The Knowledge Engineering Review, Vol. 12:4, 1997, 387–406

A knowledge-based architecture for intelligent design support

MING XI TANG

School of Design, The Hong Kong Polytechnic University, Kowloon, Hong Kong

Abstract

The development of a knowledge-based design support system is a lengthy and costly process because various computational techniques necessary for intelligent design support are not readily available in a knowledge-based environment. The systematisation of design knowledge needs combined efforts from designers and knowledge engineers. Existing knowledge-based system development tools offer limited support to intelligent design support which require sophisticated knowledge engineering techniques in terms of knowledge representation, inference, control, truth maintenance and learning. In this paper, a knowledge-based architecture for intelligent design support is described. The existing knowledge-based design system architectures are reviewed first. Five key issues in intelligent design support using knowledge engineering techniques, i.e. design knowledge representation, structure of design knowledge base, intelligent control of design process, consistency and context management of design knowledge, and modelling of design collaboration are then discussed. These discussions provide a basis for a description of a knowledge-based design support system architecture which has been implemented in a Lisp-based environment and tested in two different domains. Current application of this architecture in the development of a design support system in the domain of mechanical engineering design at the Cambridge Engineering Design Centre is presented and evaluated.

1 Introduction

All human workmanship contains some element of design in which knowledge and expertise play an important role. Design is a complex knowledge discovery process in which information and knowledge of diverse sources are processed simultaneously by a team of designers involved in the life phases of a product. Archer stated that design is *a goal-directed, problem solving activity* (Archer, 1970). Feilden (1963) pointed out that "engineering design is the use of scientific principles, technical information and imagination in the definition of a mechanical structure, machine or system to perform pre-specified functions with the maximum economy and efficiency". Asimow (1962) regarded designing as "decision making, in the face of uncertainty, with high penalties for error". Matchett (1968) described the task of designing as ascertaining "the optimal solution to the sum of the true needs of a particular set of circumstances". Andreasen (1991) stated that "design is the common term in industry for a broad range of creative activities such as problem solving, product synthesis, product development and product planning". Simon (1973) claimed that "design is perhaps best characterised as an ill-structured problem to which a solution may not be fully and consistently specified until significant effort to understand the structure of the design problem or the model of the artefact has been made".

The computer is now an established factor in designers' lives. Exploration of a design problem

and finding its solution require knowledge across different disciplines, knowledge which is usually scattered among many experts, publications and perhaps databases. Advanced computer-based design support systems must therefore provide support for the acquisition, manipulation and refinement of design knowledge throughout the life phases of a product. A knowledge-based approach offers more intelligent support to design activities than conventional Computer-Aided Design (CAD) approach (Yoshikawa et al., 1989). The conventional CAD approach offers limited support to the early and crucial phases of the design process, because it is primarily concerned with detailed drawing and the specification of a geometric model of an artefact (Smithers et al., 1990).

A *knowledge-based design support system* is a decision support system to enable designers to explore the structures of design problems and their solutions by combining human design expertise with domain and design knowledge that can be stored in advance in a computer-based design system. Knowledge-based design support systems are different from traditional expert systems which use expert knowledge to help non-experts to solve those problems for which the experts have a good understanding and know how to find their solutions. A knowledge-based design support system aims at combining design theory, AI and computational techniques to support designers through an integration of human intelligence and machine intelligence. As such, they must process a wider range of data and knowledge necessary for solving unfamiliar design problems than conventional CAD systems. The application of AI techniques in the development of knowledge-based design support systems is a typical knowledge engineering problem, the primary purpose of which is to help designers to gather, organise, process and refine design knowledge in a systematic way, and to obtain better design solutions by effectively utilising this design knowledge. The systematisation and utilisation of design knowledge require an AI architecture within which a variety of knowledge engineering techniques can be integrated.

In the last decade, various knowledge-based design system architectures have been developed in a variety of domains (Cater et al., 1991; Clarke et al., 1991; Ball et al., 1992; Sriram et al., 1992; Bowen, 1992; Smithers et al., 1990). However, many fundamental issues concerning the application of knowledge engineering techniques in design are yet to be addressed. In particular, the systematisation of design knowledge and the issue of exploring and maintaining multiple design contexts within a knowledge-based design support system have not been appropriately addressed and tested (Smither et al., 1993; Tang, 1996b). A primary source of difficulty in maintaining multiple contexts of design is to keep track of the many constraints that necessarily exist between the function, physical characteristics, structure, cost and reliability of the various design objects.

This paper presents a knowledge-based design support system architecture as an integrated software kernel for the development of knowledge-based design applications. This architecture provides integrated AI support for the representation of design knowledge, intelligent control of the design process, and the exploration and management of multiple contexts of design. Existing architectures for knowledge-based design are reviewed first. This is followed by a discussion on the evaluation criteria for knowledge-based design support systems. A knowledge-based design support system architecture is then presented. It is based on an integration of several AI-based design methods and several CAD systems that have recently been developed in Edinburgh University and at the Engineering Design Centre of Cambridge University. The key components in this architecture and their integration are described. The implementation of this architecture in a LISP-based environment and its initial application in the development of an Integrated Functional Modelling (IFM) system are presented and evaluated. Finally, some issues concerning the difficulty in integrating knowledge-based system tools involving CLOS, Scheme and 3D solid modelling systems for knowledge-based design applications are discussed.

2 Review of knowledge-based design system architectures

Pugh (1989) suggested that an integrated CAD system where design knowledge is aggregated into sets of independent design knowledge sources, each of which addresses a sub-problem, is

fundamentally sound, and that the way forward is via a system of linked *knowledge modules* to be designed using multi-disciplinary teams.

In the last decade, many knowledge-based design system architectures have used a blackboard control strategy for the control and integration of design knowledge sources to ensure their effective co-operations with a single user (designer). A blackboard system allows design knowledge sources to interact with a user via the blackboard (Hayes-Roth, 1985). A blackboard control system provides a highly structured automatic control scheme, and a special case of opportunistic problem solving. It consists of a number of *knowledge sources*, a blackboard *data structure* and a control system.

In this section, some of the design system architectures utilising a blackboard control strategy are reviewed and their limitations are discussed.

2.1 Review of knowledge-based design system architectures

The Edinburgh Designer System (EDS) is an AI-based design support system developed in an Alvey project that uses a number of AI techniques to support mechanical engineering design (Smithers et al., 1990). In the EDS, design is modelled as an *exploration process*, during which a designer explores the possible ways of constructing a new design using the available *module class definitions* in a *design knowledge base*, or tests different values of design variables within existing module class definitions.

The EDS supports the exploration of multiple design solutions using an Assumption-based Truth Maintenance System (ATMS) (Smithers et al., 1990; de Kleer, 1986). This allows all the knowledge (consistent or inconsistent) relevant to any design exploration attempt by a designer to be maintained. However, the module class definition syntax in the EDS is not suitable for representing complex design objects and design tasks. The EDS II system improved the EDS by developing a view system that can create multiple contexts in the process of design exploration (Logan et al., 1991). But the EDS II system and its view mechanism were not developed as a general design support system architecture, and its facilities were not sufficiently tested in more than one domain. Furthermore, in both the EDS and EDSII systems, a single user is modelled as a *special design knowledge source* which may have priority over other design knowledge sources acting as inference engines. Therefore, design collaboration activities are not explicitly modelled and supported (Smithers et al., 1993; Tang, 1996b).

Carter and MacCallum developed a Hierarchical Object-oriented Blackboard System (HOBS) for supporting electromagnetic design (Carter et al., 1991). In the HOBS architecture, the segments of design expertise or resources are organised into a hierarchy of *knowledge sources* which are controlled through a blackboard control system called the *executive*. Knowledge sources communicate with one another about a *product* through a common data area called the *workspace*.

Hierarchically structured knowledge sources enable the decomposition of a design problem or a task by the knowledge engineer. This hierarchical structure allows normal design management practice such as company procedure to be reflected in the control system. However, a drawback of this is that a hierarchical structure is imposed on the knowledge sources rather than on the design objects. As a consequence, the knowledge sources become less independent and less opportunistic compared with those in a standard blackboard system (Hayes-Roth, 1985). HOBS does not maintain justifications of the design results. It is therefore unable to deal with multiple context problem solving. The workspace in HOBS had a database area, a toolbase area and a message board, but it was not defined and developed as a working place sharable by a team of designers. It therefore cannot be used to support collaborative design activities involving more than one user.

The Distributed Integrated environment for Computer-aided Engineering (DICE) is a design system architecture developed by Sriram et al. (1992) that provides co-operation and co-ordination among multiple designers working in separate engineering disciplines in the domain of construction and building design. The architecture of DICE consists of a *backboard*, a *control system* and a

number of *Knowledge Modules* (KM). The blackboard is divided into three partitions: a *solution blackboard*, a *negotiation blackboard*, and a *co-ordination blackboard*. The solution blackboard contains the design and construction information in the form of object hierarchy generated by various KMs. The negotiation blackboard consists of the negotiation trace between various engineers taking part in the design and construction process. The co-ordination blackboard contains the information needed for the co-ordination of various KMs.

The negotiation process in DICE takes place once a conflict is detected by a strategy KM. In the negotiation process, constraint relaxation is first attempted for those constraints in conflicts. A goal negotiation is then tried when the first attempt fails. The goal negotiation involves the redefinition of a design goal by the design team members concerned in the conflicts. However, the negotiation in DICE largely relies on constraint relaxation techniques and it therefore lacks a formal representation and a mechanism for negotiation in collaborative design. The issue of exploring and maintaining multiple contexts is not addressed in DICE although the necessity for a truth maintenance system is mentioned in Sriram et al. (1992). The negotiation process defined in DICE was based on a scenario where a conflict must be resolved as soon as a single change is made to the data on the solution blackboard. This definition of negotiation is not suitable for collaborative design where designers may access to the system at different locations and at different times.

The Intelligent Front-End (IFE) is an approach that addresses the issue of intelligent human/computer interaction in integrated knowledge based systems (Clarke et al., 1991). An IFE system distinguishes the *front-end* (user end) and the *back-end* (system end), and emphasises the importance of user modelling and effective interaction between the two ends. IFe is a typical Intelligent Front-End system developed at Strathclyde by Clarke for computer-aided building design (Clarke et al., 1991). The IFe system is an integration of several intelligent *clients* within a blackboard system. The IFe consists of a *blackboard* for collecting, organising and storing data, *a dialogue handler* for conversing with the user in the appropriate terminology, a *knowledge handler* for generating the description of a building, a *data handler* for generating the program specific input data, an *appraisal handler* for generating the program-specific control inputs, and *an application handler* for invoking the targeted application programs.

A problem with IFe is that handlers wishing to exchange structured data on the blackboard must all know the details of the imposed data structure. There is no explicit control of user collaboration in the IFe system. The IFe blackboard is merely used as a data storage and communication base, while in many other blackboard schemes this is a case of collecting and scheduling the responses of the knowledge sources to the current situation, and on some occasions the application of a strategy. The IFE approach focused on the relation between a user (the front end) and a system (the back end), but did not address the problem of co-ordinating the collaboration among users.

The Integrated Design Framework (IDF) is a system developed by Ball et al. (1992) to support mechanical engineering design. The IDF architecture has two blackboards: a *control blackboard* and a *domain blackboard*. The IDF has the advantage of modelling the design product as well as the design process within one computational environment by having a *design strategy knowledge source*, a *conflict policy knowledge source* and *a design focus knowledge source*. However, the IDF does not support the exploration and maintenance of design contexts or design alternatives. The knowledge sources in IDF are domain-specific problem solving packages which do not necessarily operate on a unified product data model. It is therefore difficult to utilise these packages for general design problem solving.

The IDF has a process logger knowledge source that records the events on the control blackboard. A prototype induction program is identified within the IDF that can transfer these events to product data models or design planing strategies that can be reused. However, this part of the IDF is yet to be fully developed and tested (Wallace et al., 1995a, b). The IDF has a conflict policy knowledge source which deals with the conflicts resulting from various competing design knowledge sources, but not the conflicts resulted from a collaborative design team.

GALELIO2 is system that explicitly models multiple perspectives and negotiation in collaborative design using a constraint language. In GALELIO2 design variables and constraints can be

viewed from different perspectives created by the people involved in *design*, *manufacturing* and *maintenance*, etc. A perspective is a small set of design variables and constraints. Perspectives are typically overlapping in that the same variable may appear in more than one perspective. The variables within each perspective can be declared as being either *local* or *overwritable*. The local variables within a perspective can only be modified by the person who created the perspective, whilst overwritable variables can be changed from within any other perspectives. Any change to an overwritable variable in any perspective triggers a so-called "negotiation process" during which people concerned are asked to resolve the potential conflicts either by *constraint* relaxation or *goal negotiation* (Bowen et al., 1992). However, in this approach the original data is deleted once some overwritable variables are overwritten and the new values propagated throughout the constraint network. In other words, the context associated with the overwritten variables is not maintained, even though it could still have the potential value as being an alternative design solution. The negotiation process assumes that there is no interdependency between perspectives. As soon as some one changes an overwritable variable and someone else disagrees, the matter must be resolved immediately. This makes it harder for a team of designers to collaborate on a design project over a period of time. For true concurrency and collaboration it must be possible for two or more activities occur *simultaneously* (Medland, 1995).

2.2 Current positions

Four main strategies may characterise the current approaches in developing knowledge-based design systems (Wallace et al., 1995a): the intelligent CAD approach; the building block approach; the prototype approach; and the constraint-based approach. Intelligent CAD aims at extending the capability of a CAD system by employing heuristic knowledge on top of geometric models of the artefact (MaCallum, 1990). The building block approach decomposes design process into tasks that can be tackled by different classes of CAD tools and AI methods (Brown et al., 1989; Mostow et al., 1989). The prototype design approach divides design into three different activities: *prototype refinement, prototype adaptation* and *prototype creation* using a library of pre-defined design prototypes (Gero et al., 1989). Constraint-based design formulates design problems and requirements as interconnected networks in which design variables/parameters are related to each other through constraints. The networked design variables and constraints can then be manipulated to find the solutions that satisfy all the requirements (Bowen et al., 1992; Smithers et al., 1990; Tang, 1996a).

However, none of these approaches explicitly addressed the issue of providing a general knowledge-based design support architecture for intelligent design support from a knowledge engineering perspective. The original aim of AI techniques for intelligent design was to produce general purpose, domain independent design tools that would automate design (non trivial) tasks requiring intelligence. The early promise of AI systems has not been fulfilled because these tools did not *scale* within and across domains, they could not *adapt* to new contexts, failed to handle *complexity* adequately, could not *acquire knowledge* satisfactorily, and placed too much emphasis on automation at the expense of assistance. (Wallace et al., 1995b). These problems are resulted from a lack of understanding of design as an intelligent behaviour, poor or inappropriate use of knowledge representations, loosely integrated knowledge-based architectures that were difficult to maintain and extend, and a lack of commitment to design knowledge acquisition and design knowledge systematisation.

2.3 Evaluation criteria for knowledge-based design support systems

The role of a knowledge-based design support system is to extend geometric-based representation and reasoning to knowledge-based representation and inference in order to provide wholesome solutions to a wide range of design problems (Smithers et al., 1993). The competence of a

knowledge-based design support system architecture needs to be evaluated based on the following criteria:

- It should help designers to construct and extend the design knowledge base within which domain concepts, design objects, dependency information of design objects are well structured and consistently maintained.
- It should help designers to derive solutions quickly from initial, not necessarily complete and consistent design requirements; in other words, it should provide an efficient mechanism to transform an initial design requirement description to a design specification.
- It should provide explicit explanations and justifications for any chosen aspects of the current status of a design, not only in terms of how something has been derived, but also in terms of why something is not happening as expected. Locating areas of difficulty and suggesting strategies for solutions contribute to effective decision making in design.
- It should allow designers to vary data, design requirements, problem solving strategies or evaluation criteria, to obtain alternative design solutions. Simply speaking, a knowledge-based design support system must support multiple-context problem solving so that a design problem can be explored from many different perspectives by a team of designers working in a co-operative manner.
- It should provide mechanisms for capturing and refining design knowledge so that the design knowledge base can be incrementally enlarged and enhanced. In other words, it should have some learning facilities to support conceptual design tasks and design knowledge acquisition.
- It should have easy access to visualisation tools, solid modelling tools and other analytic systems.

3 Knowledge-based design support

Knowledge engineering techniques provide knowledge representation methods that suitably represent and manipulate different types of design objects, heuristic design knowledge and design constraints; they also provide inferencing support and control mechanisms for designers to perform design tasks, including analysis, synthesis, evaluation and optimisation; they also provide mechanisms for detecting and resolving conflicts among various design requirements and design constraints. This section discusses several knowledge engineering issues concerning the development of a knowledge-based design support system architecture that can be evaluated in the above criteria.

3.1 Design knowledge representation

Design of any kind is a knowledge intensive activity. A systematisation of design knowledge is essential for utilising general design problem solving methodologies and domain-specific design knowledge throughout the design process. In design, the ability of a group of designers to identify the design problem and its solutions is based on well organised and exchangeable design knowledge. It is useful to classify the knowledge necessary for design support into three categories: (1) static knowledge representing design objects and concepts (domain knowledge) that is stored in a design knowledge base; (2) heuristic and inferential knowledge representing problem solving strategies and methods (design knowledge) that can be invoked by different types of user as design knowledge sources for a range of design tasks; and (3) dynamic knowledge (new knowledge) generated during design when applying (2) to (1) as a result of carrying out a design task.

Static knowledge consists of common concepts and objects in the domain of a design application. These concepts and objects can be used to build *product data models* (Yoshikawa et al., 1989). A product data model contains descriptive product information in terms of specifications, function, attributes, behaviour, documentation, geometry, history, context, variables and constraints, etc. Figure 1 illustrates such a product data model at an abstract level in the domain of mechanical engineering design. In this product data model a product has an assembly, an interface, a context and a history. An assembly consists of parts each of which has sub-parts. A subpart is itself an

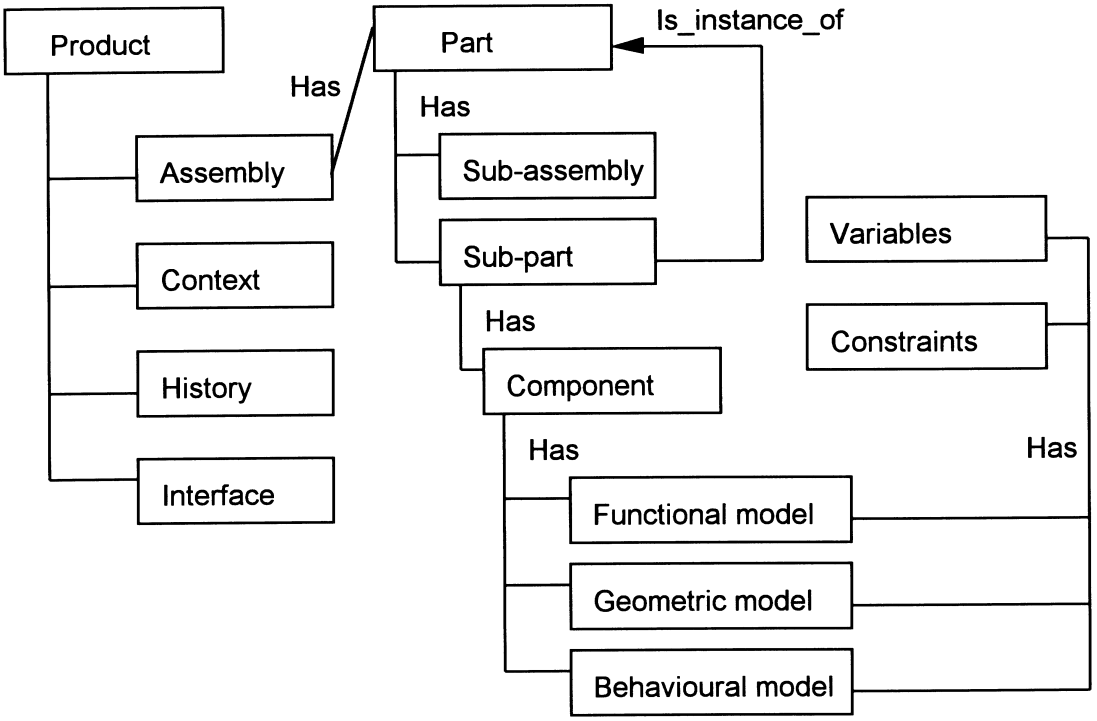

Figure 1 Class definition of a product data model

instances of another part. This recursive use of part and subpart allows the development of uniform inferencing programs capable of reasoning on different levels of a complex product data model.

A sub-part has components, each of which has in turn functional, geometric and behavioural models. Shaft, lever, tie-rod and screw, etc. are, for example, all sub-classes of a component. These sub-classes inherit general attributes associated with a component and also have their own specific attributes. For example, the function of a shaft is to transform an input torque to an output torque. The shaft has variables such as speed s, diameter d, length l, bearing force Fb, torque T, power P and mass m. Well defined constraints such as the relation $P = sT$ and $m = \rho d^2 \pi l/4$ can be associated with the shaft in advance. At a symbolic level, functional, geometric and behavioural models all have variables and constraints of different types. It is important to establish a mapping between these models and their symbolic representation. At a symbolic level, design variables and constraints can be easily manipulated by AI-based programs, whilst at functional, geometric and behavioural levels, knowledge can be more effectively associated with domain dependent contexts for the purpose of analysis and visualisation.

Heuristic and inferential knowledge is mainly for exploring the solutions of design problems. It is knowledge about the design process and design problem solving. This kind of knowledge includes mainly design standards, guidelines, prototypes or documents of previous designs, heuristics and management strategies. It also includes generic knowledge such as mathematics, chemistry and physics that can be used in different stages of the design process. In a knowledge-based design support system, this kind of knowledge is used to manipulate static knowledge to generate the knowledge of a new design. This knowledge is inferential and it needs to be constantly re-organised as *design knowledge sources* to perform various design tasks. A *design knowledge source* is an encapsulation of inferencing methods, rules or programs representing an autonomous piece of design expertise for a *specific design task* such as kinematic analysis, or a *general design task* such as satisfying a set of functional or geometric constraints imposed on a set of design variables.

A design knowledge source is self-contained and independent because it defines inferencing methods for a class of design objects, as well as the conditions that must be satisfied before these methods can be invoked. It needs to be opportunistic and adaptive in the sense that it can invoke a rule, a procedure or an external software system to perform a design task according to the levels or

versions of the product data model specified by the designers. The action of a design knowledge source needs to be justified by the satisfactory matching of its preconditions against the available data in the system. These preconditions can be recorded together with the results generated so that when a design session is concluded, it is possible to trace the results that have been derived by the system.

3.2 The structure of a design knowledge base

There are four levels upon which domain and design knowledge can be structured in a design knowledge base using an object-oriented approach:

1. an element (or component) level,
2. a task-independent level,
3. a task dependent level, and
4. an interface level.

At the lowest level, i.e. element level, domain concepts and static information in terms of design components and parts are represented as object classes. These object classes are self-contained and they encapsulate reasoning methods to determine their behaviour in normal situations. The elements at this level may be linked to external material or CAD databases. The knowledge at this level provides basic building blocks for the construction of more complex knowledge structures.

At the second level, i.e. task-independent level, there are objects which connect those elementary building blocks at the element level through their structural, functional, and causal relationships (Tang, 1996b). A structural relationship between design objects determines how elements are geometrically or physically related to each other and how, for example, the characteristics of a power train can be derived from its constituent components such as piston, crank pin, crank web, crank shaft and bearing, etc.; a functional relationship between elements relates them in terms of their performances or behaviour such as input/output transmission, or their relevance to a general design requirement and a design task; a causal relationship between two elements decides how they depend upon each other and what the consequences are of a change in either of them (Iwasaki et al., 1986). The knowledge at this level provides common sense and qualitative relations with which the building blocks at the element level can be associated with each other meaningfully. The knowledge at this level is task independent because it is mainly used for the system to perform general design support tasks such as: assembling design components, deriving design variable/parameter values, predicting the consequences of design modifications, searching for a set of design variable values that satisfies a set of design constraints, etc.

At the task dependent level (or design process level), there are specific product or process modules, with which the elements and relations at the element and task-independent levels can be dynamically constructed and controlled to form special process-based design models for solving new design problems. The knowledge at this level provides embodied process-based or task-oriented modules for planning, scheduling and controlling specific design tasks. Therefore, the knowledge at this level is particularly domain dependent.

At the highest level is the interface level that provides straightforward answers to design enquiries from different types of user. It also provides graphical explanation and interface management facilities for the designers to see what knowledge is available in the system, what the current status of the design problem is, and what solutions can be derived by the system, giving a functional requirement of a new design. At this level, the user is treated as a special source of knowledge. That is, a user may dynamically re-organise the knowledge in the system for solving specific design problems or for adding new knowledge into the system.

3.3 Intelligent control of design process

Knowledge-based design support systems are often developed for working in an environment

where: groups of designers work co-operatively to complete a complex and frequently large design; close interaction exists between group members through shared design data and information; interdependent design knowledge sources or programs exist and need to be controlled and co-ordinated; use of databases, solid modelling and CAD tools is commonplace; and management of multiple design contexts is necessary for a group of designers. The control and integration of design knowledge sources to ensure their effective co-operations with human designers are the main tasks of an intelligent control system (or a design manager) in a knowledge-based design support system architecture.

A blackboard is a flexible control system that allows a design support system's design knowledge sources to interact with users' inputs via the blackboard (Hayes-Roth, 1985). The blackboard control strategy contributes to design support tasks in the following aspects:

- it allows a knowledge-based design support system to work as a self-organising system whose problem solving design knowledge sources can respond dynamically to new design situations;
- it allows the development of independent and self-contained design knowledge sources, thus making it easy to integrate and modify design knowledge; and
- it is suitable for different knowledge representation schemes, i.e. anything that can be treated as a black box such as a rule set, a procedure, or an external software package can be regarded as a design knowledge source.

Given the integrated and complex nature of knowledge-based design support systems, the integration and control of a large number of diverse design knowledge sources are important. The blackboard control strategy has been widely adopted in knowledge-based design support system architectures, especially integrated system architectures as reported in Smithers et al. (1990), Carter et al. (1991), Clarke et al. (1991), Sriram et al. (1992) and Ball et al. (1992). However, none of these systems has been developed as a generic knowledge-based design support system architecture. Furthermore, few systems have developed a mechanism for consistency maintenance and context management within a blackboard control system. In DICE, for example, the role of truth maintenance in its architecture is identified as providing justifications for a knowledge module. But the use of an assumption-based truth maintenance in the exploration and maintenance of multiple contexts of design is not explored in DICE (Sriram et al., 1992). The suitable integration of a truth maintenance system with a blackboard control strategy allows the development of a design context management system to support the exploration of multiple design contexts by a design team.

3.4 Truth maintenance and design context management

In design, multiple design solutions arise in the presence of under-constrained design variables and parameters. Multiple design solutions are derived from a space in which *design requirements*, *design methods* and *design criteria* are subject to frequent change. This means that the exploration of multiple design solutions is context dependent. The management of design contexts in a knowledge-based design support system is concerned with the following issues:

- maintaining the consistency of design knowledge in the system;
- representing newly derived design objects, their dependants and antecedents; and
- supporting the exploration of multiple design solutions based on different design decisions made by the designers.

This section discusses computer-based exploration and maintenance of multiple contexts of design and describes the role of a *design context management system* based on an integration of a blackboard control strategy and an ATMS.

3.4.1 Modelling context in design

One of the frequently carried out tasks in design is to make plausible modifications to part of an initial design solution to observe the repercussions to find alternative design solutions. The design

decision making process is typically *nonmonotonic*, *constructive* and *incremental* in that even experienced designers cannot guarantee to get it right first time. More often, designers need to make changes when information about the consequence of earlier decisions becomes available, and the design solution space has become more and more constrained. Most design tasks are likely to create new design objects and concepts rather than simply modifying existing designs. This view is supported by noting that:

- design activities are often undertaken simultaneously by a group of designers;
- designers explore the design solution space rather than systematically search in it;
- it is important for incompatibility in design requirements, constraints and evaluation criteria to be discovered early during the design process;
- designers need to know the consequences and implications of any change either in input data, design method or design evaluation criteria; and
- designers need explicit explanations of the design results derived from any design decisions, or the reasons for any difficulties in finding a satisfactory design solution.

These issues can be dealt with in a knowledge-based design support system using a consistency maintenance and design context management system, the central role of which is to maintain the consistency of the knowledge generated during design and to support the exploration of this knowledge when design context changes.

In engineering design, a product data model (or structure of a design problem) is often defined by design objects and their relations. Design objects contain attributes which can be classified as *design variables* and *dependent design parameters*. The values of design variables and dependent design parameters are determined by the constraints in which these variables and parameters are functionally, structurally or causally related. Design variables and dependent design parameters are used in order to distinguish the part of a design problem that is flexible to change (described by the design variables) from other parts of the design problem (described by the dependent design parameters) that are relatively dependent on the design variables (Tang, 1996b).

A design solution is a complete set of values for all the design variables and dependent design parameters which satisfies all the constraints. The space of potential design solutions is determined by the constraints between all the design variables and dependent design parameters. The constraints are typically combinations of mathematical equations, rules, and qualitative reasoning modules. When designers explore this space of potential design solutions, many plausible choices may arise in the presence of under-constrained design variables, giving rise in turn to many plausible values of dependent design parameters. Furthermore, the way in which design variables and dependent design parameters are related and explored may depend on what design strategies (or methods) are employed by the designers. In other words, using a different design problem solving strategy (or method) may result in the same set of design variables and dependent design parameters being constrained differently.

Design exploration involves exploring the convergence of all the design variables using the most appropriate design methods (or design knowledge sources) available in the system. The exploration of a design solution in a knowledge-based design support system is usually carried out by the designer making decisions (or assumptions) by way of selecting or constructing a product data model containing design variables and dependent design parameters using the available building blocks in a design knowledge base, assuming the values of these design variables, or selecting design knowledge sources that manipulate the design variables and the constraints. The system processes these decisions in a systematic way by invoking the necessary design knowledge sources that propagate the assumed values of design variables throughout the constraint network associated with the product data model.

A design context is a design solution (or a partial design solution) that is derived as a result of the system's inferencing from the designer's choices of initial data, design method (or design procedure) and design evaluation criteria. Because a designer's assumption is part of a design context, different

designers working on sets of overlapping design variables can explore design solutions within their own design contexts simultaneously.

3.4.2 Design context management using an ATMS

The co-existence of potentially conflicting assumptions and their derivatives in a computer-based design system presents a problem for maintaining the truth of the knowledge and for modelling design context within a knowledge-based design support system. The role of a design context management system is therefore:

- to maintain the design results and their justifications generated during the design process;
- to provide easy access to, and a good explanation of, the existing knowledge in the system;
- to make the best use of the knowledge already held in the system to generate new knowledge without performing redundant inferences; and
- to help designers compare different, sometimes conflicting design solutions.

A number of truth maintenance techniques have been developed in AI to deal with the problem of consistency and context management. While some of the truth maintenance systems are concerned only with maintaining the justification of derived knowledge, Assumption-based Truth Maintenance System (ATMS), based upon de Kleer's work, offers sufficient facilities for working with inconsistent information, and for multiple context problem solving (de Kleer, 1986). An ATMS is preferable to other truth maintenance systems when different design knowledge sources, including users, hold distinct perspectives over the same problems. An ATMS is particularly suitable for design exploration because of its incremental updating of a dependency network and its ability to maintain a multi-contextual environment to allow different design solutions to be explored simultaneously.

The application of ATMS in design has been reported in Smithers et al. (1990, 1993), Logan et al. (1991), Sriram et al. (1992) and Banares-Alcantara (1991). But none of these systems has formulated design context in such a way that a design context management system can be developed based on a blackboard control strategy to support the exploration of multiple design contexts as part of a general knowledge-based design support system architecture (Tang, 1996a).

A blackboard control system works mostly in an opportunistic way. The integration of an ATMS with a blackboard control system provides a unique facility for knowledge-based design applications by building a multi-contextual justification network to maintain the consistency of the derived knowledge, and to support the exploration of multiple design solutions. In such a so-called *truth maintained blackboard system* (Logan et al., 1991), the ATMS builds a justification network in the form of *assumption nodes* representing decisions or assumptions made by the designers, and *derived nodes* representing the results inferred by the design knowledge sources in the system. The ATMS maintains the dynamic growth of this justification network by updating the information associated with each node whenever a piece of new information is derived by the design knowledge sources. This ensures that all the newly derived knowledge is justified in the context of basic design decisions (user assumptions), and maintained throughout a design session.

The design context management system based on an integration of a blackboard control strategy and an ATMS (Tang, 1996b; Ross, 1989) performs the following operations: control the creation of design contexts for individual designers, support the exploration of the design problem in these contexts using design knowledge sources, sort out alternative design solutions from all the design contexts, maintain the justification of the knowledge generated within individual design context, provide explanations of any chosen aspect of the system status, and control the negotiation process and conflict resolution.

3.5 Modelling of design collaboration

A knowledge-based design support system needs to model the close co-operation between the designers and the system by providing explicit explanations of the system's behaviour, and by

providing ways for designers to intervene in the design process at various stages of design exploration. Designers' expertise, intuition and understanding of the design problems play an important role in this co-operation.

A design difficulty may arise when an expected design solution cannot be derived by the system given the available information in the system, or when a designer's choice of some design variable values results in some of the constraints set by other designers being violated. In these situations the designers need to consult each other to find out the source of the difficulty and to negotiate to find a solution. However, few of the existing knowledge-based design system architectures explicitly address the issue of providing design collaboration management facilities for a team of designers. The reasons why these architectures cannot support design collaboration include:

- No mechanism has been developed to model design collaboration between a team of designers. In almost all of the reviewed system architectures in section 2, a single designer is modelled in the design process which may be treated in some architectures as a special high priority knowledge source. The focus of research has been on how to deal with the problem of one user interacting with a number of design knowledge sources, rather than how different users interact with each other to solve a design problem.
- No system had a design process model with which necessary design collaboration operations may be defined. Design collaboration takes place throughout the design process, not just on some isolated specific design tasks. There is a need to define what a design collaboration process is in a knowledge-based design support system, especially within the context of a networked association across the world wide web.
- No system provided good facilities for documenting the design information as well as the decision process that has led to the specification of this information. There has been no explicit modelling of a design collaboration history that could be used to explain what has been done, or what are being done by the others involved.

The widespread use of world wide web on the Internet makes it more urgent to solve these problems concerning user modelling and HCI. User modelling in knowledge-based design is concerned with data input, inferencing control and result explanation. A knowledge-based design support system must be able to answer user questions such as *What has been derived?*, *How is something derived?* and *What could be done next?,* etc. In design collaboration, an additional issue needs to be addressed. That is, the issue of knowledge-based communication and negotiation.

Negotiation during a design project can take place in two ways:

1. In the design exploration process, during which the designers consult with each other whenever they need to change the value of a design variable set by others.
2. After the design exploration process, at which time a design context management system sorts out all the alternative design solutions from the multiple design context network maintained by the system, and then ranks them based on the preferences of most designers.

It is believed by the author that the first approach may create unnecessary communications and delay the design project, whilst the second approach is more suitable for collaborative design as it supports concurrency and maintains multiple design solutions. The selection of a final design solution ultimately depends on a collective decision that has to be made after both approaches are exhausted. But by identifying the areas where there are still conflicts in the multiple design context network, the system is well placed to explain the source of difficulty and help the designers to reach a final design solution.

In a knowledge-based design support system, repeating a particular design session or restoring an interrupted design session is particularly useful for design collaboration, especially when a design project lasts for a long time. The traceability of a long term design project can be maintained by repeating any previous design session using recorded design history files. In this way, the knowledge of an experienced designer can be monitored and recorded, allowing design knowledge to be

accumulated and shared by the others. A design history file is best stored in such a format that it can be replayed (Mostow, 1989).

4 Implementation of the architecture

The mechanism for consistency maintenance of design knowledge was first developed in Edinburgh in the EDS system as part of the Alevy large scale demonstrator Design to Product (Smithers et al., 1990, 1993) using an Assumption based Truth Maintenance System (ATMS) (de Kleer, 1986; Ross, 1989). It was extended in the EDSII project by adding a view mechanism that allows the creation of multiple views of design defined by designers (Logan et al., 1991). It was further formalised by the author as a design context management system within a lisp-based knowledge-based design support architecture (Tang, 1996a, b). This Lisp-based architecture as an integrated AI tool consists of components or sub-systems of a software system for developing knowledge-based design applications. It consists mainly of a *design knowledge base*, a *design concept learning system*, an *assumption-based truth maintained blackboard control system*, a *design context management system*, a *design documentation system*, and a *graphical user interface* (Tang, 1996b).

A Knowledge Based System (KBS) development tool called GoldWorks III^{TM} was used to implement this architecture. GoldWorks III is a Lisp-based KBS tool whose facilities are based on different types of objects, namely, frame, instance, assertion, relation, attempt, sponsor, agenda item, message passing handlers and daemon, etc., with which reasoning and control programs can be developed (Tang, 1995). Rule- and frame-based knowledge representations, forward, backward, goal-directed forward chaining, dynamic object-oriented graphics are general functions available within GoldWorks III.

The implemented architecture on top of GoldWorks III is intended to provide a Lisp-based environment with enhanced design support functions, including mainly:

- management of a design knowledge base,
- control of design knowledge sources,
- creation and maintenance of multiple design contexts,
- documentation of design history, and
- graphical explanation of design results.

The integration of a blackboard control strategy and an ATMS forms the core of this architecture (Ross, 1989; Tang, 1996b). This integration is done by creating a special Knowledge Source Activation Record (KSAR) within the blackboard control circle. When a KSAR is proposed by any design knowledge source, it carries the necessary information for the ATMS to establish the link between consequence and precondition of a proposed inference action. This link is then passed to the ATMS for building the justification and context for any inferred results.

Figure 2 illustrates this control process that is not available as a generic design support function in any other knowledge-based system development tools.

- A KSAR is firstly checked for its format by the system. If the format is right, and the new KSAR is not a duplicate of any KSARs already in the agenda, it is placed onto the blackboard agenda. Otherwise the KSAR is ignored.
- Before a KSAR is executed, it is checked again to see whether its preconditions still hold, i.e. to see whether the antecedents of the KSAR remain consistent. The system does this by checking whether the KSAR has a valid set of ATMS nodes and assumptions as the justification for the proposed action.
- When a KSAR has been executed, the generated result is placed on the blackboard. The system creates a new ATMS node for the result and adds its justification to the ATMS database.

The main difference between a standard blackboard control system and a truth maintained blackboard control system is that in the latter system, the inference is based on justified

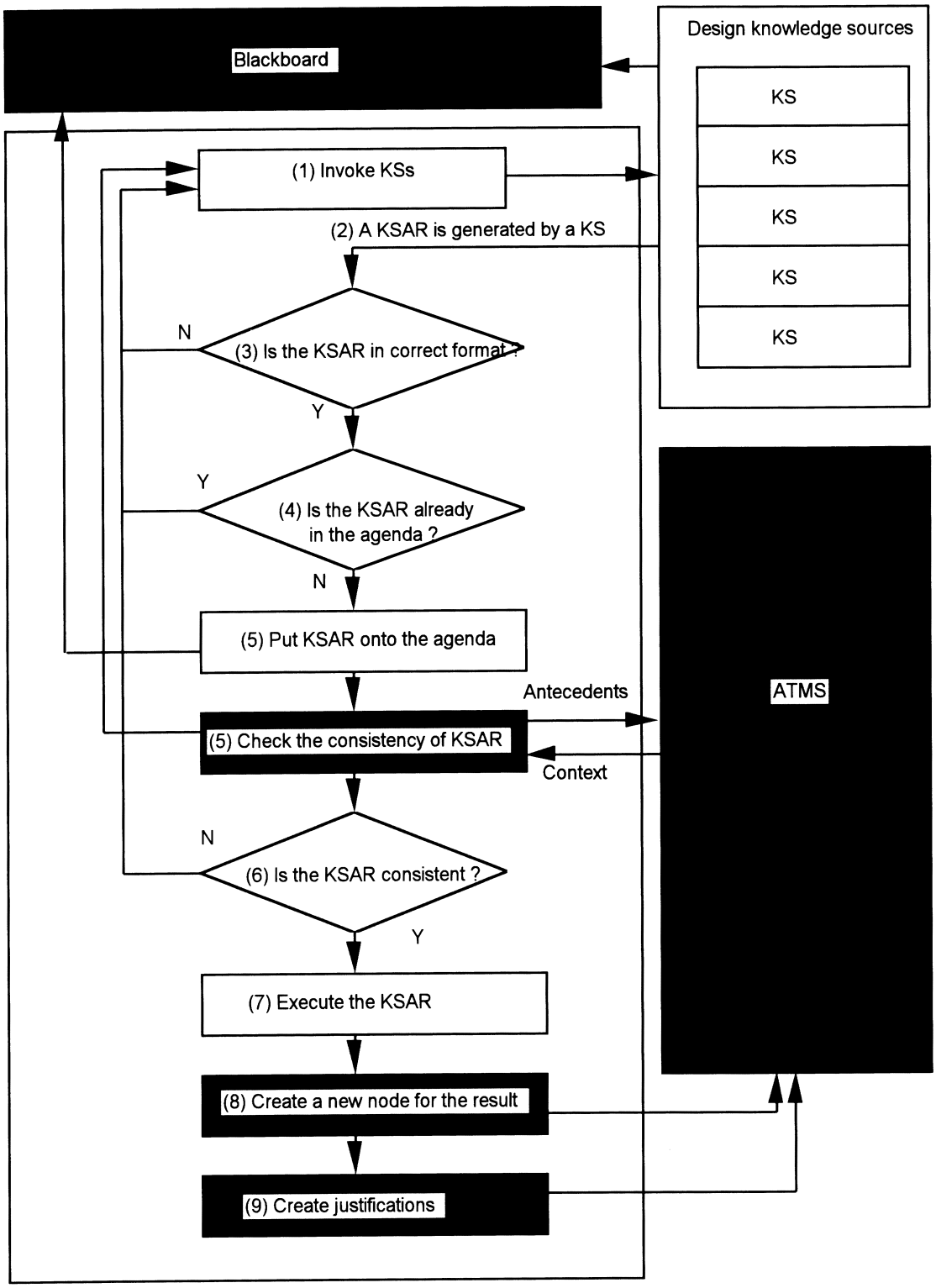

Figure 2 Truth maintained blackboard control process

preconditions, and these preconditions are used as the antecedents to build up the network of justifications from which multiple design contexts can be established using the ATMS. In this way, the ATMS *safeguards* the blackboard as well as effectively creating multiple contexts for the data on the blackboard. The detailed specification and integration of this architecture and its application in the domain of small molecule drug design can be found in Tang (1996b).

5 Applications

The architecture described in this paper has been successfully applied to the domain of small molecule drug design in the Castlemaine project (Smithers et al., 1990, 1993). However, to be used as a general tool for knowledge-based design applications, it needs to be tested in more than one domain. The recent development of an Integrated Functional Modelling (IFM) system by the author at the Cambridge Engineering Design Centre (EDC) is an attempt to further enhance the capability of this architecture. The IFM is an integration of this architecture with other systems that have been developed in the Cambridge EDC recently to build an intelligent design tool for mechanical engineering design (Wallace et al., 1995a, b; Tang, 1996a, b).

5.1 IFM

The IFM combines functional, symbolic and geometric knowledge to support functional synthesis of design concepts, embodiment generation of product design and dynamic simulation of product behaviour within a knowledge-based environment. The IFM system also aims at establishing a knowledge-based framework that enables design process management, design product specification and design knowledge capturing in the domain of mechanical engineering design.

The development of IFM is focused on three areas of knowledge engineering techniques: systematisation of engineering design knowledge at the early stage of the design process; development of several general support systems for computer-based engineering design activities; and integrated application of AI techniques including inductive learning techniques in engineering design (Tang, 1996a). Figure 3 illustrates a demonstrator of the IFM developed using the architecture described in section 4.

In the IFM demonstrator, the design knowledge base contains a library of functional components and parts, a design object hierarchy representing the relationships between these functional components and parts, a set of constraints on the assembly of these functional components and parts, and a database containing detailed information about these functional components and parts such as material, geometry, etc.

The IFM system supports engineering design tasks in an incremental way. The functional components can be selected and synthesised first using a functional synthesis system to form a set of solution concepts (Chakrabarti et al., 1994). The solution concepts are then clustered using an inductive learning system to allow easy browsing and selection by the designers. A selected solution concept forms a so-called conceptual product data model. A conceptual product data model contains abstract information in terms of functional components and their topological arrangement to the satisfaction of a stated functional input/output requirement. A conceptual product data model can then be transferred into a constraint-based product data model containing design variables, dependent design parameters and constraints required for embodiment design, kinematic analysis and dynamic simulation. This constraint-based product data model can be manipulated by any member of a design team using a symbolic constraint manager until a satisfactory embodiment solution is found.

The constraints associated with a complex product data model need to be partitioned in a number of ways so that AI methods can be used more easily to satisfy them. This partition is largely based on the structural relationships of the product data model such as parts, components and assemblies, or based on the algebraic relationships of the design variables. Any partition identifies a small region in the design space which might be usefully searched for partial solutions using either simulated annealing or genetic algorithms, both of which treat a design problem as a *goal-directed search*, the goal being to minimise the number of constraints that are violated (Thornton, 1993).

The use of symbolic computation techniques such as constraint propagation, constraint simplification and symbolic equation solving (for removing equality constraints) ensures that this search space is well identified and confined before any automatic search methods such as genetic

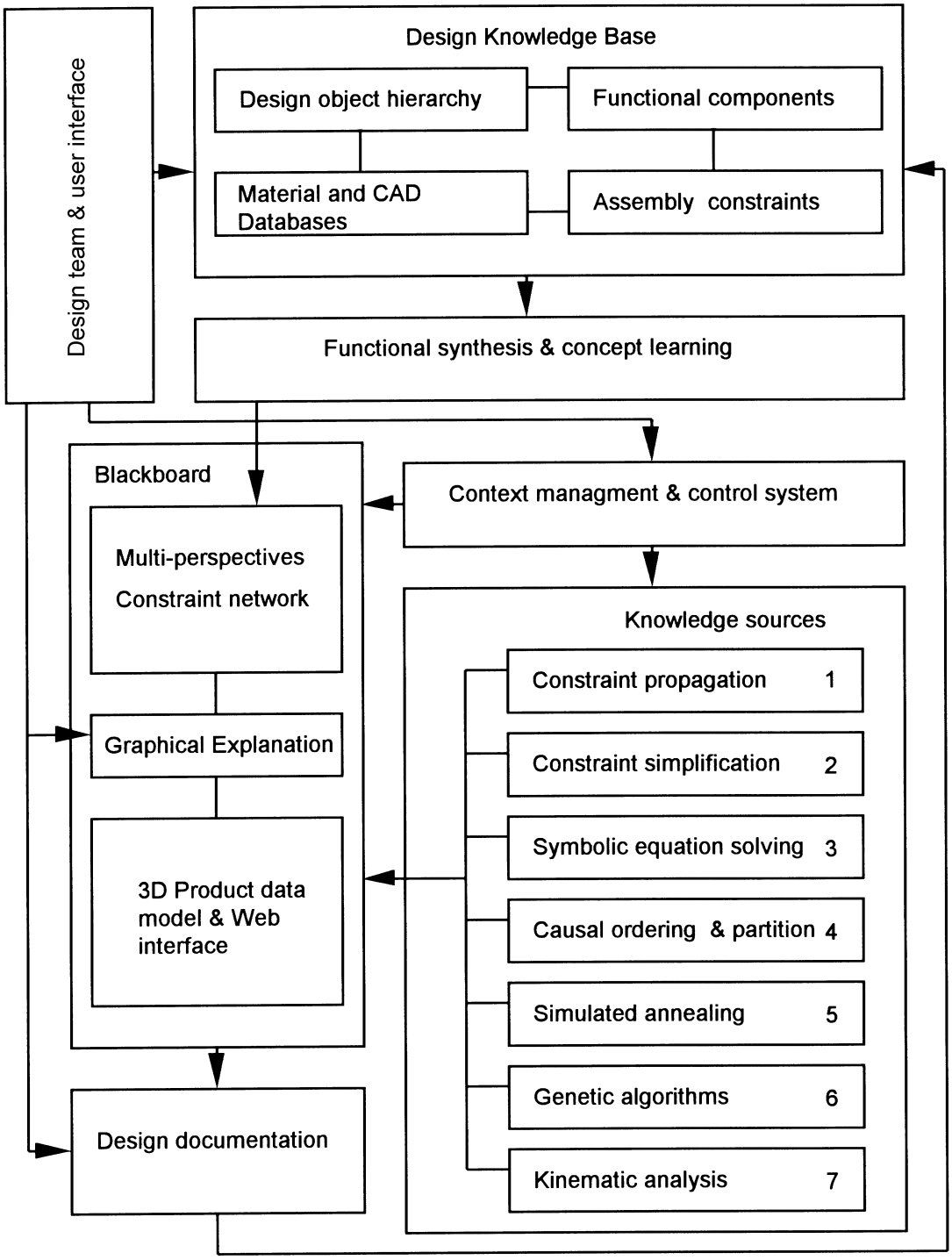

Figure 3 Integrated functional modelling system

algorithms and simulated annealing are used. Therefore, a symbolic constraint manager can be seen as a system integrating heuristic-based AI methods with automatic methods for design optimisation.

Collectively, the designers, the functional synthesis system, the inductive learning system and the symbolic constraint manager allow the embodiment generation of an artefact from a functional requirement statement using a selected set of basic functional components. The result generated provides sufficient information for 3D kinematic analysis (Johnson, 1988; Johnson et al., 1993).

In exploring the constraint network, many alternative solutions may be created by different members of a design team. These contexts are maintained in the system throughout the design

process, which is easily accessible through a graphical user interface by any member of a design team. The graphical explanation system is also used to provide visual links between the 3D model of an artefact and its underlying constraint network. That is, what is done by the computer system at a symbolic level is actually visualised in a 3D model. A design documentation system records the evolving product data model in terms of its specifications and the design history that has led to these specifications. The recorded design history can be reviewed and replayed by any member of a design team after the design session has concluded (Tang, 1996a). Part of the implemented IFM demonstrator is being evaluated by experienced designers within the Cambridge EDC (Chakrabarti et al., 1994).

5.2 Integration issue

The development of the IFM system is motivated by the need to systematise engineering design knowledge at a functional level using the best available knowledge engineering techniques, and to form a prototype of a new generation of CAD systems that makes extensive use of this systematised knowledge (Chakrabarti et al., 1996). The implementation of the initial IFM demonstrator involves complex software integration in a Lisp-based environment. It involves the integration of a Lisp-based functional synthesiser system called FuncSION by Chakrabarti et al. (1996) with ACIS 3D toolkit™; the integration of a genetic algorithm for constraint optimisation developed by Thornton (1993); the development of a Lisp-based constraint manager and a design context management system.

A difficulty arises from the fact that none of the currently available knowledge-based system development tools is able to support sophisticated 3D graphics. However, 3D modelling is an essential part of any computer-based design system even in the stage of conceptual design. Most knowledge-based system development tools such as KEE™, ProKappa™, ArtEnterprise™, GoldWorks III™, KnowledgeWorks™, LispWorks™, etc. only have limited facilities for developing 2D graphics. To reduce the complexity of software integration, the IFM is now being further developed using CLOS and ACIS 3D toolkit via a Scheme interface (Martin, 1995). The feasibility of such a platform for the commercialisation of the complete version of the IFM in the near future is being evaluated. Other approaches involving the integration of other commercially available tools such as ICAD, ProEngineer for the further development of IFM system are also currently being explored.

6 Evaluations and discussion

The integration of a blackboard control system and an assumption-based truth maintenance system in the architecture described in this paper provides an effective mechanism for the exploration and management of multiple contexts of design, which is absent from many of the design systems such as HOBS, IFe, DICE and IDF, etc. (Carter et al., 1991; Clarke et al., 1991; Sriram et al., 1992; Ball et al., 1992). This mechanism is suitable for intelligent design support because the design decision process is typically *nonmonotonic*, i.e. some later decisions may contradict the earlier ones.

The following features are uniquely associated with the architecture presented in this paper:

- It combines rule-based and object-oriented knowledge representation for the development of design knowledge bases and design knowledge sources.
- It has been designed and implemented as a self-contained AI software kernel with basic and essential components, including an inductive conceptual learning system, for intelligent design support.
- It provides flexibility for the designer to explore multiple contexts of design by making different design decisions (even conflicting decisions). A design context can be created by any member of a

design team by making assumptions. The design context management system maintains the justifications of any derivations based on these assumptions.

- It has the potential for support concurrent engineering design and web-based design. The knowledge on the blackboard can be shared by members of a design team and explored from different perspectives. Negotiation can be delayed until such a time when it is necessary to evaluate a number of alternative design contexts (or solutions). The negotiation decisions can also be added to the system as *assumptions*, resulting in some of the undesirable design results becoming eliminated.
- It provides facilities for documenting design results and design history that can be replayed. Any derived knowledge can be graphically explained using the information kept in the design document.
- It provides a good basis for web-based systematisation of engineering design knowledge and for supporting web-based design collaboration.

The architecture described in this paper has been tested in the domain of small molecule drug design. It has proved to be effective in this domain, and it supported the task of identifying alternative design solutions (pharmacophore descriptions) (Smithers et al., 1990, 1993). However, a number of limitations have been observed:

In the current implementation of the design context management system, a design context is defined based on a constraint-based approach to design. That is, a design context is defined on a domain independent basis, i.e. on the basis of the *design data*, the *design method (constraints)*, and the *design variable values*. The difficulty arises from the fact that some issues, such as the question of what forms the context of a design, and how the designers explore design contexts, are not necessarily domain independent. A more general design context definition should take into account a wider range of issues such as the *design process*, the *design product data model*, the *manufacturing constraints*, the *materials*, and the *user type*, etc. The definition of a design context in the current design context management system needs to be extended in order to support design applications where multiple solutions need to be explored simultaneously by a team of designers, each of which may have different priorities and concerns;

The architecture described in this paper does not provide an explicit partition of the blackboard like other design systems such as HOBS, IFe, DICE and IDF (Carter et al., 1991; Clarke et al., 1991; Sriram et al., 1992; Ball et al., 1992). The current implementation of the architecture only supports an implicit partition of the information available on the blackboard by means of creating design contexts. It is possible to create more than one blackboard within the current implementation of the architecture because the blackboard itself is an object class. However, the issue concerning the co-operation between different blackboard instances has not been addressed. This may appear to be over restrictive for systems in which different software packages must operate on different data structures (Tang, 1996b).

In the current implementation of the architecture, all user actions are treated as *assumptions* upon which the system's inferences depend. It is therefore unable to deal with different types of users, each of which may be concerned with only one part of an integrated system and may have to carry out domain specific design tasks using the data created by others involved in the design process. The inability of the current architecture to model different types of user restricts its application in concurrent engineering design, especially in web-based design where it is customary for different types of users to work collaboratively to solve a complex design problem (Tang, 1996b).

7 Conclusions

In conclusion, an integrated application of knowledge engineering techniques is important for the successful development of a knowledge-based design support system. A knowledge-based architecture provides a computational platform for the integration of knowledge engineering techniques. The architecture presented in this paper is for providing support for knowledge-based design system

development, and it is based on an integration of a blackboard control system and an assumption-based truth maintenance system. Apart from providing general support for the acquisition of design concepts, design knowledge representation, intelligent control of design process and design documentation, this architecture provides a unique mechanism for the exploration and maintenance of multiple design contexts. It is thus particularly suitable for design collaboration and design knowledge systematisation. This architecture has been implemented in a Lisp-based environment and successfully tested in the domain of small-molecule drug design and the domain of mechanical engineering design. The limitations of the current architecture have been identified. These limitations are being addressed further in the current development of an Integrated Functional Modelling system in the domain of mechanical engineering design, and in the development of a design collaboration management system using the World Wide Web (WWW) as an enabling technique.

Acknowledgements

My previous research on knowledge-based design system was supervised in the Department of Artificial Intelligence in the University of Edinburgh by Dr Peter Ross and Dr Tim Smithers (now with the University of the Basque Country in Spain). The research presented in this paper is a continuation of the research carried out earlier by Smithers and his AI in design group in Edinburgh University. The implemented architecture reported in this paper integrates a C-based ATMS implemented by Dr. Peter Ross of Edinburgh University's AI department. The development of the IFM is part of an ongoing research theme (Functional Modelling) at the Engineering Design Centre (EDC) of Cambridge University. The EDC is funded by the EPSRC in the UK. I would like to thank Dr Nigel Ball, Dr Amaresh Chakrabarti of Cambridge EDC, and Aylmer Johnson of Cambridge University Engineering Department (CUED) for their support on the software implementation of the IFM system. The current demonstrator of the IFM integrates a software called FunCSION developed by Dr Amaresh Chakrabarti. I would like to thank Ken Wallace, John Clarkson of CUED, Professor Roy Farmer, Dr Lucienne Blessing, Dr Stuart Burgess and Dr Tim Murdoch of Cambridge EDC for their support in this research.

References

Andreasen M, 1991. "Design methodology" *Journal of Engineering Design* **2**(4).

Archer LB, 1970. "An overview of the structure of design process" in Moore, GT (Ed), *Emerging Methods in Environmental Design and Planning*, Cambridge, MA: MIT Press, pp. 285–307.

Asimow M, 1962. *Introduction to Design*, New York: Prentice-Hall.

Ball N and Fauert F, 1992. "The integrated design framework: Supporting the design process using a blackboard Ssystem" *Proceedings of 3rd International Conference on Artificial Intelligence in Design*.

Banares-Alcantara R, 1991. "Representing the engineering design process: Two hypotheses" *1st International Conference on Artificial Intelligence in Design* Edinburgh, Scotland.

Bowen J and Bahler D, 1992. "Supporting multiple perspectives: a constraint-based approach to concurrent engineering" *AI in Design'92*.

Brown DC and Chandrasekaran B, 1989. *Design Problem Solving: Knowledge Structures and Control Strategies*, London: Pitman.

Carter I and MacCallum K, 1991. "A software for design co-ordination" *1st International Conference on Artificial Intelligence in Design* Edinburgh, Scotland.

Chakrabarti A et al., 1994. "A two-step approach to conceptual design of mechanical device" *AI in Design'94*.

Chakrabarti A and Tang M, 1996. "Generating conceptual solutions on FuncSION: Evolution of a functional synthesiser" *Artificial Intelligence in Design Conference (AID96)* Stanford, CA.

Clarke J and Randal D, 1991. "An intelligent front-end for computer-aided building design" *Artificial Intelligence in Engineering* **6**(1).

de Kleer J, 1986. "An Assumption-based TMS" *Artificial Intelligence* **28**.

Hayes-Roth B, 1985. "Blackboard architecture for control" *Artificial Intelligence* **26** 251–231.

Fielden GBR, 1963. "The Fielden report" *Engineering Design*, London: HMSO.

Gero JS and Roseman MA. "A conceptual framework for knowledge-based design research at Sydney

University's Design Computing Unit'' in Gero JS (Ed), 1989. *Artificial Intelligence in Design* London: Springer-Verlag.
Iwasaki Y and Simon H, 1986. ''Causality in device behaviour'' *Artificial Intelligence* **29** 3–32.
Johnson AL and Chen F, 1993. ''Modelling functionality in CAD: implications for product representation'' *Proceedings of the 9th International Conference on Engineering Design*.
Johnson A, 1988. ''Functional modelling: A new development in computer-aided design'' *Proceedings of IFIP WG5. 3 workshop on Intelligent CAD*.
Logan B, Millington K and Smithers T, 1991. ''Be economical with the truth, assumption-based context management in the Edinburgh Designer System'' *Artificial Intelligence in Design Conference* Edinburgh, Scotland.
MacCallum KJ, 1990. ''Does intelligent CAD exist? '' *Artificial Intelligence in Engineering* **5**(2).
Matchett E, 1968. ''Control of thought in creative work'' in Gregory S (Ed), *The Design Method* London: Butterworths.
Mostow J, 1989. ''Design by derivational analogy: Issues in the automated replay of design plans'' in Carbonell JG (Ed), *Machine Learning, Paradigms and Methods* Cambridge, MA: MIT Press.
Martin E, 1995. ''Getting started with ACIS 3D Toolkit'' Schemers Inc.
Medland AJ, 1995. ''Managing design and manufacturing constraints in a distributed industrial environment: the creation of a managed environment for engineering design'' *Lancaster International Workshop on Engineering Design*.
Pugh S, 1989. ''Knowledge-based systems in the design activity'' *Design Studies* **4**(10).
Ross P, 1989. ''A simple ATMS'' Department of Artificial Intelligence, University of Edinburgh, UK.
Simon HA, 1973. ''The structure of ill-structured problems'' *Artificial Intelligence* **4** 181–201.
Smithers T, Conkie A, Doheny J, Logan B, Millington K and Tang M, 1990. ''Design as intelligent behaviour: An AI in design research programme'' *Artificial Intelligence in Engineering* **5**.
Smithers T, Tang M and Tomes N, 1993. ''Approach in intelligent drug design'' *26th Hawaii International Conference on System Science* Hawaii, USA.
Sriram D, Logcher R, Groleau N and Cherneff J, 1992. ''DICE: An object-oriented programming environment for cooperative engineering design'' in Tong C (Ed), *Artificial Intelligence in Engineering Design* Academic Press.
Tang M, 1995. ''Development of an integrated AI system for conceptual design'' *Lancaster International Workshop on Engineering Design*.
Tang M, 1996a. ''An AI-based architecture for concurrency management in engineering design'' *ASME Design for Manufacturability Conference* Chicago, IL.
Tang M, 1996b. ''Knowledge-based design support and inductive learning'' PhD Thesis, Department of Artificial Intelligence, University of Edinburgh.
Thornton A, 1993. ''Constraint specification and satisfaction in embodiment design'' PhD Thesis, University of Cambridge, Department of Engineering.
Wallace K, Ball N and Tang M, 1995a. ''AI in engineering design'' *Fourth Workshop on Research Directions for Artificial Intelligence in Design* Enschede, The Netherlands.
Wallace K, Ball N and Tang M, 1995b. ''Object-oriented technology and engineering design'' *Workshop on Object-Oriented Technology* Strathclyde.
Yoshikawa M, Arbab F and Tomiyama T, 1989. ''Intelligent CAD III'' *IFIP WG5. 2 Workshop on CAD* Osaka, Japan.