Transactions on Information and Communications Technologies vol 8, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

SINE: support for single function agents

D.C. Brown, B.V. Dunskus, D.L. Grecu, I. Berker
Computer Science Department, Worcester Polytechnic Institute, 100 Institute Road, Worcester, MA 01609, USA

1.0 Introduction

Our premise is that Design Expert Systems can be built using many small, cooperating, limited function expert systems, called Single Function Agents (SiFAs). We expect to be able to investigate and discover specific design-related primitive problem-solving and interaction patterns. The approach should also lead to a deeper understanding of the types of knowledge involved. In contrast to other multi-agent approaches, which are based on powerful agents with relatively unconstrained functionality and knowledge, we try to start with elementary components and add structure and functionality as necessary.

The research goals are: Definition of a Domain Independent Set of Agents; Investigation of Agent Negotiation for Conflict Resolution; Analysis of Communication Patterns between Agents; Identification of Appropriate Agent Knowledge and its Representation; and Classification of Conflicts.

The work is motivated by the fact that in Concurrent Engineering (CE), participants from different backgrounds have to work together, in order to anticipate the downstream consequences of design decisions. Due to their heterogeneous backgrounds, their knowledge, goals and preferences are different.

If we try to build expert systems that support or model the CE approach, we face the issue of inherently conflicting knowledge and interests. Such conflicts are difficult to accept in traditional expert system architectures. Usually this means that we have to build a knowledge base which is mostly conflict free, which in turn requires us to anticipate at development time all possible situations and outcomes, in order to avoid the conflicting ones.

This conflict-anticipation would have to be done not only while developing, but also during maintenance of the knowledge base. As expert systems are meant to work in domains where enumerating or trying all possible solutions is prohibi-

tively expensive, and unreasonable to do in advance, we want to find a different way of solving this system design problem.

The work is influenced by several aspects of the research literature. The design-related piece of the Generic Tasks (GT) work, the language DSPL [Brown & Chandrasekaran 1989] [Brown 1992 b], has provided a model of design problem-solving, and gives developers a powerful set of knowledge primitives in which they can describe the design task. SiFA research also relates to Distributed Artificial Intelligence (DAI) [Huhns & Gasser 1989]. It shares with DAI such important issues as memory sharing, information passing between agents, conflict resolution via negotiation, splitting a task into multiple subtasks and assembling the solution from sub-solutions.

1.1 Single Function Agents

Based on experience with concurrent engineering expert systems, we have developed a model which involves multiple agents that cooperatively produce a solution. In particular, it has been found useful to separate the functionality of the entire system into many, small functional types and assign exactly one of these to each individual agent. These agents are called Single Function Agents, SiFAs for short. We define an agent by specifying three parameters: Function, Target and Point-of-View.

The *function* of the agent defines what kind of work it performs. Possible functions are providing Advice, Analysis, Criticism, Estimation, Evaluation, Planning, Selection and Suggestion. The *target* defines on what parameter or object the agent has an immediate effect. A target could be an artifact's material, or the process used to produce the object. The *point of view* specifies the perspective that the agent takes, as it performs its function on the target. That might be cost, strength, manufacturability etcetera. Agents will usually try to optimize the performance of the artifact with respect to its point of view.

Given these three parameters, we can specify agent roles, such as a selector (function) for a material (target) from the point of view of strength. Many agents share some of their parameters, i.e., they have identical functions, targets and/or points of view. This will prove to be useful in the design of the system, as well as interesting during the analysis of conflicts. Not all combinations of the three parameters will necessarily be meaningful.

1.2 Previous Work

Klein [Klein 1991] introduces an approach to conflict resolution based on classifying the conflict and mapping it to a specific strategy. Sycara's work (e.g., [Sycara 1990]) proposes methods for resolving conflicts via negotiation and is the seminal work in the area. SNEAKERS, the first SiFA system [Douglas et al. 1993], assists the designer by using expert systems that offer criticisms and suggestions concerning design decisions. It had several agent types [Brown 1992 a]. I3D, the next SiFA system [Victor et al. 1993], had points of view for the agents. A rigid sequencing of the agents' execution was adopted. Conflicts were

eliminated at development time. I3D+ removed some of I3D's restrictions. Conflicts were allowed and a conflict resolution mechanism implemented. Rigid agent sequencing was replaced by a flexible task-based agenda mechanism. Agents negotiate directly with each other using a language based on speech acts.

2.0 Single Function Agents

2.1 Agent Types

A *Selector* picks one item from a set of alternatives (e.g., materials), using preferences to rank alternatives, and constraints to restrict them. An *Advisor* is a more general form of the selector, not restricted to a list. For example, the options could be described by a set of constraints on a numeric parameter.

An *Estimator* produces approximate values derived from values of design attributes. These values are "approximate" because: not all the input parameters are available; some of the input values are by themselves approximated; the agent uses a faster estimation function that is only close to the real function. The *Evaluator* uses design goals and artifact attributes to evaluate the design from one point of view. The result is information about design quality, i.e., how well the design meets the requirements or some explicit goal. The evaluation can be expressed numerically or in an abstract way (e.g., 'high'). The evaluation does not judge whether the quality is satisfactory.

A *Critic* points out potential problems, and poor/suboptimal decisions that stand in conflict with the requirements or preferences. Positive comments are generated by *Praiser* agents. The *Suggestor* takes a criticism and its context and suggests alternative solutions or recommended actions to achieve a solution.

2.2 Knowledge Structure & Agent Functionality

Agents need Design Knowledge, Conflict Resolution Knowledge, and Communication Knowledge. The design knowledge allows the agent to perform its design task, and can be encoded in rules or in a more functional language. The method of encoding is not a key issue for this work. There is also knowledge devoted to redesigning, used after design decisions are retracted.

Agents need conflict resolution knowledge, containing all the methods used to detect and classify conflicts, as well as strategies for conflict resolution. One task is to analyze which agent has generated the conflict. This "blame assignment" task in SiFA systems consists of tracing origins of values, or using knowledge about value dependencies, to find the "culprit".

Communication knowledge consists of speech act construction and interpretation. The first produces and dispatches messages to other agents, the second retrieves messages that are addressed to the agent and provides immediate reaction to the message. In addition to the knowledge types described above each agent includes in its structure a local memory that serves as a repository for the current design history, local facts and goals.

Negotiation relies heavily on the ability of the agents to communicate and to understand each other. Negotiation depends on the conflict type and the available strategies. Once a common solution has been agreed on, the agents will need functionality to implement it. For some agents that might involve redeciding their values and often it includes adding, relaxing or removing constraints, or activating different rule-sets.

3.0 Conflicts

Agent-agent conflicts can be summarized in a matrix with rows being the agent types that initiate the conflict (by discovering it), and the columns the partners with which they negotiate. The entries summarize the conflict causes and solutions. In some places multiple types of conflicts exist between the same pair of agent types (e.g., Selector-Selector). Not all possible agent pairs can conflict. Also, some conflicts are not yet supported by SINE.

For some of the more important conflict situations we describe the object of the conflict, the cause, the strategy that could be used to solve it and an explanation of why this conflict can occur.

Estimator and Selector/Advisor Conflicts: Not enough information available for estimation, because the estimator agent does not have or cannot find enough information about attributes or qualities of the current choice. This could happen, for example, if the estimator does not have information in its database about the material that got selected. ➪ Strategy: Select an object which can be estimated better, i.e., which has that information.

Evaluator and Estimator Conflicts: Evaluator cannot produce an accurate/reliable enough output, due to inaccurate estimator output. May be due to use of a rough model or partial use of available information. When an evaluator has to produce a detailed evaluation, it might need detailed estimates from the estimator. ➪ Strategy: Switch to more detailed model or use more information.

Evaluator and Selector/Advisor Conflicts: Selector/advisor decided on a value, but evaluator doesn't have enough information to perform its analysis. This could happen, for example, if the evaluator does not know all the properties of the material selected. ➪ Strategy: Select an object which can be better evaluated, for which the evaluator has the necessary information.

Selectors Conflicts: Different selections were made by the agents because of differing preferences. This conflict is most commonly found in the literature. ➪ Strategy: Agree on common value, or convince other agent to abandon choice in favor of more important/global goals.

Selector and Estimator/Evaluator Conflicts: Not enough information available for selection, because some criteria pertinent to the selection process are not available. The value(s) provided by the estimator/evaluator do not provide enough information for the selector (e.g., for statistical values: an estimated value, a reliability factor and an error tolerance always go together). ➪ Strategy: Add needed information to the evaluation/estimation result.

Critic and Estimator/Evaluator Conflicts: Critics can be used to ensure that a certain quality of work is performed by other agents. When an agent uses a quick and shallow approach (e.g., during rough design), this might not be acceptable in later design stages. ➪ Strategy: Switch to a more detailed model.

Critic and Selector Conflicts: When a specific constraint fails for a particular selection, and the critic believes that the selector could interpret the constraint itself, then the critic can request that the selector incorporate the failing constraint into its design knowledge. ➪ Strategy: The selector redoes the selection, with the additional constraint.

3.1 Conflict Types

Although there are multiple possible conflicts between SiFAs, they fall into a limited number of categories. The conflict types recur in several agent type combinations.

Not enough information in output: One agent needs information from another agent and the information provided is less descriptive than needed. This conflict occurs with all agents that use the output of another agent, especially the Evaluator-Estimator, Evaluator-Selector, Selector-Estimator, Selector-Evaluator, and Critic-Selector.

Information quality not sufficient: Where an agent uses another's output, the using agent might have quality requirements. Example agent pairs are Evaluator-Estimator, Selector-Evaluator, Critic-Estimator, and Critic-Evaluator.

Poor processing model: Occurs when agent has knowledge about another agent's different processing model (e.g., An agent has both a rough and a detailed model). Knowledge about these processes is acquired through communication. It is mostly used by critics, suggestors and selectors.

Differing preferences: When several agents compete in assigning a value to the same design parameter, they can have differing local optima. Two selectors can have different first choices.

Design constraint violation: Violations of constraints relating to values of parameters produced by selectors and advisors are most commonly discovered by critics.

3.2 Conflict Resolution Types

Add needed information: When an agent needs more information as input, the agent which produces this information as its output can try to fill in the kinds of data that have been requested. If the agent can't produce the values, it may supply default values, or the information user may make default assumptions.

Improve information quality: Information quality is often dependent on the amount of processing. For estimators, statistical information can sometimes be improved by a more thorough analysis of the data, producing a higher reliability and/or less error in the estimate.

Change to a better processing model: A poor or shallow model may be used for rough design. For detailed design these models should be abandoned in favor of better models, which use the additional information available at that point.

Agree on common value: Depending on the strategy, an agreement can be reached through different means. The agents are assumed to be cooperative.

4.0 SINE: A Platform for SiFA Systems

SINE is the platform developed for research into negotiating SiFAs. After the development of several SiFA systems, there was considerable evidence that some of the structure and functionality was recurring. Based on this experience several goals appeared to be relevant for the development of the SINE platform:

Build a reusable SiFA system: Allow easier development of SiFA based systems. Much domain-independent design functionality should be made available, in order to facilitate reusability of most of the knowledge.

Implement domain-dependent and domain-independent conflict resolution knowledge: Supply general conflict resolution rules and strategies that could be used as a basis for implementing more specific techniques in critical areas.

Design and implement a negotiation language for SiFAs: A language for negotiation would allow present and future agents to be compatible and set a standard for SiFA communication.

The current SINE platform is geared toward routine parametric design problems. These are characterized by the fact that the structure of the design object, its features and the parameters influencing it are known in advance. The design task consists of making decisions about values for design attributes, which can be either parameters (lengths, widths, colors, materials etc.) or general design decisions (e.g., selecting a structure type for the artifact or determining the class of material to use).

The SINE system allows a flat, totally connected structure. This means that all agents are able, if necessary, to communicate with all others, and no agent has a hierarchical priority over any other agent. For efficiency and coherency reasons, all the information about the design parameters is located in one place. Agents use their communication features to request information from and send information to other agents. The results of direct agent interaction through negotiation are used to update the design status.

4.1 Agent Implementation

Every agent in SINE is an instance of some class. There is a specific class for every agent, where the class is based on features inherited from a specific agent type class, a target class and a point of view class.

The selector agent uses an algorithmic approach using constraint and preference objects to filter and rank a set of initial alternatives. If it cannot select its most preferred choice because of a conflict with another agent it starts negotiating with that agent. The critic uses an algorithm that iterates through a list of con-

straint objects. If a constraint fails, blame assignment is done by tracing the origin of the design parameter that lead to the failure and a negotiation is initiated with that agent. The estimator and the evaluator can use preference class objects to perform their evaluations, but experience has shown that more rule-based and domain dependent knowledge is used here.

The SINE system uses a combination of rules and reactive objects to represent knowledge. Constraints and preferences are represented as objects. The knowledge referring to the artifact to be designed consists of general rules, constraints and preferences which can be applied to the design problem, as well as the specific requirements of the current design task.

The communication mechanism is implemented through messages. The message format is based on speech acts. They can mention alternatives (e.g., "Add the following alternative to your list of considered choices"); constraints (e.g., "The constraint is satisfied); preferences (e.g., "What are your preferences?"); proposals (e.g., "I don't agree with your proposal for design-value x").

4.2 Conflict Detection, Notification, and Resolution

Conflicts are detected by agents. The object-oriented architecture supports this. For example, *Selection-Selection Conflicts* are detected by means of a failure indication from a parameter object when an agent tries to assert a value for a design parameter which has already been determined. Conflict information is posted in the central database, and the initiating agent starts the conflict resolution. How exactly the agent solves its conflict is up to the developer of the agent, as long as the agent uses the SINE language. It is not necessary that all agents have the same conflict resolution knowledge. The agents are responsible for recognizing that the conflict has been solved or that no solution is possible.

4.3 Implementation

SINE has been developed using the CLIPS (C Language Integrated Production System) language. CLIPS is a forward chaining rule-based inference engine. For this project, we have used the UNIX–X/Windows version of CLIPS. The system was developed on a DEC Alpha^{TM} 3000-300, using GNU C.

5.0 Conclusions

The SINE platform has been used to implement some of the of the I3D+ system [Dunskus 1994]. The system design was much simpler, flexible and maintainable than the previously. A sailboat design advisor was also implemented. SiFAs are easy to use and facilitate system development. The small agent granularity allows a large amount of inheritance of knowledge and function. Detecting conflict causes and blame assignment is made easy because the responsibilities are very explicit, due to the strict functional separation of the agents. The conflict resolution can usually be performed within a few question/answer cycles, which is also due to the small agent size. The SiFA approach to investigating negotiating multi-agent design systems offers a new way of studying their ingredients. It provides ways of discovering and testing the compo-

nents of negotiation, patterns of communication, functional primitives of design, and the types of knowledge needed. By restricting the ability of agents, their behavior can be studied and categorized. This is not possible with larger, more general-purpose agents. The approach, although still being investigated, has already produced interesting results and shows much promise.

References

Brown 1992 a

D.C. Brown. "Design". *Encyclopedia of AI*, 2nd edn., S. Shapiro, ed., J. Wiley, 1992.

Brown 1992 b

D.C. Brown. "The Reusability of DSPL Systems". *Workshop on Reusable Design Systems: Understanding the Role of Knowledge, 2nd Int. Conf. on AI in Design*, Carnegie-Mellon University, Pittsburgh, USA.

Brown & Chandrasekaran 1989

D.C. Brown, B. Chandrasekaran. "Design Problem Solving: Knowledge Structures and Control Strategies". *Research Notes in Artificial Intelligence Series*, Pitman Publishing, Ltd., London, England, May 1989.

Douglas et al. 1993

R.E. Douglas Jr., D.C. Brown & D.C. Zenger. "A Concurrent Engineering Demonstration & Training System for Engineers and Managers". *Int. Jnl. of CAD/CAM & Comp. Graphics*, special issue, (ed.) I. Costea, Hermes, Vol.8, No.3, 1993, pp. 263-301.

Dunskus 1994

B.V. Dunskus. *Single Function Agents and their Negotiation Behavior in Expert Systems*. MS Thesis, Worcester Polytechnic Institute, 1994.

Huhns & Gasser 1989

M.N. Huhns, L. Gasser (eds.). *Distributed Artificial Intelligence Vol. 2*, Pitman Publishing, London, and Morgan Kaufmann Publ., CA, 1989.

Klein 1991

M. Klein. "Supporting Conflict Resolution in Cooperative Design Systems", *IEEE Trans. Systems, Man & Cyb.*, Vol.21, No.6, Nov/Dec 1991.

Lander & Lesser 1991

S.E. Lander & V.R. Lesser. "Customizing Distributed Search Among Agents with Heterogeneous Knowledge", *Proc. 5th Int. Symp. on AI Applications in Manufacturing & Robotics*, Cancun, Mexico. Dec 1991.

Sycara 1990

K.P. Sycara. "Cooperative Negotiation in Concurrent Engineering Design", *Cooperative Engineering Design*, Springer Verlag, 1990.

Victor et al. 1993

S.K. Victor, D.C. Brown, J.J. Bausch, D.C. Zenger, R. Ludwig, R.D. Sisson. "Using Multiple Expert Systems with Distinct Roles in a Concurrent Engineering System for Powder Ceramic Components", *Int. Conf. on AI in Engineering*, Toulouse, July 1993.