Gero JS and Brazier FMT (eds) (2002) *Agents in Design 2002*
Key Centre of Design Computing and Cognition, University of Sydney, pp. 233-246.

A KNOWLEDGE BASED COLLABORATIVE DESIGN ENVIRONMENT

HONG LIU, MINGXI TANG and JOHN HAMILTON FRAZER
Design Technology Research Center
The Hong Kong Polytechnic University Hong Kong China

Abstract. In this paper, we propose an agent based collaborative design environment in which human designers and software agents interact with each other, exchange design information and keep track of state information to assist with collaborative design. First of all, it presents a hierarchical multi-agent system architecture for integrating design and engineering tools, software agents and human specialists in an open environment. The hierarchical multi-agent system architecture offers a promising framework with their novel approaches for dynamically creating and managing design tasks in widely distributed and ever-changing design environments. Secondly, it introduces a collaborative design process model and the dynamic management approach for collaborative design process. Then, the structure of a design agent, its static knowledge and dynamic knowledge are introduced respectively. The knowledge based design approach provides a foundation for supporting reusable design activities. Finally, the cooperative design process is illustrated by a bicycle design example.

1. Introduction

Design is increasingly becoming a collaborative task among designers or design teams that are physically, geographically, and temporally distributed. The complexity of modern products means that a single designer can no longer complete design task. Design is a team effort in which groups of designers with different intent, background knowledge work together. Close operation among them will accelerate the product development by shortening the development cycle, improving the product quality and reducing investment. Global collaboration is the key to ensure the competition in product design and development.

Designers are no longer merely exchanging geometric data, but more general knowledge about design and design process, including specifications, design rules, constraints, etc. In addition to sharing and exchanging information, pressure to reduce product development times has resulted in an increased focus on methods for representing and storing engineering artifact knowledge in a way that facilitates its retrieval and subsequent reuse. As design becomes increasingly knowledge intensive and collaborative, the need for collaborative design environment to support the representation and use of knowledge among distributed designers becomes more critical.

Current design practice frequently does not provide enough inter-participant interactions to maintain the coherence of the design team. The complexity and alterability of design practice demand a dynamic organizational structure for design teams. By using a computer-aided design environment which provides collaborative mechanisms, design team can maintain the distributed nature of engineering design and, at the same time, obtain the evolutionary nature of dynamic changed environment.

This paper introduces a multi-agent collaborative design environment. The aim is to provide a collaborative platform for supporting designers in teams each with different intent, background knowledge, area of expertise and responsibility.

The remainder of this paper is organized as followings. Section 2 presents a hierarchical multi-agent system architecture. Section 3 introduces dynamic collaboration among agents. Section 4 is the structure of a design agent and the knowledge of a design agent. Section 5 shows a bicycle design example for illuminating the collaborative design process in this environment Section 6 summarizes the paper and gives an outlook for the future work.

2. Hierachical Multi-Agent System Architecture

While there is no unified agreement about the definition and capabilities of an agent, a software agent is believed to be a software component capable of 1) perceiving and acting at a certain level, 2) communicating in some fashion with other agents, 3) attempting to achieve particular goals or perform particular tasks, and 4) maintaining an implicit or explicit model of its own state and the state of its world (Barber and Kim 2000).

The general architecture of a multi-agent collaborative design environment is organized as a population of asynchronous semi-autonomous agents for integrating design and engineering tools and human specialists in an open environment, Figure 1. Each tool (or interface for human specialist) can be encapsulated as an agent. These tools and human specialists are connected by a local network and communicated via this network. Each can also communicate directly with other agents located in the other local networks by the Internet. The agents exchange design data and knowledge via a local network or the Internet via the management agent.

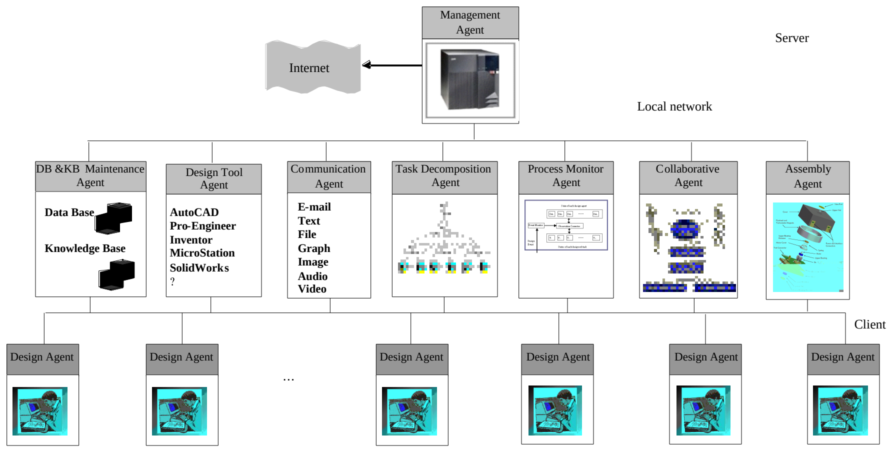

Figure 1. The architecture of a multi-agent collaborative design environment

All agents in this environment form an agent group. There are three classes of agents: management agent, tool agents and design agents. These agents are situated on the different layers (Liu and Zeng 1997). The hierarchical relation limits the authority of the agents in the group.

· Management agent locates on the server and manages the whole design group. The action of management agent usually shows the decision and inquiry for the problem, control and supervision for lower layer agents. The knowledge in the KB of a management agent includes all design agent's name, address, and skills or competencies, the history records of performing task and the reward in the group. When an agent is added to or deleted from the group, the corresponding knowledge of management agent will be modified.

· Tool agents include design tools, and management tools. They help management agent to complete system management tasks, such as communication management, task decomposition, database management, knowledge management, collaboration management and system maintenance.

Task decomposition agent help design engineer to decompose a large design task into several sub-tasks.

Collaborative agent matches the sub- tasks and suitable design agents. It also deals with conflict coordination during collaborative design process.

Design tool agents include AutoCAD, Pro-Engineer, Inventor, MicroStation, SolidWorks and so on. It also includes Video Conferencing system for synchronous collaborative design providing run-time support.

Communication agent provides support for interaction among agents and designers by E-mail, text, file, image, graph, audio and video. The exchange of data and files is based on the file transfer protocol (FTP) and TCP/IP protocol.

Process monitor agent watches the whole design process via its event monitor and dynamically maintains the information about the state of each design agent and the status of current design sub-tasks. Whenever a design event (such as submission, modification and so on) happened, the event monitor will be triggered and the correlative message will be passed suitable agents.

Assemble agent checks assembly constraints for finished design components. When constraint violation is found, it will ask collaborative agent and communication agent to solve problem by coordination among design agents.

Knowledge maintenance agent and database agent maintain knowledge base and database respectively.

Design agents are a kind of domain-dependency agent. They have special design knowledge and ability and can help designers in a special domain.

The various design agents would accomplish the same goal but in different manners. Because, in design, there is no single or clear-cut answer, different design agents working on the same problem can generate completely different solutions. By having agents with different abilities contributing to designs, the process gains robustness and variety in solving various conceptual design problems.

The creation of complex design in this environment is due to collaboration among several different agents. These agents contain knowledge of how to design based on their individual strategies and preferences. They are constructed to understand the representation of a design state, be it complete or incomplete, and contribute in a manner that leads to successful solutions. The strategies used by these agents are based on deterministic algorithms, such as genetic algorithm, classifier algorithm and so on. In the current implementation, agents are not autonomous, but are triggered by the system or by other agents.

When a large design task comes, task composition agent help design engineer to decompose it into several subtasks and send the sub-tasks to collaborative agent. The collaborative agent matches the sub-tasks and design agents according their ability. After a dynamic task assistant process, design agents and designers perform their own design tasks respectively. During design process, communication agent takes charge of interaction among agents by E-mail, text, file, image, graph, audio and video passing. The KB & DB agents maintain knowledge and database. Process monitor agent watches the whole design process. When constraint violation is found by assembly agent, it will inform collaborative agent and communication agent to solve problem by coordination among design agents. When a design phase is over, the design result will be evaluated by experts. Then, design process will be finished or restarted according to experts' decision.

3. Dynamic Management Among Multi-Agent Design Environment

One of the things that makes multi-agent environment so attractive is that we can change or reorganize the task and agent set in response to new technologies or unanticipated requirements. However, this flexibility makes it impossible to eliminate conflict through knowledge engineering; we cannot engine an agent at design or implementation time to be in agreement with all other potential future agents (which may not even be imagined yet). Therefore, dynamic management is an inherent requirement in this environment.

Following we will introduce a task-oriented collaborative design process model to describe certain phenomena in which the design tasks are undertaken to possibly reach the final design. The model is important for all participants to understand his/her position in design collaboration, and for researchers to analyze design activities.

3.1. *THE TASK-ORIENTED COLLABORATIVE DESIGN PROCESS MODEL*

Definition 3.1: DA_s denotes a design agent, in which, D means the type of an agent and s is a character string that represents which group the agent belongs and its serial number in the group. For example, DA_{11} is a design agent with number 1 in the group 1.

Definition 3.2: T_S stands for a design task, c is a character string that represents the decomposed layer of the design task and the dependency relation. For example, an initial design task can be represented as T_1, its subtasks are T_{11},T_{12}, ,T_{1n} and the sub-processes of T_{1i} are T_{1i1},T_{1i2}, ,T_{1im} separately, i.e. the length of string denotes the decomposed depth while the value expresses the dependency relation i.

The dependency relation of design tasks forms a design task tree (see section 4 for product data model).

Definition 3.3: T_i^j denotes the task i is being done by the design agent j.

We can know the group members who is performing the task T_i by vector $(T_i^{j1}, T_i^{j2}, \ldots, T_i^{jk})$ and the current tasks of the design agent j by vector $(T_{i1}^{j}, T_{i2}^{j}, \ldots, T_{il}^{j})$.

Definition 3.4: The prior relation of design task is indicated by pair PRIOR (T_{s1}, T_{s2}), which means that the T_{s2} takes the fulfillment of T_{s1} as the starting pre-condition; T_{s1} and T_{s2} are the sequences of tasks respectively.

Definition 3.5: The concurrent relation CONCUR (T_i, T_j) expresses the design tasks T_i and T_j can be carried out simultaneously.

Definition 3.6: The exclusive relation EXCLUDE (T_i, T_j) expresses the two tasks T_i and T_j can't be performed simultaneously.

Definition 3.7: The event is expressed by denotation E (i).

3.2 PROCESS MONITOR AGENT

The task-oriented problem solving relation is a kind of dynamic organized relation that is formed when agents complete tasks for a common design goal. The relation among agents is dynamically changed. As soon as the tasks are fulfilled, the relation is dissolved voluntarily. When new task comes, the new problem solving relation may be formed by a group of new agents.

This dynamic task and agent set and their relation are watched and recorded by process monitor agent. Whenever a design event (such as submission, modification and so on) happened, the event monitor of process monitor agent will be triggered and the correlative message will be passed to observation corrector for renewing the knowledge of the process monitor agent. The information by process monitor agent maintained will be passed to task decomposed agent if a design task has been finished or a task need to be re-decomposed. This information is also be used by collaborative agent to deal with conflict.

3.3 THE DYNAMIC MANAGEMENT OF COLLABORATIVE DESIGN PROCESS

Construction of a complex design task is accomplished by several design agents. In this process, communication plays an important role. In general, communication can be synchronous or asynchronous, and the communication mode can be point-to-point (between two agents), broadcast (one to all agents), or multicast (to a selected group of agents). The environment takes KQML (The Knowledge Query and Manipulation Language) as communication language for the interaction among agents.

During collaborative design process, design agent sends REQUEST, SUBMIT or MODIFY with corresponding message to CA (Collaborative Agent) according to its situation. CA passes the information to correlative agents by NOTIFY, PUBLISH and mediates conflict according to design task correlations.

For example, one design agent, say DA_1 , will perform task T_i and need all of the messages about T_i . DA_1 will send REQUEST message to CA for each of these. By checking design task correlations, from PRIOR(T_i, T_j), CA knows that the prior condition of staring task T_i is the finish of task T_j , i.e. T

$_j$ has been done by one agent, say agent DA_2. A REQUEST message will be forwarded to DA_2 by the CA. Then, DA_2 will REPLY CA and pass related information to DA_1. It depends on the content of REQUEST by CA. If CA requires DA_2 send related information to it first, the information will be forward to DA_1 by CA later.

When DA_2 modifies the task T_j, it will send a MODIFY message to CA. Then CA checks all corresponding tasks and agents, such as DA_1, the CA will send MODIFY message to DA_1 together with the modified set from DA_2. Any agent who has task input related with T_j will get this notice for modifying the corresponding design, all previous outputs of the correlative agents will be considered for update.

The CA informs agents not only when a task is completed, but also when some unexpected events happened. For example, a given subtask becomes superfluous because the super-task has been discarded. Whenever a subtask no longer has any valid justification, the owner of the subtask should be notified. We can find all of them by design task tree and correlations among agents and tasks.

4. Design Agent

The majority of agents in the design environment are design agents. A design agent is a computer software that in some way helps users to complete design tasks. It is a designer’s assistant and can adapt its own ability via interaction with designers and the other agents.

4.1 THE STRUCTURE OF A DESIGN AGENT

A design agent includes adaptive interfaces that guide a user in performing design tasks and some software components that can transfer input into output according to the design goal.

A design agent can be defined by a seven-tuples (Aid, Input, Communication, Transformer, Output, Goal, Trigger).

In which, Aid is the identifier of a design agent;

Iuput is the input interface component. It gets input information (such as design specifications, design constrains) and passes them to design goal and transformer separately. It also gets some information coming from communication component and passes them after transform;

Communication is the communication component. It receives message from other agents or systems and passes them to input interface;

Transformer consists of Knowledge Base, Learning Engine and a knowledge-based transform component, which transform the input into output.

Output is the output interface component;

Goal is the design goal component and consists of some constraints. It gets goal information from other agents or designers. The goals of a design activity determine the type of design activity performed by the design agent.

Trigger is the design activity triggering component. It consists of Event-Condition-Action Rules.

An agent gets the information from the environment (by the user, by communication or by the feedback for improving design) and then translates it to the internal description of the situation. This description is divided into three parts: one part is used to improve the design goal of the agent, another part is passed to transfer component as the design requirement and the third part is the situation for knowledge update.

4.2 THE DESIGN KNOWLEDGE

In a multi-agent environment, design tasks are divided into subtasks, and the knowledge for solving individual subtasks is stored in a knowledge base as independent knowledge sources. Knowledge is regrouped into several knowledge modules in the knowledge base.

There are two major knowledge categories: (1) static knowledge representing design objects and feathers; (2) Dynamic knowledge representing problem solving strategies and methods (Tang 1996).

Static knowledge can be expressed by product data model. The information contained in the product data model can be thought of as divided into component layer and feature layer. The component layer is concerned with the general specifications of the components (including sub-components or parts) and the relationships among components, while the feature layer contains information regarding individual primitive features. Form features are the main building blocks of the components and act as the communication medium between the design process and decision support procedures. The geometrical and technological requirements by the designer for each feature are used as input information.

Dynamic knowledge is mainly for exploring the solutions of design problems. It is knowledge about the design process, design strategies and design problem solving. In a knowledge based design support system, this kind of knowledge is used to manipulate static knowledge to generate the knowledge for reuse in a new design.

Dynamic knowledge is stored as goals and schemes in the knowledge base. Design agent regards them as the control knowledge for the current design session. The definition of goals, schemes, working space and relative concept are as following.

Definition 4.1: A *design goal* can be expressed by a twain (G, D). In which, G is goal name, D is design expression including all restrictions and technical criteria.

Definition 4.2: A *design goal tree* (called GoalTree for short) is a tree which takes total design goal as the root and layered design sub-goals as sub-nodes.

Definition 4.3: A *design scheme tree* (called SchemeTree for short) is a 'AND/OR' tree which takes one design goal as the root and the realizable schemes of the goal as sub-nodes. "AND" relation denotes a parent scheme composed by some children schemes while 'OR' relation denotes the different schemes for realizing the same parent design goal.

Definition 4.4: A *design scheme space* (called SchemeSpace for short) is a set of all design schemes corresponding to one goal tree.

Definition 4.5: A *design working space* is a quaternion (SavepointTree, WorkingPath, CurrentWorkingScheme, Status), in which:

(1) *SavepointTree* is triad (S_0,S,B) that represents one multi-forked tree, in which: S_0 SchemeSpace is the root of the tree; S SchemeSpace is the node set of the SavepointTree; B is the set of binary (S_i , S_j), (S_i , S_j) denotes a path from parent node S_i to sub-node S_j , S_i, S_j S.

(2) *WorkingPath* is a sequence (S_{i0},S_{i1},S_{i2} , , S_{in}), $S_{i\ j}$ (j=1,2,…,n) SchemeSpace.

(3) *CurrentWorkingScheme* is an active scheme on WorkingPath currently.

(4) The value domain of *Status* is the set {ACTIVE, ACHIEVE, INACTIVE, PAUSE}.

ACTIVE, ACHIEVE, PAUSE express active, achieve, inactive and pause status of a design activity respectively.

The *SavingPointTree* is used in this representation for backtracking. The main idea is that when a design agent thinks the current design status is valuable for saving (it is possible to backtrack to here), one saving point will be set up here. The main body of design working space is a tree that is composed by design status (saving points). When mending begins, design agent can start from a saving point for saving time.

The design process starting from the nearest saving point forms the current working path. The current working path is a linear sequence of statuses, in which the transfer between two statuses is caused by an object operation. Design agent can search for a useful design save point and do Undo or Redo operation on the current working path.

Definition 4.6: *Design state transfer operation* Transfer changes one design state named Design.state1 that satisfies Condition to another design state named Design.state2.

Transfer (Design.state1$\rightarrow$ Design.state2) While Condition

Definition 4.7: Design sate transfer operation space is the set of all design state transfer operations. A design state transfer operation is composed of OPName and OP. In which:

OPName={transaction.begin, transaction.commit, transaction.abort, savepoint, backtrack, undo, redo, pause, resume }

OP={opi(scheme1 $\rightarrow$ scheme2), opi   OPName, scheme1, scheme2   SchemeSpace, i=1,2,…,9}.

Definition 4.8: Design affair is a sequence of design state transfer operations. It starts from transaction.begin and ends at transaction.commit or transaction.abort.

Design state transfer operations savepoint and backtrack support the saving and backtrack in design process, undo, redo present the part testing actions on current path, and pause, resume embody that the one segment design can span many courses.

4.3 KNOWLEDGE UPDATE

After a design session is finished, the newly solved design is stored in the knowledge base for future reuse. Knowledge base management agent records design knowledge in several forms, including problem inputs, final solution, intermediate solutions, design history, and design strategies. The knowledge base update process is as follows:

· Identify major design operations. The design process is analyzed. Then the system filters out unnecessary design steps that led to unsuccessful alternatives or that do not directly contribute to the solution process.

· Create goals. A goal is created for each identified major operations in order to prefer the same type of action in the future.

· Modify design schemes by comparing the difference between design and redesign goals. The major design operations can be classified into design and redesign operations. Redesign operations are those that are executed when a constraint violation is present in output result. Design operations are those that lead directly to the eventual solution. The goals corresponding to design operation are grouped into design schemes, while the redesign goals are grouped into redesign schemes corresponding to each backtracking episode resulting from constraint violations. Redesign schemes will be used to modify design schemes by pruning backtracking paths. These actions on the backtracking path will not be executed in the similar design next time.

· Record the critical constraints. The constraints that were violated and caused backtracking are recorded so that they can be considered early in future design.

5. A Collaborative Design Example

In this section, we will introduce a bicycle design example for showing the collaborative process in our multi-agent collaborative design environment. Another example can be seen in (Liu, Tang and Frazer 2001a).

Step 1. When a product design task (such as bicycle design) is introduced, the design engineer attempts to find a suitable design product class which matches the current design task. In this example, he/she will select a bicycle from a list. If there is a bicycle on the list, the corresponding product data model tree will be shown on the screen (Figure 2).

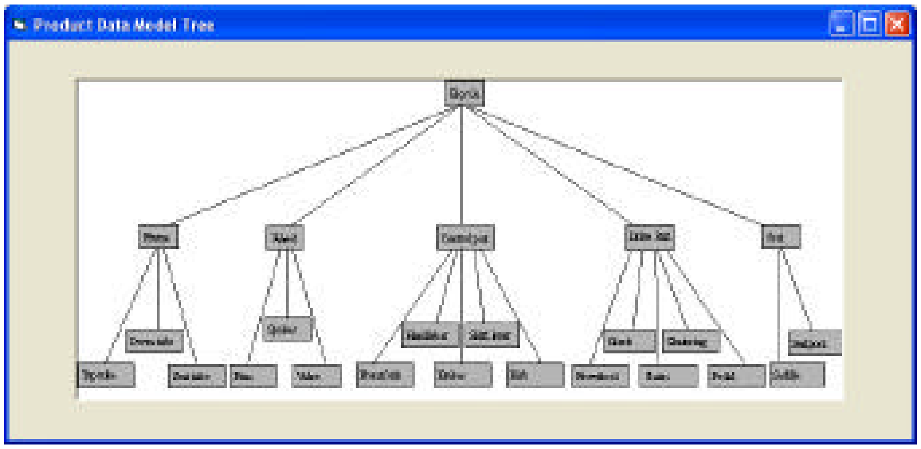

Figure 2. A bicycle product data model tree

If the design engineer can not find a suitable match for the current design task (for instance, there is no bicycle in the list), he/she can create a new class and product data model tree by selecting **Create** on the menu and following the guide. He/she should decompose the bicycle first, Figure 3. Then, he/she should answer some questions step by step, such as how many components a product can be divided into and the name of every component. Then the question is how many sub-components are included in one component and so on. The design engineer can select **Back** to correct any mistakes at every step. He/she can also make modification after the guide process has ended.

Step 2. In Step1, the design task is decomposed and forms a product assemble model tree. The design engineer can add or delete components from the product data model tree until it accurately matches the current design task. The collaborative agent matches the sub-tasks and design agents according their ability. After a dynamic task assistant process, the component design tasks and corresponding constraints are assigned to several designers and design agents.

Step 3. When a designer receives a design task, he/she inputs the design requirement according to the task and constraints. Then, design agent attempts to find a suitable component, which satisfies the requirements. If a component is found, its sketch and corresponding attributes will be shown on the display screen, Figure 4. The designer must then decide whether or not to accept the component. If the component is accepted then the designer will, after possibly having made certain adjustments and alterations, submit it to the assemble agent. On the other hand, if the component is not accepted then the designer will create a new component with the help of design agent. Whenever the old design is perfected or a new design is finished, the design knowledge of design agent will be updated automatically.

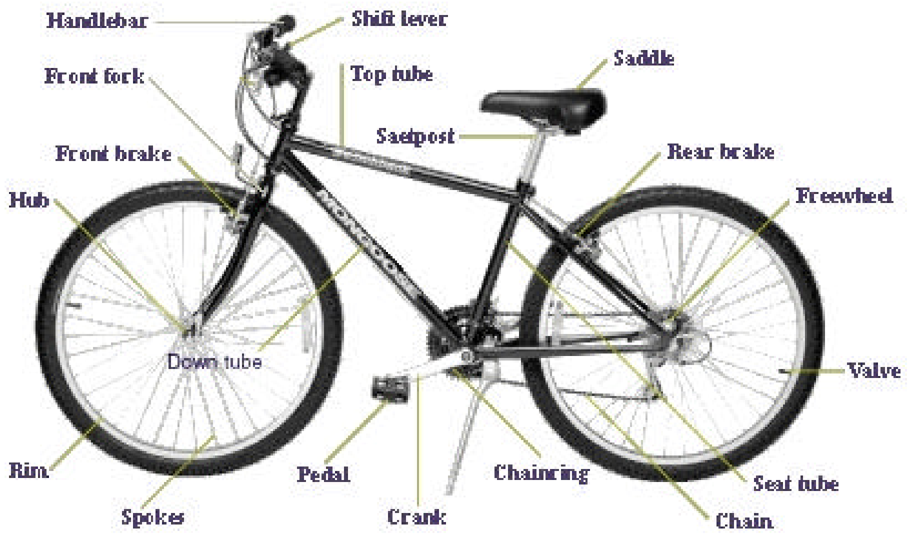

Figure 3. A decomposed bicycle sketch

Step 4. Whenever a component design task has been finished, the component will be passed to the assemble agent. The assemble agent checks the components according to their relations and constraints of the design requirements. If conflict occurs, the design engineer, with the help of communication agent and collaborative agent, solves it by negotiation. This will initiate the redesign process. This process will repeat until all components are assembled and satisfy the design requirement.

6. Conclusions

The work described in this paper is a part of the continuing project done by the Design Technology Research Centre (DTRC) in the School of Design at the Hong Kong Polytechnic University (Frazer 2001; Liu, Tang and Frazer 2001b).

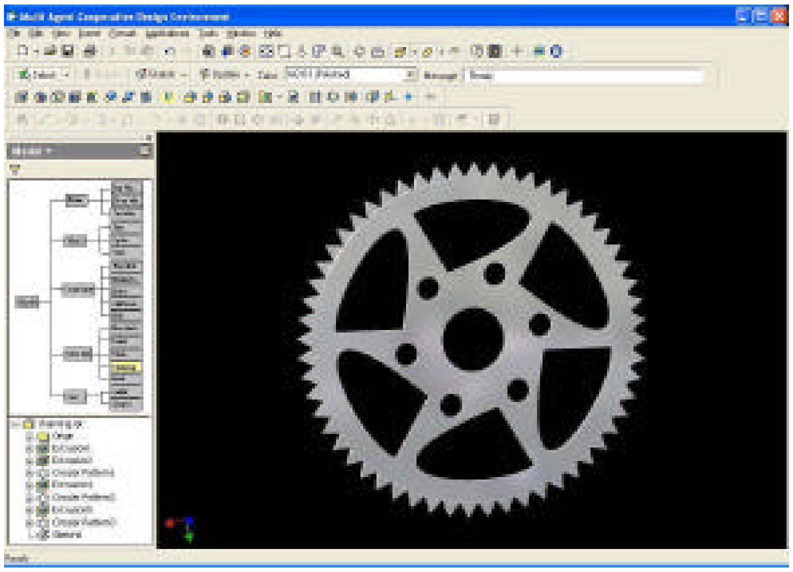

Figure 4. A chain ring component and related features

There is still much work to be done before the full potential power of the system can be realized. Our current work is to use the multi-agent architecture as an integrated knowledge-based system to implement a number of learning techniques including genetic algorithms and neural networks. These new algorithms will then be fully integrated with a selected set of 2D (sketching) and 3D (surface and solid modelling) tools and other design support systems. This integrated system is intended for supporting knowledge based collaborative design in a visual environment.

Acknowledgements

This project is founded by the Research Fellow Matching Fund Scheme 2001 (No. G.YY.34, No. G.YY.35) of the Hong Kong Polytechnic University, and supported by The Hong Kong Polytechnic University (No. 5197/98H and 5275/00H) and NSFC (No. 69975010).

References

Barber, KS and Kim, .:1990, Toward flexible tolerant intelligent manufacturing: sensible agents in shop-floor control, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **14**:337-354.

Frazer, JH: 2001, Design workstation on the future, *Proceedings of 4th International Conference on Computer Aided Industrial Design and Conceptual Design,* International Academic Publishers, World Publishing Corporation, pp.17-23.

Liu, H and Zeng, GZ: 1997, An agent-based approach to cooperative design, *Proc. of Workshop on CSCW in Design'97*, International Academic Publishers, pp. 191-195.

Liu, H, Tang, MX and Frazer J: 2001a, Supporting learning in a shared design environment, *Journal of Advances in Engineering Software* **32** (4): 285-293.

Liu, H, Tang MX and Frazer J: 2001b, Supporting learning in a multi-agent cooperative design environment, P*roceedings of 4th International Conference on Computer Aided Industrial Design and Conceptual Design,* International Academic Publishers, World Publishing Corporation, pp. 477-482.

Tang, MX: 1996. *Knowledge-based Design Support and Inductive Learning,* PhD Thesis, Department of Artificial Intelligence, University of Edinburgh.