allows the BB1 scheduler to assign priorities as problem solving proceeds. For example, control KSs express strategic knowledge on the desirability of domain actions, as well as on the desirability of control actions. The control knowledge sources allow a BB1 application to alter its strategy *dynamically* and select its actions *opportunistically*.

The BB1 scheduler embodies an incremental problem solver, so it activates only one KS at a time (see [Tommelein 1989a] and Chapter 8 of [Dym 1990b] for details)—even though there can be multiple executable KSs at any cycle. Control KSs make changes to the control data BB, on which they can post or modify one of three things: a strategy, a focus, or a heuristic. *Strategies* provide high-level statements of what needs to be done to solve the problem. *Focuses* (or foci) do the same, but they describe the preferred steps in more detail and are used by the scheduler to determine *which* of the executable KSs is most desirable at that point. *Heuristics*, which implement foci, prescribe ways for the scheduler to compute this desirability.

Thus, we see that the blackboard architecture emulates a design or problem-solving process that is opportunistically sequential, i.e., one in which a sequence of design steps are taken based upon some measure of making the "best" possible progress at any point in the process. However, the process is controlled centrally by the scheduler or meeting moderator, much as a project manager controls the interaction of the specialist designers in a meeting, that is, by recognizing those who can contribute to the flow of the discussion. In fact, the metaknowledge of the control KSs might be said to embody the experiential knowledge of the project manager or design team leader. This suggests that the individual designers or agents are rather independent of each other, do not require much knowledge of each other's capabilities, and are best instantiated only when their

specific expertise is required. Thus, these designers have little autonomy, dependent as they are upon the scheduler or project manager to hand them their assignments.

5.3. BLACKBOARD ARCHITECTURES FOR CONCURRENT ENGINEERING

We cannot yet evaluate the quality of solutions produced for concurrent engineering because the applications developed so far in BB1 and other blackboard architectures such as GBB [Corkill 1986a, 1986b] and DICE [Sriram 1989] are still in the prototype stage. However, on the basis of our description of concurrent engineering (cf. Section 3), we can examine the extent to which a blackboard approach captures the essence of a concurrent design process.

We have argued that concurrent engineering, as practiced by humans for routine or standard kinds of engineering tasks, usually involves decomposition of the task into loosely-coupled subtasks performed by separate specialists who are locally independent of each other. The specialists here are analogous to the domain knowledge sources in the blackboard analogy, and they interact with each other only through the blackboard, that is, through a common data structure accessible to all of them and to the design leader. Coordination of decision making occurs only through periodic design reviews, and the hierarchical coordination of decision making at each step of problem solving is implicit in the control KSs. Thus, since blackboard architectures appear to call on the specialists through the exertion of centralized, hierarchical control, they may be good models of how human organizations perform routine concurrent engineering when the design can be so effectively decomposed. In fact, even the sequential, opportunistic style of this hierarchical control can be said to mimic the way that unexpected problems and "fires" are dealt with as inconsistencies arise from the integration of what were thought to be loosely coupled design tasks.

However, it is not clear that a blackboard architecture is entirely appropriate for modeling a full range of concurrent engineering activities, especially when we consider the design of complex artifacts involving significantly variant or innovative subtasks. The restriction that the design task be decomposable into loosely coupled subtasks means that the specialists need have little or no knowledge of what other specialists are doing or their capabilities, a situation that does not obtain for complex, innovative design. The performance of novel tasks with a high degree of interdependency, in which parallel reasoning is likely to produce many inconsistencies, clearly requires knowledge exchange and negotiation between subtask specialists. This, in turn, requires direct contact between the specialists, while the conventional blackboard approach permits only one specialist at a time to take decisions which are then broadcast to others via the blackboard.

Another drawback of blackboard architectures for complex, innovative design is that their opportunistic style of problem solving requires *a priori* specification of the control knowledge. One or more problem-solving strategies involving task decomposition and the sequenced solution of partial problems must be specified in order to guide problem solving in a blackboard system. This imposes a certain rigidity which inhibits more globally opportunistic revisions of problem solving strategies.

One approach to using blackboards for complex design might be to model individual specialists as independent blackboards that could communicate directly (in order to model the negotiation process). Then these individual blackboard agents would be integrated into a "global design leader" blackboard that serves to schedule, maintain and monitor the global design. There are important issues to be addressed in such an extension of the blackboard architecture, including: the nature and extent of communication between individual specialist blackboards; the degree to which the global scheduler could analyze and critique an evolving design; and the extent to which the global scheduler can identify and communicate repairs for design inconsistencies. We will see that some of these ideas of blackboards within blackboards are embodied in the CDPS approach discussed in the next section.

Another approach to using blackboards for complex concurrent design depends upon the degree to which good interfaces can be developed for human users of blackboard KBESs. If user-friendly (by domain expert users!) interfaces can be developed, then a human designer can function as another knowledge source in the system, probably with a higher priority than the system's internal knowledge sources. In this regard, the SightView interface for the SightPlan system suggests intriguing opportunities for KBESs that exploit knowledge-based interactive graphics [Levitt 1989]. Such an interface also combines and manages the input of multiple experts—of both the flesh-and-blood and the dirty silicon varieties—while exploiting the strengths of both humans and computers in problem solving [Levitt 1987, 1989; Mittal 1986].

There is also a larger, more philosophical question involved in the quest to model the performance of human designers in complex organizations. Why should departing from human problem-solving approaches be a disadvantage for a KBES? Computers have very different strengths and weaknesses in problem solving than do humans, so that a system for machine reasoning in the domain of concurrent engineering should probably not be restricted to emulating

the problem-solving styles of humans.[10] Nevertheless, in seeking to draw an analogy with the concurrent engineering process used by teams of human designers, the cooperative distributed problem-solving approach described next may be a better fit.

6. COOPERATIVE DISTRIBUTED PROBLEM SOLVING

We now turn to cooperative distributed problem solving, the idea of distributing a computation or a task over an assembly or network of processors or agents that, together, solve a complex problem or achieve some overarching goal. Each KBES within the network works on a piece of the larger problem, hopefully in cooperation with other KBESs in the network. In decomposing the problem solving process, we must confront the issues of how we represent and decompose the original, global task and how we maintain control of the solution process so as to ensure a coherent outcome. CDPS is a rapidly developing set of ideas which, as we will point out, provides a suitable approach for managing multidisciplinary design. Thus, it it useful to compare CDPS to blackboard architectures, although we begin by commenting on more traditional distributed processing.

In distributed processing, which is often viewed as a hardware issue, each of the processors in a network is strongly coupled to a central processor with which it communicates directly [Stankovic 1984]. The individual processors typically do their work alone in a predetermined and prescribed manner, in which they may share data and communication resources. However, the processors do not communicate about goals nor about their tasks, both of which are preset by the system designer. This means that the processors can deal with their control issues independently with only local processing. The independence of control, which is predetermined, reflects the fact that the problems solved in distributed processing are decomposable into separate tasks that require little interaction and can therefore be done in parallel without intermediate coordination steps.

Thus, the individual nodes need have little knowledge of the tasks performed by or within the network, other than their own. Further, these individual task agents will not function outside the context provided by the designer of the network. This model is a sharp departure from the opportunistic control offered by blackboards, and it is not particularly useful for the complex engineering organizations that we have outlined in Section 3. In such organizations, tasks are

[10] This question is explored in some depth by Tommelein [1989b].

generally not decomposable into highly compartmentalized, independent subtasks that can be performed in parallel with a rigid, predetermined control structure. Thus, the distributed processing model is not useful for describing concurrent engineering.

The CDPS approach resembles a blackboard system to a certain extent, but CDPS goes beyond the ideas of knowledge sources operating within a particular hierarchical structure and controlled by a centralized scheduler. In fact, a very simple model of CDPS would replace each KS in a blackboard with a powerful KBES free to operate at all levels in the hierarchy. Moreover, it would give each KBES in the network the autonomy to do its problem solving, based on its own knowledge and resources, without waiting to be ordered into action by a scheduler. These elementary ideas seem much more consistent with how a complex engineering task would be done in a human, organizational setting.

Blackboard architectures are, however, suitable for accommodating ideas of CDPS. Individual KBESs can be sets of domain and local control knowledge sources, and they could communicate with other KBESs through a shared blackboard space. Moreover, as we have hinted above (cf. Section 5.3), we can model the agents in a CDPS system as blackboard systems which cooperate with each other. In fact, the blackboard architecture concept has been adopted by many projects in CDPS of which DVMT is probably the best known [Corkill 1983; Durfee 1987b].

In this survey, we focus on CDPS in which, broadly speaking, the work is done through a network of loosely coupled, semi-autonomous problem-solving agents or nodes. In a CDPS network, each node is capable of sophisticated problem solving and cooperative interaction with other nodes to solve a single problem. Each node may itself be a complex problem-solving system that can modify its behavior as circumstances change and can plan its own communication and cooperation strategies with other nodes. Our discussion begins with an overview of applications of and motivations for CDPS. We then discuss some characteristics of CDPS systems, providing an architectural view of how CDPS systems might look. Next, we describe some basic problems of,and approaches to building, a CDPS system for concurrent engineering design tasks. In organizing our brief survey, which follows that in [Dym 1990b], we have drawn heavily on the work of [Bond 1988; Durfee 1989; Gasser 1989; Huhns 1987].

6.1. MOTIVATIONS AND APPLICATIONS

CDPS is an important area for several reasons. First, hardware technology has advanced to the point where the construction of large distributed problem-solving networks is both possible and

economically feasible. Although current networks may consist of only a small number of nodes, CDPS networks can eventually contain hundreds or thousands of nodes. And while exciting hardware possibilities draw nearer, the problem-solving technology required for their effective utilization lags behind. Second, there are applications that are inherently distributed, many spatially, some temporally, and some functionally. The distribution usually results from the limited resources in one location, limited communication bandwidth and bounded rationality of single problem solvers. Distributed architecture that matches the distribution of intelligence in the application offers many advantages over a centralized approach. Concurrent engineering appears to be an application that could be enhanced through the use of CDPS techniques.

Third, understanding the cooperative aspects of CDPS is an important goal in its own right. Whether we are talking about social, managerial, biological, or mechanical systems, we know more about competition within them than we do about cooperation. In the same way that the development of AI systems has helped our understanding of problem solving and intelligence in linguistics, psychology, and philosophy, we may find that the development of CDPS networks will serve to validate theories in sociology, management and organizational theory—all of which are related to concurrent design—as well as to fields like biology. That is, research that views CDPS as cooperative problem solving could have a salutary impact on our understanding of cooperative behavior between independent agents ranging from design teams to simple biological organisms. The top-level goals of such a research approach have been nicely summarized as follows: (1) to use parallelism to speed up task completion; (2) to expand what is achievable by resource allocation and sharing; (3) to increase reliability by (selective) redundancy; and (4) to reduce the interference between tasks by avoiding harmful interactions [Durfee 1986]. These generic goals can then be elaborated as sets of goals for the individual agents or KBESs in a CDPS network.

There are four general application areas that seem well suited for CDPS approaches.

- *Distributed interpretation*: Distributed interpretation applications require the integration and analysis of distributed data to generate a (potentially distributed) model of the data. Application domains include network fault diagnosis and distributed sensor networks. In these applications, agents and must exchange enough information in order to form partial interpretations that are globally relevant.

- *Distributed planning and control*: Distributed planning and control applications involve developing and coordinating the actions of a number of distributed effector nodes to

perform some desired task. Application domains include distributed air traffic or ship control, control of groups of cooperating robots, remotely piloted vehicles, distributed process control in manufacturing, and resource allocation in transportation and delivery systems. Distributed planning and control applications often require distributed interpretation for determining and monitoring node actions.

- *Coordination networks*: Coordination network applications involve the coordination of a number of individuals in the performance of some task. Application domains include intelligent command and control systems, multiuser project coordination, and cooperative environments where work is shared among workstations. Concurrent engineering clearly can be viewed through the prism of network coordination, as it is precisely that type of coordination of the engineering process that we described in Section 3.

- *Cooperative interaction among KBESs*: Cooperative interaction mechanisms would allow multiple KBESs to work together toward solving a common problem; this would be one means of applying expert system technology to larger problem domains. One example is the integration of a number of specialized medical diagnosis systems. Another is the negotiation between the expert systems of two corporations about details of price and delivery time on a major purchase. Again, this kind of application is central to engineering practice in general, to the integration of KBES and conventional tools, and to concurrent, multidisciplinary engineering.

The earliest work in CDPS was in three application domains: distributed sensor networks, distributed air traffic control, and distributed robot systems. Each of these applications requires performing distributed interpretation and planning and control tasks. By planning we mean not only determining what actions to take, such as changing the course of an airplane, but also deciding how to use the network resources to carry out the various tasks effectively. These applications are also characterized by a natural spatial distribution of sensors and effectors, and by the fact that both the local interpretation of sensory data and the planning of effector actions are interdependent in time and space. For example, in a distributed sensor network that tracks vehicle movements, our detection of a vehicle in one area would imply that we will shortly sense a vehicle of similar type and velocity in an adjacent area. Similarly, a plan for guiding one airplane must be coordinated with the plans for other nearby airplanes in order to avoid collision. Interdependence also arises from redundancy in sensory data. Often, different nodes sense the same event because of overlaps in sensor range or because different types of sensors pick up the

same event in different ways. The interdependencies among subproblems and the exploitation of redundant and corroborative views require that nodes cooperate in order to interpret and plan effectively. Such cooperation leads us to view network problem solving in terms of solving a single problem rather than as seeking solutions to a collection of independent subproblems.

6.2. CHARACTERISTICS OF CDPS SYSTEMS

We characterize CDPS systems along three dimensions: the typical problem domains; the structure of CDPS systems, and the features of individual agents in CDPS systems.

6.2.1. Problem Domains for CDPS Systems

We begin our characterization of CDPS systems by looking at the *problem domain* in which CDPS systems are designed to work. Problem domains of CDPS are usually large in size and permit natural decomposition of action, perception, resources or control. The resource limitation is often the important factor that makes the problem difficult to solve. For example, the design of a building can be decomposed into a set of specialized subproblems, such as structural design, mechanical design, etc., which interact with each other because decisions made for one subproblem may influence those for other subproblems. Moreover, it is often the case that the lead time of the design is very short. Solving this kind of problem requires two or more agents and/or perspectives which are interdependent with one other. This interdependence among agents raises the need to coordinate of agents' activities, which is the basic problem for CDPS.

From the perspective of designing CDPS systems, an important feature of a problem is its *granularity*, that is, the fineness or level of detail at which the agents in a CDPS network operate. In our discussion, the granularity issue is the question of deciding on the appropriate level of description of the problem, and of the decomposition of the CDPS system. For example, a group of autonomous (or nearly so) cooperating experts would be able to solve problems at a very large grain size. At the other extreme, the problem of computing based on a connectionist model must be of very fine grain size. An important feature of CDPS systems is that their domain problems are often coarse grained. This also means that the decomposition of problems is often at the task level rather than at the statement level.

6.2.2. System-Level Features of CDPS Systems

Next, we describe the characteristics of CDPS networks at the system level by focusing on the *structure* of the networks. Structure is defined by the number of agents included in the networks, their heterogeneity, and their pathways for interaction.

- *Number of agents.* Just as human beings organize themselves into small groups, medium-sized teams, or large societies for solving different problems, the number of agent involved in a CDPS network depends on the overall size and the granularity of the problem. Obviously, more coarse-grained problems require fewer, more powerful agents.

- *Heterogeneity of agents.* The heterogeneity of agents is another important structural feature of a CDPS network because it may influence the coordination strategy of the network. The agents are heterogeneous if they differ in their internal structures, capabilities or purposes. For example, the agents in the distributed traffic control systems [Thorndyke 1981; Jin 1990b] are homogeneous, whereas those in a multidisciplinary design system are explicitly more heterogeneous. Generally, coordination between heterogeneous agents is more difficult than homogeneous agents.

- *Pathways for interaction among agents.* Depending on the problem and resource conditions, the interaction structure of a CDPS system may vary from a heterarchy to a hierarchy. Hierarchical structures work well when there is a need to concentrate either control or results at one point in the network, but they are then sensitive to the loss of a high-level node in the hierarchy. Heterarchical structures may be more robust, and thus less sensitive to the loss of nodes, but they can exhibit increased communication and control problems. It is not obvious whether concurrent engineering in general requires an approach that is either hierarchical, heterarchical, or some combination of hierarchical and heterarchical substructures. What can be fairly said, however, is that complex engineering tasks are often decomposed by technical discipline area in order to accommodate the organizational structure of the institution. This results a set of work groups whose structure is much more heterarchical than hierarchical.

The means of *result formation* and the *communication paradigm* are two additional attributes that describe the system level features of a CDPS network.

- *Communication paradigm.* The paradigm by which communication takes place in distributed problem solvers is either shared global memory, message passing, or some combination of the two. Each of these methods has advantages and disadvantages, and

many systems use both by providing shared memory communication for agents working locally and message passing for communications between agents. The blackboard model described above is the most frequently used model for shared global memory communication. As described in Section 5.2, a blackboard is usually partitioned into levels which are used for different representations or levels of abstraction of the problem at hand. Agents working at a particular level of abstraction view a corresponding level of the blackboard along with the adjacent levels. In this way, data that have been synthesized can be communicated to higher levels, while higher level goals can be filtered down to drive the expectations of lower level agents. Message passing is a straightforward paradigm and it offers a more abstract means of communication than shared memory. Agents doing message passing do not directly affect one another, so they retain the object-like quality of independent agents. Many CDPS systems adopt both modes of communication, using one or more blackboards to share knowledge and using message passing for interagent communication.

- *Result formation.* Result formation specifies how a CDPS network arrives at a solution to a problem. Usually, a network formats its solution either by synthesizing nodes' local solutions to pieces of the overall problem, or by decomposing the original problem so that the solution can be represented by a simple collection of local solutions. A system that uses synthesis for result formation usually takes a bottom-up view of problem solving, while one that works by decomposition takes a top-down view. Synthesis allows agents that are solving inherently distributed problems to work semi-autonomously, and to build up overall solutions through communication. In contrast, decomposition allows agents to decompose and allocate their problem-solving responsibilities and structure their actions and interactions to work more effectively as a team for a particular problem. As will be described in the following subsection, the approaches of negotiation and functionally accurate cooperation accommodate decomposition and synthesis, respectively, and permit intricate organizational structuring in hybrid systems containing both.

6.2.3. Features of Agents in CDPS Systems

The final dimension that enters into our characterization of CDPS concerns the features of the *agents* that form the network. We describe the agents as individuals in terms of agents' autonomy and dynamism.

- The *autonomy* of agents is defined by how control decisions are made by agents during the problem solving process. At one end of the spectrum of autonomy of agents, the form of control by which agents make their control decisions—i.e., deciding on what should be done next—might be preset by the system designer so that the agent has no freedom at all. This is the case of distributed hardware processing. Agents in CDPS networks are, however, either *semi-autonomous,* that is, agents can make their control decisions on their own in an interdependent way, or *autonomous,* that is, agents can make control decisions entirely independently.

- The *dynamism* of an agent describes the ability of that agent to update its knowledge base during problem solving. In CDPS networks, agents can change their incomplete knowledge bases via communication with other agents in a programmable, teachable or autodidactic way. In most cases, updating a CDPS agent's knowledge base to make it more consistent with the current problem solving situation is the direct result of interaction or cooperation between agents.

6.3. ISSUES AND APPROACHES IN BUILDING CDPS SYSTEMS

The problems that we will encounter when building a CDPS system include how we describe and decompose our domain problems, how we distribute the subtasks among agents, and how we can make agents coordinate their activities so that they can act coherently. These problems are fundamental to both the research on CDPS and the application of CDPS to concurrent engineering. In this section, we elaborate these basic problems and briefly describe some recently developed important approaches that are appropriate for solving these problems. In our discussion, we will be concerned with the design of CDPS systems for solving multidisciplinary design problems. We conclude that extending and using the approaches described here may make it possible to build CDPS systems to support concurrent engineering.

6.3.1. Task Description and Decomposition

When a task is to be done by a group of agents, the most immediate question to be answered is, *"How are tasks to be distributed among agents?"* This basic question involves the important issues of task decomposition and distribution. The choice of both decomposition and distribution are critically dependent on how the task is described, because it is the collection of attributes and descriptive categories that provides a language for expressing subproblem and interagent dependencies. At present, the description of tasks is typically carried out by the designer of the

system, and there is little automated assistance available. This also means that in order to accommodate the design metaknowledge that corresponds to task decomposition, we need first to choose a suitable description of the design task.

The problem of task decomposition can be viewed from several perspectives. When we look at a typical decomposition process, a single task is decomposed into smaller ones for reducing the complexity of the problem, because smaller tasks require less capable agents and fewer resources. When we view the decomposition in a general sense, we will need to determine whether there are alternative task decompositions. These are conventionally obtained by alternative problem-reduction operators, corresponding to an *OR branch* in a goal graph, or by problem transformation methods. Successful task decomposition depends greatly on the system designer's decisions on the construction and description of *operators*—i.e., the agents, for problem-solving—because most decomposition processes flow directly from the descriptions of the available agents for problem solving. In multi-agent systems where agents are naturally determined, the decomposition process must consider the resources and capabilities of the different agents for solving the subtasks. In addition, there may be interactions among the subproblems and conflicts among the agents. Difficult problems of decomposition arise because of dependencies among subproblems and among the decisions and actions of separate agents.

Solutions to the problem of task decomposition can be classified into several general classes, although there seem to be few principles, methods, or experimentally validated techniques for doing so.

The first approach is to *pick tasks that are inherently decomposable.* In this approach, the representation of the task contains its decomposition, as in an *AND-OR tree structure* for subproblems. The description of states, of the space of states, and of operators, leads to a natural decomposition, using the selector operations of the data structure. Examples of this approach include spatial decomposition of information in distributed sensor networks and functional decomposition of knowledge in cooperating KBESs. Concurrent engineering would probably exploit both inherent functional and spatial decomposition, incorporating different functions such as analysis and design, but also for doing conceptual design, redesign and manufacturing.

Hierarchical planning is another method of task decomposition. A hierarchical planner does genuine task decomposition. It generates tasks as goals to be achieved. This approach uses abstraction as its decision base and depends heavily on task representation and agent description.

The last approach to decomposition is *decomposition by the system designer.* The system designer takes decomposition issues into consideration in the system design stage, based on the designer's experience, since there seem to be few known principles or methods for automatically decomposing tasks.

In the concurrent engineering context, this means that we will probably need to acquire knowledge about formulating CDPS systems through experimentation with alternate approaches, after attempting to understand how experienced human design managers perform this task.

6.3.2. Task Distribution

Task distribution is the problem of how to allocate particular tasks to particular agents, or in another words, how to assign responsibility for a particular activity. The distribution of a task or the allocation of the responsibility for accomplishing the task is carried out by interaction between agents. A distributing agent can construct a task completely or partially by itself and send the complete problem-solving description to the agent responsible for performing the task, including in this local description the overall problem description, a solution method and a control trigger. Alternatively, an agent can be provided with only that data to which the agent can apply the methods it already has, or it can receive a method to apply to locally available data. In yet another approach, an agent may have access to both the problem description and solution method, and then the control trigger must be provided.

In deciding how a CDPS system should solve the task distribution problem, the system designer should consider issues such as: selecting agents with the most global view to assign tasks to other agents; avoiding overloading critical resources; assigning overlapping responsibility to agents to achieve coherence; assigning highly interdependent tasks to agents in similar spatial or semantic regions of the system; and reassigning tasks as necessary for completing urgent tasks.

The problem of task distribution can be properly solved by a number of approaches, including negotiation, multi-agent planning, organizational structuring, and market mechanisms. We will discuss only the first three approaches because they seem to be most relevant to our engineering concerns.

Negotiation is generally important for CDPS because it is the fundamental strategy observed in cooperation among human agents. It is, at the same time, particularly important for task distribution because most negotiation mechanisms developed to date are used for solving task

allocation problems. The Contract-Net protocol is the best known negotiation protocol and was employed in one of the earliest and most influential research projects in CDPS [Smith 1981; Davis 1983]. Agents use the Contract-Net protocol to make contracts about how they should allocate tasks in the network.

The Contract-Net protocol can be described as an information exchange process, involving task announcement, bidding, and awarding. A *manager* agent announces the existence of tasks to other agents via a (possibly selective) broadcast termed a *task announcement.* Agents having expertise and resources, such as time and availability, evaluate task announcements, and some of these agents (*bidders*) decide to submit bids to the manager (*bidding*). After receiving a number of bids from the bidders, the manager evaluate all the bids and *awards* a contract to the most appropriate *contractor* agent. Once a contract has been established, the manager and contractor communicate privately during the execution of contract. The Contract-Net approach provides a framework for the dynamic allocation of tasks, and decisions about *who performs which task* are more informed because there is local evaluation and mutual selection. It also provides a reliable mechanism for problem solving because control is distributed, the hierarchy is dynamic and failure recovery is inherent.

Researchers have developed other protocols besides the Contract Net negotiation protocol for task allocation. Multistage negotiation was developed by Conry and her colleagues for a class of task allocation problems called distributed constraint satisfaction problems [Conry 1988], and expectation based negotiation was developed by Jin and Koyama for the problems of role (task or responsibility) allocation among agents [Jin 1990a].

Negotiation appears to be the most important approach for engineering design domains, because it is basically a top-down problem solving model and it can help agents to reconcile the disparities and conflicts that are major issues in concurrent design.

Multi-agent planning and *organizational structuring* are also useful mechanisms for task distribution. In a multi-agent planning approach, a planner or collection of planners can combine the work of task decomposition and task allocation by treating agents as specialized resources and objects that interact and depend upon one another. Multi-agent planning can be done using a single centralized planner with global plan synchronization and conflict elimination or by distributed planners that make joint multi-agent plans. Conflicts in task allocations can be resolved by allowing agents with related interests to exchange and elaborate proposed activities, using techniques termed *partial global plans and expectations.* In an organizational structure,

each agent is forced to play a certain role—which is usually predefined—in order to assign designated responsibilities to agents. An agent cannot accept tasks that do not match its role in the organization and, assuming role knowledge is disseminated, cannot be presented with such tasks by other agents. An agent can assume its role in the organization in either a static or a dynamic manner, although the latter is more difficult to implement. Jin and Koyama suggest using negotiation to assign roles dynamically to agents [Jin 1990a].

6.3.3. Coherence and Coordination

After the problem of allocating tasks has been resolved, the next question is, *"How can the system or network be made to behave effectively and efficiently?"* This question raises the issues of coherence and coordination of the system. Coherence refers to how well a system behaves as a unit along dimensions such as solution quality, efficiency and clarity, and the extent to which the system degrades gracefully. Coordination, on the other hand, is the property of interaction among a set of agents performing some collective activities. The degree of coordination is the extent to which agents avoid extraneous activity. In most cases, it is not easy to establish coherence and coordination in a CDPS system. The difficulty in achieving these goals stems from the attempt to achieve them *without a centralized control or viewpoint.* Although the link between coherence and a variety of system attributes is still not clear, much research has been done to achieve global or regional coherence and coordination in CDPS systems.

An organizational structuring approach derives from organization theory and attempts to ensure the coherence of a CDPS system by providing a framework of constraints and expectations about the behavior of agents that focuses the decision making and action of particular agents. An organizational structure of a CDPS network is the pattern of information and control relationships that exist between the nodes and the distribution of problem-solving capabilities among the nodes. It defines general, and relatively long-term, information about the relationships between nodes. Using this information, nodes can ensure that they meet conditions that are essential to successful problem solving, including *coverage*—each necessary portion of the overall problem must be within the problem solving capabilities of at least one node; *connectivity*—nodes must interact in a manner that permits the covered activities to be developed and integrated into an overall solution; and *capability*—coverage and connectivity must be achievable within the communication and computational resource limitations of the system and the reliability specifications of the network.

An organizational structure can specify authority and connectivity for the flow of information and control between nodes in terms of topologies, such as hierarchical, heterarchical, flat structures, groups or teams and market or price systems. Fox has developed a taxonomy of how organizational types evolve as an organization forms groupings, becomes more complex and encompasses more diverse activities [Fox 1981]. He pointed out that *complexity* (high demands on rationality) and *uncertainty* (the difference between information available and the information necessary to make the best decision) are two important factors in deciding how to structure an organization. Complexity forces a distribution of tasks, ultimately resulting in a heterarchical structure. Uncertainty pushes in the opposite direction, vertically integrating tasks into a more hierarchical structure. Jin and Koyama have proposed a method for exerting organizational structuring [Jin 1990a]. The organization can be structured by defining roles and relationships between roles. During the problem-solving process, agents can choose their proper role or roles through an expectation-based negotiation process. Once all the relevant agents have assumed their roles through expectation-based negotiation, the organization is instantiated and agents may pursue their activities coherently according the role specifications, which are an important element of the common knowledge shared by agents.

Another well known CDPS model for achieving coherence is *functionally accurate cooperation* (FA/C) [Lesser 1981]. In this model, the problem-solving system is a collection of semi-autonomous processing nodes, each of which receives information from others. The model is useful for dynamic domains in which data is distributed and noisy and it is generally impossible to maintain complete and consistent information among the processing nodes. In the FA/C paradigm, the processing nodes cooperatively exchange and integrate partial, tentative, high-level results. They detect inconsistencies between local partial results and those from other nodes, and then integrate into local databases only those portions of other nodes' results that are consistent with local information. In this way, the nodes can use the newly integrated results as a basis for supplying information that is missing locally.

This process of exchanging and integrating partial and tentative results can lead to a global consensus despite inconsistent and incomplete local information. Error resolution is an integral part of the problem-solving process. The FA/C approach can reduce communication costs by exchanging only high-level results rather than raw data. It can also reduce synchronization costs by allowing nodes to act autonomously. Because the nodes are designed to process uncertain data and results, the resultant system has increased robustness and concurrency. The disadvantages of FA/C are that we can neither guarantee accurate answers nor predict the time

when the final result will emerge. Because the FA/C approach is inherently oriented to bottom-up problem solving, it appears less relevant for solving design problems.

Improved coordination can be achieved by aligning the behavior of agents toward common goals, with explicit divisions of labor. The multi-agent planning approach provides ways for agents to align their activities by explicitly assigning tasks after reasoning through the consequences of doing those tasks in particular orders. In centralized multi-agent planning, there is a single planning agent who builds the plan that specifies activities for all agents in the network. Key issues to be addressed in this approach are: how do we represent interference between actions of different agents, and how do we choose a planner? Research on distributed air traffic control has proposed a policy for selecting the most appropriate node as a planner [Cammarata 1983]. In distributed multi-agent planning, the plan is produced by cooperation of several agents. The problem here is how to reconcile conflicts between the subplans produced by each agent. One hierarchical approach is to synchronize levels of planning in all the agents, communicating shared variables between goals and resolving conflicts at each level before refining plans to lower levels Corkill [1979]. Durfee and Lesser originated the idea of partial global planning as a mechanism to enable communicating problem solvers to construct mutually coherent plans incrementally [Durfee 1987]. Jin and Koyama proposed an expectation-based negotiation protocol for multi-agent planning in which agents exchange their expectations and resolve the conflicts through compromises between agents [Jin 1990a].

As we noted earlier, complex engineering projects are undertaken within heterarchical organizational structures. We feel that organizational structuring may help us to approach large scale, complex engineering design problems. We can define organization structures to impose long-term responsibilities and disparities among design agents, and let short-term technical conflicts be resolved by agents through negotiation and/or multi-agent planning processes. It is not yet clear for what degree of problem complexity a completely flat structure of design agents can be effective. Imposing a simple or complex hierarchy by introducing mediators or meta-designers to accommodate design management knowledge might provide opportunities for us to build CDPS systems for concurrent engineering design.

7. CONCLUSIONS

We conclude this paper with a recapitulation of the key challenges in using artificial intelligence approaches such as blackboard architectures and CDPS systems to model, and ultimately to enhance, concurrent design by teams of specialized human and computer agents.

7.1. CHALLENGES FOR AI IN CONCURRENT DESIGN

We have identified the following kinds of design metaknowledge as being needed by computer systems to support concurrent design:

- Partitioning artifacts into nearly independent (from a design standpoint) subsystems;
- Selecting levels of conservatism for key interface parameters of critical subsystems;
- Evaluating and limiting proposed changes in system or subsystem design parameters after initial values have been communicated to and used by other design agents;
- Identifying agents impacted by approved changes and selectively communicating to them the minimum information required to maintain consistency; and
- Controlling the sequence and timing of key design decisions by specialized agents.

7.2. APPLICABILITY OF BLACKBOARD AND CDPS APPROACHES

We have provided an exploratory review of blackboard and CDPS systems in the context of concurrent design, and have speculated about the extent to which blackboard architectures and CDPS systems might be employed as computing paradigms in modeling and ultimately aiding concurrent design of complex artifacts.

Based upon this initial evaluation, we have concluded that blackboard architectures are likely to be most useful in modeling and supporting concurrent design for relatively routine artifacts, in which sufficient control knowledge can be defined to establish hierarchical control of the design process. For less standard design tasks in which a "once through" sequencing of design decisions is not obvious a priori, and in which considerable negotiation of constraints among agents will be required, the CDPS approach may be more applicable. We believe that the best approach for using CDPS in this domain will be to assign fixed, specialized roles to agents hierarchically, and then to obtain decentralized control through a process of heterarchical negotiation of decision sequences and constraints among agents with specialized expertise. This style of CDPS appears to match closely the way in which human organizations deal with innovative design tasks. One experimental effort of this type of CDPS approach concerns modeling the processes of concurrent facility design, construction planning and maintenance planning and management for industrial plants. This work will be a synthesis of work already

under way in the area of distributed planning for ship collision avoidance [Jin 1990a] and current research on "virtual design teams." [Cohen 1990]. Currently available hardware and software tools appear to be adequate to support initial experimentation in this area.

8. ACKNOWLEDGEMENTS

The authors are grateful to Dr. Barbara Hayes-Roth of the Knowledge Systems Laboratory of Stanford University for helpful comments on an early draft of this paper.

9. REFERENCES

[Amarel 1968] S. Amarel, "On Representations of Problems of Reasoning About Actions," in D. Michie (Editor), *Machine Intelligence 3*, Edinburgh University Press, Edinburgh, 1968.

[Asimow 1962] W. Asimow, *Introduction to Design*, Prentice-Hall, Englewood Cliffs, NJ, 1962.

[Bond 1988] A. H. Bond and L. Gasser, *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann, San Mateo, CA, 1988.

[Brown 1989] D. C. Brown and B. Chandrasekaran, *Design Problem Solving: Knowledge Structures and Control Strategies*, Pitman, London, 1989.

[Cammarata 1983] S. Cammarata, D. McArthur and R. Steeb, "Strategies of Cooperation in Distributed Problem Solving," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 767--770, Karlsruhe, Federal Republic of Germany, August 1983. (Also published in [Bond 1988], pp. 102–105).

[Cohen 1990] G. P. Cohen and R. E. Levitt, "The Virtual Design Team: An Object-oriented Model of Information Sharing in Project Design Teams," *ASCE Construction Congress II*, Boston Mass., 1990.

[Conry 1988] S. E. Conry, R. A. Meyer and V. R. Lesser, "Multistage Negotiation in Distributed Planning," in [Bond 1988], pp. 367–384.

[Corkill 1979] D. D. Corkill, "Hierarchical Planning in a Distributed Environment," in *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, Tiblisi, Georgia, USSR, August 1979. (An extended version was published as Technical Report 79-13, Department of Computer and Information Science, University of Massachusetts, Amherst, MA, February 1979.)

[Corkill 1983] D. D. Corkill and V. R. Lesser, "Coordination in a Distributed Problem-Solving Network," in *Proceedings of the Conference on*

Artificial Intelligence, Oakland University, Rochester, MI, April 1983.

[Corkill 1986a] D. D. Corkill, D. D. Corkill, K. Q. Gallagher, and K. E. Murray, "GBB: A Generic Blackboard Development System," *Proceedings of AAAI-86*, Philadelphia, PA, 1986.

[Corkill 1986b] D. D. Corkill, K. Q. Gallagher, and P. M. Johnson, *From Prototype to Product: Evolutionary Development within the Blackboard Paradigm*, Technical Report 86-46, Department of Computer and Information Science, University of Massachusetts, Amherst, MA, 1986.

[Davis 1983] R. Davis and R. G. Smith, "Negotiation as a Metaphor for Distributed Problem Solving," *Artificial Intelligence*, 1983.

[Durfee 1986] E. H. Durfee, *An Approach to Cooperation: Planning and Communication in a Distributed Problem Solving Network*, Technical Report No. 86-09, Department of Computer and Information Science, University of Massachusetts, Amherst, MA, 1986.

[Durfee 1986] E. H. Durfee and V. R. Lesser, "Incremental Planning to Control a Blackboard-Based Problem Solver," in *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, PA, August 1986.

[Durfee 1987a] E. H. Durfee and V. R. Lesser, "Using partial global plans to coordinate distributed problem solvers," in *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987. (Also published [Bond 1988, pp. 285–293.)

[Durfee 1987b] E. H. Durfee, V. R. Lesser, and D. D. Corkill, "Cooperation Through Communication in a Distributed Problem-Solving Network," in M. N. Huhns (Editor), *Distributed Artificial Intelligence, Research Notes in Artificial Intelligence*, Pitman, London, 1987.

[Durfee 1989] E. H. Durfee, V. R. Lesser and D. D. Corkill, "Trends in Cooperative Distributed Problem Solving," *IEEE Transactions on Knowledge and Data Engineering, KDE-1 (1),* 1989.

[Dym 1990a] C. L. Dym, *Representation and Problem Solving: The Foundations of Engineering Design*, Report 05–50–90, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA, 1990. (To appear in *Environment and Planning (B)*).

[Dym 1990b] C. L. Dym and R. E. Levitt, *Knowledge-Based Systems in Engineering*, McGraw-Hill, New York, 1990.

[Dym 1991] C. L. Dym and R. E. Levitt, "Towards the Integration of Knowledge for Engineering Modeling and Computation," *Engineering with Computers*, to appear, 1991.

[Eastman 1975] C. M. Eastman, "The Scope of Computer-Aided Building Design" in C.M. Eastman (Editor), *Spatial Synthesis in Computer-Aided Building Design,* Applied Science Publishers, UK, 1975.

[Erman 1980] L. D. Erman, F. Hayes-Roth, V. R. Lesser and D. R. Reddy, "The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty," *ACM Computing Survey, 12,* 1980.

[Feigenbaum 1977] E. A. Feigenbaum, "The Art of Artificial Intelligence: Themes and Case Studies of Knowledge Engineering," *Proceedings of IJCAI 77,* Cambridge, MA, 1977.

[Fox 1981] M. S. Fox, "An Organizational View of Distributed Systems," *IEEE Transactions on Systems, Man, and Cybernetics, 11* (1), January 1981. (Also published in [Bond 1988], pp. 140–150.)

[Gasser 1989] L. Gasser and M. N. Huhns (Editors), *Distributed Artificial Intelligence*, Volume II, Morgan Kaufmann, San Mateo, CA, 1989.

[Goel 1990] A. Goel, "AAAI '90 Workshop Reports: AI in Manufacturing," *IEEE Expert, 5* (6), December 1990.

[Hayes-Roth 1985] B. Hayes-Roth, "A Blackboard Architecture for Control," *Artificial Intelligence 26,* 1985.

[Hillis 1985] W. D. Hillis, *The Connection Machine,* MIT Press, Cambridge, MA, 1985.

[Howard 1989b] H. C. Howard, R. E. Levitt, B. C. Paulson, J. G. Pohl, and C. B. Tatum, "Computer-Integrated Design and Construction: Reducing Fragmentation in the AEC Industry," *ASCE Journal of Computing in Civil Engineering 3 (9),* 1989.

[Huhns 1987] M. N. Huhns (Editor), *Distributed Artificial Intelligence,* Pitman, London, and Morgan Kaufmann, Los Altos, CA, 1987.

[Jin 1990a] Y. Jin and T. Koyama, "Multiagent Planning Through Expectation Based Negotiation," in *Proceedings of the 10th AAAI International Workshop on Distributed Artificial Intelligence,* November 1990.

[Jin 1990b] Y. Jin, T. Koyama and Z. J. Zhang, "The Marine Traffic Control System As A Distributed Problem Solving Network," in *Proceedings of IEEE 1990 International Conference on Systems, Man, and Cybernetics,* Los Angeles, USA, November 1990.

[Jin 1990c] Y. Jin **on work on ship collision avoidance.**

[Kunz 1989] J. C. Kunz, *Concurrent Knowledge Systems Engineering,* Working Paper No. 005, Center for Integrated Facility Engineering, Stanford University, Stanford, CA, 1989.

[Lesser 1981] V. R. Lesser and D. D. Corkill, "Functionally Accurate, Cooperative Distibuted Systems," *IEEE Transactions on Systems, Man and Cybernetics, 11* (1), 1981.

[Lesser 1985] V. R. Lesser, "Cooperative Distributed Problem Solving," Presentation at IBM—Bethesda, Bethesda, MD, 1985.

[Lesser 1987] V. R. Lesser and D. D. Corkill, "Distributed Problem Solving," in S. C. Shapiro (Editor), *Encyclopedia of Artificial Intelligence, 1,* John Wiley, New York, 1987.

[Levitt 1984] R. E. Levitt, "Superprojects and Superheadaches: Balancing Technical Economies of Scale Against Management Diseconomies of Size and Complexity," *Project Management Journal, 15 (4),* 1984.

[Levitt 1987] R. E. Levitt, and J. C. Kunz, "Using Artificial Intelligence Techniques to Support Project Management," *Artificial Intelligence in Engineering Design, Analysis and Manufacturing, 1* (1), 1987.

[Levitt 1989] R. E. Levitt, I. D. Tommelein, B. Hayes-Roth,and T. Confrey, *SightPlan: A Blackboard Expert System for Constraint Based Spatial Reasoning About Construction Site Layout,* Technical Report No. 020, Center for Integrated Facility Engineering, Stanford University, Stanford, CA, 1989.

[Logcher 1979] R. D. Logcher and R. E. Levitt, "Organization and Control of Engineering Design Firms," *ASCE Engineering Issues, 105,* (EI1), January 1979.

[Maher 1984] M. L. Maher, *HI-RISE: A Knowledge-Based Expert System for the Preliminary Design of High Rise Buildings,* PhD Dissertation, Department of Civil Engineering, Carnegie Mellon University, Pittsburgh, PA, 1984.

[Mayer 1988] A. K. Mayer and S. C-Y. Lu, "An AI-Based Approach for the Integration of Multiple Sources of Knowledge to Aid Engineering Design," *Journal of Mechanisms, Transmission, and Automation in Design 110* (3), 1988.

[Mittal 1986] S. Mittal, C. L. Dym and M. Morjaria, "PRIDE: An Expert System for the Design of Paper Handling Systems," *Computer, 19* (7), 1986.

[Nii 1986] H. P. Nii, "Blackboard Systems: Part I and Part II," *AI Magazine, 7* (2, 3), 1986.

[Pohl 90] J. Pohl and J. Cotton, "ICADS Working Model Version I: A Responsive CAD Environment," *Proceedings of the Symposium on Knowledge-Based Systems in Building Design*, Baden-Baden, West Germany, 1990.

[Riitahuhta 1988] A. Riitahuhta, "Systematic Engineering Design and Use of an Expert System in Boiler Plant Design," *Proceedings of the ICED International Conference on Engineering Design,* Budapest, Hungary, 1988.

[Roos 1967] D. Roos, *ICES System Design,* MIT Press, Cambridge, MA, 1967.

[Rosenschein 1985] J. S. Rosenschein, *Rational Interaction: Cooperation Among Intelligent Agents*, PhD Dissertation, Department of Computer Science, Stanford University, Stanford, CA, 1985.

[SIGMAN 1989] Special Interest Group on Manufacturing (SIGMAN), *Workshop on Concurrent Engineering Design*, Working Notes, International Joint Congress on Artificial Intellegence, Detroit, MI, August 1989.

[Simon 1975] H. A. Simon, "Style in Design," in C.M. Eastman, (Editor), *Spatial Synthesis in Computer-Aided Building Design,* Applied Science Publishers, UK, 1975.

[Simon 1981] H. A. Simon, *The Sciences of the Artificial,* 2nd edition, MIT Press, Cambridge, MA, 1981.

[Smith 1981] R. G. Smith and R. Davis, "Frameworks for Cooperation in Distributed Problem Solving," *IEEE Transactions on Systems, Man, and Cybernetics, SMC-11* (1), January 1981. (Also published in [Bond 1988], pp. 61–70,.)

[Sowa 1984] J. C. Sowa, *Conceptual Structures: Information Processing in Mind and Machine,* Addison-Wesley, Reading, MA, 1984.

[Sridharan 1987] N. S. Sridharan, "Report on the 1986 Workshop on Distributed Artificial Intelligence," *AI Magazine, 8* (3), 1987.

[Sriram 1989] D. Sriram, R. Logcher and S. Fukuda (Editors), *Proceedings of the MIT-JSME Workshop on Cooperative Product Development*, Massachusetts Institute of Technology, Cambridge, MA, November 1989.

[Stankovic 1984] J. A. Stankovic, "A Perspective on Distributed Computer Systems," *IEEE Transactions on Computers, C-33* (12), 1984.

[Stonebraker 1986] M. Stonebraker and L. Rowe, "The Design of POSTGRES," *Proceedings of the ACM SIGMOD Conference*, 1986.

[Talukdar 1987] S. Talukdar, "Design Environments and Software Organizations," in G. Nadler (Editor), *1987 International Congress on Planning and Design Theory: Plenary and Interdisciplinary Lectures,* American Society of Mechanical Engineers, New York, 1987.

[Thompson 1967] J. Thompson, *Organizations in Action*, McGraw-Hill, New York, 1967.

[Thorndyke 1981] P. W. Thorndyke, D. McArthur and S. Cammarata, "Autopilot: A Distributed Planner for Air Fleet Control," in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, August 1981.

[Tommelein 1989a] I. D. Tommelein, *SightPlan: An Expert System that Models and Augments Human Decision-Making for Designing Construction Site Layouts*, Ph.D. Dissertation, Department of Civil Engineering, Stanford University, Stanford, CA, 1989.

[Tommelein 1989b] I. D. Tommelein, *Comparing Design Strategies of Agents with Limited Resources*, M.S. Thesis, Department of Computer Science, Stanford University, Stanford, CA, 1989.