Using Single Function Agents to Investigate Conflict

B. V. Dunskus, D. L. Grecu, D. C. Brown & I. Berker

Computer Science Department, WPI, Worcester, MA 01609, USA.

ABSTRACT

Design Expert Systems can be built using many small, cooperating, limited function expert systems called Single Function Agents. Using this approach we will be able to investigate and discover primitive problem-solving and interaction patterns, specific for multi-agent design systems, and should gain a deeper understanding of the types of knowledge involved. This paper presents some categories of conflicts that have been studied using the SiFA approach, and makes a brief presentation of the SINE implementation of SiFAs.

1.0 Introduction

The premise of this research is that Design Expert Systems can be built using many small, cooperating, limited function expert systems. We have called these expert systems **Single Function Agents** (SiFAs), for reasons which will become obvious later in this paper. The belief is that by using this approach we will be able to investigate and discover primitive problem-solving and interaction patterns, specific for design using multi-agent systems. The approach should also lead to a deeper understanding of the types of knowledge involved. In contrast to other approaches for distributed design systems, which are based on powerful agents with relatively unconstrained functionality and knowledge, we try to start with elementary components and add structure and functionality based on encountered necessities.

1.1 Goals

The research follows both theoretical and practical goals:

- *Definition of a Domain Independent Set of Agents.*

  Our design systems, based on ideas from Concurrent Engineering, are built by using a set of agents, possessing various types of knowledge and different functionalities. A major aim is to design agent types that are suitable for larger classes of design tasks, not only for individual problems.

- *Investigation of Agent Negotiation.*

  Agents may conflict during the design process. For example, two agents may propose different values for the same parameter, or an agent may criticize the decision of another agent. Conflict resolution, with negotiation as the main technique, is therefore a very important issue. Solution generation depends on the capability of the agents to reach an agreement.

- *Analysis of Communication Patterns.*

  Based on the agent interaction, we intend to find common patterns in the information exchange between the agents. Suitable communication primitives can highly improve the convergence to a solution. Agents of limited functionality can be expected to have limited information to communicate. Common patterns of information exchange should form the basis of all design-related inter-agent negotiation.

- *Knowledge Representation.*

  Past experience has shown that the way in which the knowledge is stored in a system can have a severe impact on its design, maintenance and its performance. Hence, we want to analyze possible representations and choose the ones most suited for the task. We wish to identify the necessary types of knowledge for making design decisions, for making redesign decisions, for detecting and identifying conflicts, for selecting a conflict resolution strategy, and for responding to the other agent during negotiation.

- *Catalog of Conflicts.*

  Investigating the design solution generation and the agents' negotiation behavior will provide us with information about potential and actual conflicts. We intend to set up a conflict catalog, in which to specify when, where and why the conflicts appear, as well as what strategies might be used to resolve them. Single Function Agents provide an approach to addressing this problem.

- *Making design agents learn.*

  A longer term goal is to investigate the opportunities that Single Function Agents provide for learning during design. In addition we are interested in how more powerful design agents might learn by using exchanged beliefs, intentions, plans and constraints, as well as from design history.

1.2 Motivation

1.2.1 The Practical Side

From a practical point of view, two major issues were important to this work. The first is 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 prohibitively expensive, and unreasonable to do in advance, we want to find a different way of solving this system design problem.

The second practical issue is that we often want to integrate human and computer experts in one system. This is essential for computer aided engineering. We therefore need a way to provide an interfacing possibility that is suitable for both the agents and the human experts. The knowledge that is exchanged has to ensure that each person or agent understands the requests, the explanations, and the decisions of the other.

1.2.2 Theoretical Motivation

On the theoretical side, the work carried out in the area of Generic Tasks (GT) and design system languages such as DSPL [Brown & Chandrasekaran 1989] [Brown 1992 b], has provided a significant impact on the abstraction of design problem-solving. The generic tasks research attempts to formulate human information processing tasks on an abstract level. Design System Languages such as DSPL give developers a powerful computer language in which they can describe the design task. The language supplies constructs, which represent the steps or tasks that would have to be executed by a human designer and the system simulates that task and, if possible, comes up with a design, in response to requirements. A major challenge is to incorporate into Single Function Agents those elementary reasoning processes that are specific for concurrent engineering influenced design.

Another influence on SiFAs is Distributed Artificial Intelligence (DAI) [Bond & Gasser 1988] [Huhns 1987] [Huhns & Gasser 1989]. SiFAs share important issues with that work, such as memory sharing, information passing between agents, splitting a task into multiple subtasks and assembling the solution from sub-solutions.

Conflict resolution (CR) has recently become an increasingly formalized area and general frameworks are being proposed. SiFAs offer a rich set of possibilities to explore conflicts among agents at a low level of granularity and to extract results about CR by putting together individual pieces of evidence discovered in elementary interactions.

1.3 Single Function Agents

While investigating expert systems to support concurrent engineering tasks, we have developed a model which involves multiple agents that cooperatively produce a solution. The functionality of the entire design process is separated into many, small functional types. One functional type is assigned to each individual agent. These agents are called Single Function Agents, SiFAs for short. As this idea proved its usefulness in applications, it has been extended to encompass additional aspects. In our current approach we define an agent by specifying three parameters (see Figure 1-1).

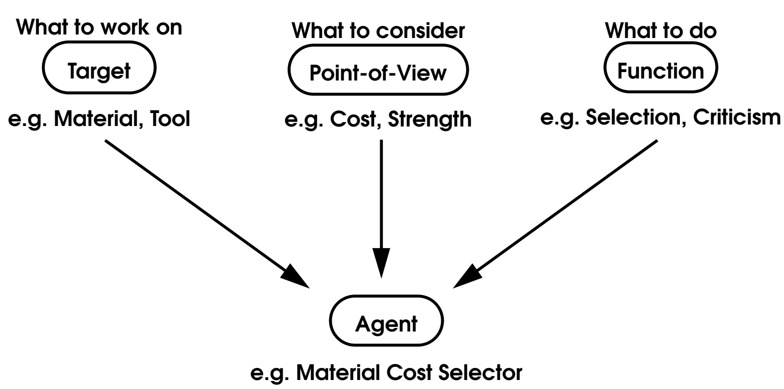

FIGURE 1-1: SiFA Definition

- *Function*

  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. These functions are similar in spirit to the generic tasks previously mentioned.

- *Target*

  The target defines the parameter on which the agent has an immediate effect. A target could be the material that an artifact is made from, or the process that will be employed to produce the component.

- *Point-of-View*

  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 etc. Agents will usually try to optimize the performance of the artifact with respect to their point of view.

Given these three parameters, we can now easily specify agent roles. For example, we could have a selector (function) for a material (target) from the point of view of strength. Another role could be criticism of material from the point of view of cost. Maybe a third agent would select material from the perspective of thermal-conductivity, which would probably lead to different preferences than the first selector, and therefore to conflicts.

Figure 1-2 shows examples of the wide range of choices that can be used to specify the agent roles.

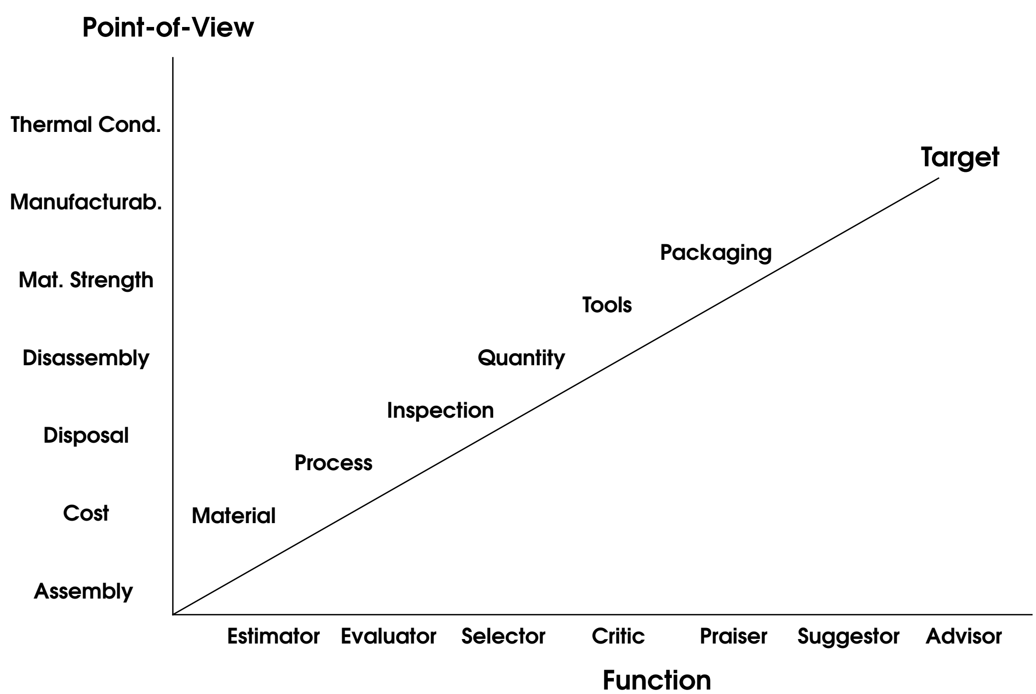

FIGURE 1-2: SiFA Dimensions

We already see that many agents can 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 negotiation.

1.4 Conventions

For clarity, we define the following terms:

- SINE: The negotiation platform we have developed for SiFAs.
- User: The person that uses a design expert system based on SINE.
- Developer: The designer and programmer of an expert system that uses the SINE platform and language.
- Agent: An expert system that can act and behave, based on built-in knowledge, and that communicates with other agents.

1.5 Outline of the Paper

The remainder of this paper is structured into four parts:

The following section will give a brief review of previous work relevant to our approach. This description will refer to a few models which exemplify current methods used for conflict resolution and the work which led us to the development of Single Function Agents.

A second section is devoted entirely to the insight in SiFAs, their classification, knowledge structure and functionality. A special part is reserved for conflict analysis and conflict resolution strategies.

Strongly connected to the section about SiFAs is the section introducing the negotiation platform SINE, developed to allow SiFA conflict resolution. In addition to the issues related to conflicts we briefly investigate communication, agent control and implementation.

The conclusions summarize the insights drawn from the ongoing research and the experiences from the development of, and work with, the SINE platform. We also present the opportunities we consider most promising for the future research in the field of SiFAs.

2.0 Previous Work

In this section, we review some of the relevant research literature. We will present research that has focused on conflict resolution and generated various models to support it. We will also present three systems that have been developed using the SiFA paradigm.

2.1 Conflict Resolution

Klein [Klein 1991] introduces an approach to conflict resolution based on conflict and strategy hierarchies (Figure 2-1). The first type of hierarchy consists of a tree of conflicts, the most abstract one at the top and the specialized conflicts at the bottom. If a conflict is detected, it is matched against the nodes of the tree to recognize it and to find the most specific conflict type. At the same time as the conflict is analyzed, the path to the root of the tree provides a series of abstractions that can be used during conflict resolution.

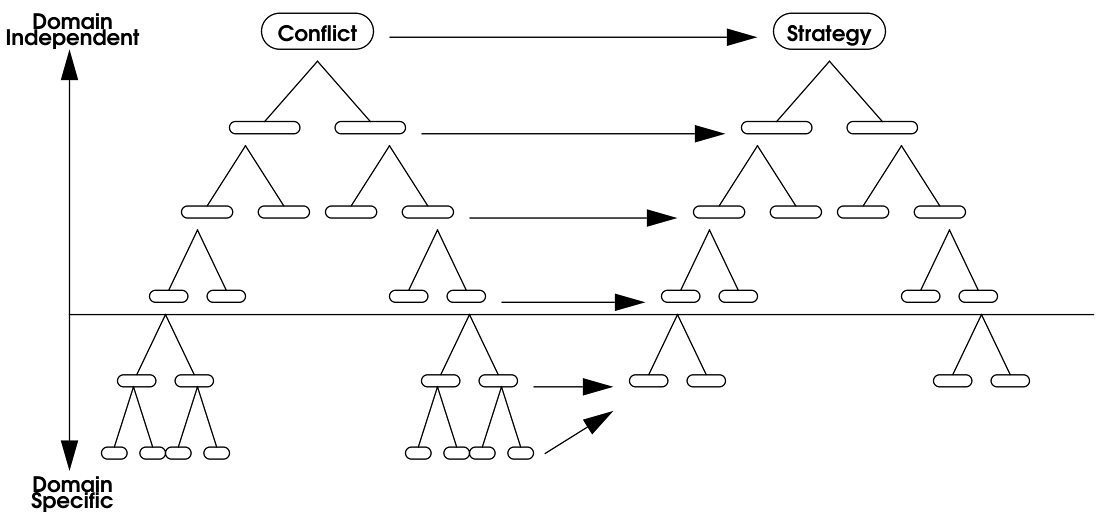

FIGURE 2-1: Klein's Conflict Resolution Approach

The second tree consists of conflict resolution strategies. After the conflict type and its abstractions are determined, the system tries to find the most specific strategy that is appropriate for this conflict and solve the conflict by applying it. If the strategy fails, a more general method is used, and so forth, until all the strategies up to the root of the strategy tree have been applied. If that one fails too, it implies that the conflict definitely cannot be solved.

The remarkable aspect of this model is that it supports both domain dependent and domain independent conflict resolution, and that there is a smooth transition from one to the other. It is also able to recover from failures in the conflict resolution process, by applying the more general approaches, if the more specific ones fail.

With respect to conflict resolution by negotiation**,** Lander and Lesser [1991] present the TEAM system, which implements the negotiated search paradigm. In this approach to conflict resolution *extended search* is applied by an agent recognizing a conflict in order to find another solution in its local search space that avoids the conflict, while *relaxation* is used as a way to expand the local solution space if the search is constrained or expensive.

Conflict resolution combines *search* and *solution composition*. Local search by an agent produces a solution to a subproblem. These solutions have to be integrated into an overall solution in the composite space given by the intersection of the local solution spaces. A selected solution is acceptable if it satisfies the requirements imposed by each agent and by the user.

Sycara [1990] proposes *case based negotiation* as an efficient method. As negotiation is often a complex and expensive process, efficiency is important. Reusing sets of decisions which proved to be successful in the past, especially if the same problems repeat, provides for reduced communication during conflict resolution.

2.2 SiFA Systems

SNEAKERS, WPI's first SiFA system, is a single-user demonstration system that teaches principles of concurrent engineering. The user's task is to design towers composed of pieces. The system assists by using various expert systems running in the background that offer criticisms and suggestions concerning design decisions [Douglas et al. 1993].

SNEAKERS has several agent types (adopted from [Brown 1992 a]). *Advisors* make recommendations to the user for the next decision; *Critics* compare the design to standards, and offer criticism on the last user action; *Suggestors* take the criticism and offer suggestions for satisfying them; *Analysts* offer numerical analysis to derive attributes, such as strength, cost or size; *Evaluators* evaluate the whole design from one perspective, and determine how well the design meets the needs of a certain perspective. Agents can have different points of view, such as Design, Manufacturing, Assembly, Cost, Marketing etc. As this work was based on an incipient version of the SiFA idea, it only separated the agents by function and point of view. There was no distinction by target.

I3D, a design aid in the development of powder ceramic components, interacts with a designer sitting at a workstation [Victor et al. 1993]. As the designer moves through the requirements specification, the conceptual design and the detailed design of a part, the system graphically displays the state of the design. In both design phases, the system provides feedback about the material, the production process and the inspection points. There are different agent types, and Manufacturing, Cost, Reliability, and Durability points of view. The agents are partially dependent on one another, so a rigid sequencing of the agents execution was adopted. The system does not deal with conflicts. They were eliminated at development time, by not allowing more than one agent to select a value for the same design parameter.

In I3D+ an attempt was made to remove I3D's restrictions [Victor 1993]. The potential for conflicts was allowed and a mechanism was implemented to demonstrate how conflict resolution might be handled. The rigid agent sequencing was replaced by a flexible agenda mechanism, which schedules the agents, based on tasks that they announce. Since the agents handle conflict resolution by negotiation, they communicate directly with each other using a language based on speech acts.

The conflicts are classified into 6 situations, depending on the relation between the agents' local goals and the global goal, as shown in Table 1-1. (The grey areas have not been implemented.)

	Agent1 Goal	**Agent2 Goal**	**Global Goal**	**Result**
1	X	Y	Z	Either one will win
2	X	Y	X	Agent1 will win
3	X	Y	Y	Agent2 will win
4	X	Y	X,Y	Either one will win
5	X	X	Z	Either one will win, depending on their expertise
6	X	X	X	Either one will win, depending on their expertise

TABLE 2-1: Conflict Situations in I3D+

However, some aspects of I3D+ invite for further extensions and reconsideration. Only goal importance is used as conflict resolution knowledge. The conflict resolution is domain dependent, encoded in the agents knowledge base. There is no support for more abstract, domain independent resolution. Agents have no facility to talk about their design knowledge, and therefore cannot learn about other agents' reasons for failure. Agents have to detect potential conflicts themselves, by checking the values on the blackboard. This is expensive, as it has to be done every time an agent is considered for scheduling.

3.0 Single Function Agents

This section focuses on describing single function agents in more detail. The agent types reflect our view of the major functions involved in the design process. The knowledge and functional requirements determine the ingredients used for constructing such agents. Finally, the conflict types and conflict resolution give us a flavor of the behavior of these agents, and their potential.

3.1 Agent Types

The current set of agent types has resulted mostly from the experiments with the previous agent-based systems, described above. The various agent types mirror the most often required functions during design.

3.1.1 Selector/Advisor

A Selector picks one item from a set of alternatives. In doing so, it can use preferences to rank alternatives, and constraints to restrict the alternatives to valid choices. It could pick a choice of a material from a list, or a value for some parameter of the design.

The Advisor is a more general form of the selector. It is not bound to use an enumerated list of possible choices, but it has an abstract description of what it can choose from. For example, the options could be described by a set of constraints on a numeric parameter, requiring the advisor to use an appropriate method to produce a specific value.

3.1.2 Estimator

An Estimator produces approximate values (usually numeric values with error margin and probability) derived from values of attributes of the design. These values are "approximate", because not necessarily all the input parameters are available, or some of the input values are by themselves approximated or statistical. In addition to that, the agent can use an estimation function that is only close to, but is not exactly the function that applies in real life. This is often done to save time and effort in early design stages.

3.1.3 Evaluator

The Evaluator uses design goals and attributes of the artifact to evaluate the design from one point of view. The result is information about the quality of the design, i.e., how well the design meets the requirements or some explicit goal. The evaluation can be expressed numerically (e.g., in percent), or in an abstract way (e.g., 'good', 'average' or 'poor'). The evaluation does not make any statements about whether the achieved quality is satisfactory or not.

3.1.4 Critic/Praiser

A Critic points out potential problems, suboptimal decisions and poor choices that stand in conflict with the preferences that are expressed in the requirements. Positive comments are generated by Praiser agents.

3.1.5 Suggestor

The Suggestor takes a criticism, and its context, and suggests alternative solutions or recommended actions to achieve a solution.

3.2 Knowledge Structure

Agents need several kinds of problem-solving knowledge to be able to carry out their tasks: Design Knowledge; Conflict Resolution Knowledge; Communication Knowledge.

The design knowledge contains all the functionality that the agent needs to perform its design task. The knowledge can be encoded in rules or in a programming language. The method of encoding is *not* the main issue for this work. In addition to that knowledge, there is also knowledge devoted to redesigning, to be used after design decisions are retracted.

In order to resolve conflicts, the agents need conflict resolution knowledge. It contains all the methods that are used to detect and classify conflicts, as well as the strategies for conflict resolution. One part of this knowledge is used to analyze what agents have 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". The resulting information is used to analyze the conflict type. In general this could involve tracing conflict history and/or design history information and/or querying the other agent. In SINE, agents can also use their knowledge about normal agent interactions to gather information about the agent type with which they conflict.

The knowledge structures about conflicts also have a memory for storing historical information about previous CR interaction with other agents, such as, their goals, previous proposals that failed, preferences of other agents etc. (see Figure 3-1).

The communication knowledge consists of two parts: speech act construction knowledge and interpretation knowledge. The first is used to produce and dispatch messages to other agents, the second analyzes messages that are addressed to the agent and provides immediate reaction to the message.

3.3 Agent Functionality

Besides the design function which defines each agent, they have additional functionality which makes conflict resolution possible. *Communication* facilitates the information exchange among the agents [Bussmann & Mueller 1993]. Negotiation relies heavily on the ability of the agents to communicate and to understand each other [Davis & Smith

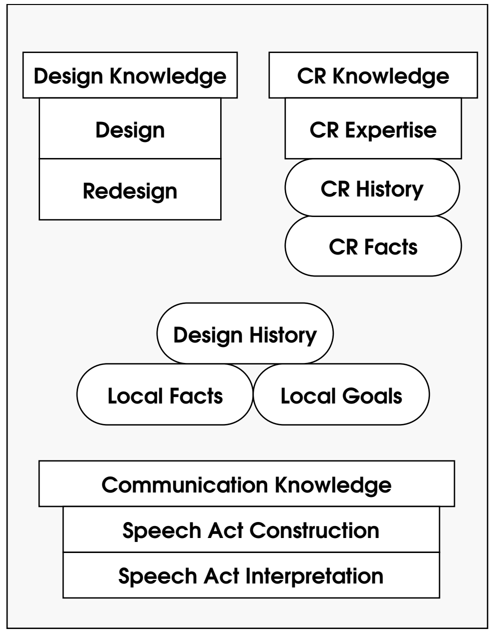

FIGURE 3-1: SINE Agent Structure

1981] [Werkman & Barone 1991]. *Conflict solving* is needed to be able to negotiate and to find possible points of agreements. Negotiation depends on the conflict type and the available strategies. Once a common solution has been agreed on, the agents will need *Solution Implementation*. For some agents that might involve redeciding their values, often it includes adding, relaxing or removing constraints, or activating different rule-sets and functions.

3.4 Conflict Occurrences

In Table 3-1, we show an overview of possible interactions between the different agent types. The first column lists the agents that initiate the conflict by discovering it, while the upper row displays the partners, with which they negotiate. The squares summarize the conflict causes and solutions. The shaded squares show conflicts that are not yet supported by the SINE system. The outlined squares, however, show places where multiple types of conflicts exist between the same pair of agents. Observe that not all the possible agent pair combinations can enter into a conflict, based on the structure, knowledge and functionality currently assigned to the agents. In the table, the Selector entries are also applicable to Advisors.

Initiators (below):	Estimator	Evaluator	Selector	Critic	Praiser	Suggestor
Estimator	Conflicting Estimations ⇒ Modification of Estimation Basis		Poor Estimation Basis ⇒ Select different alternative, or give more details			
Evaluator	Incomplete Evaluation Basis, Poor Estimation ⇒ Modify Estimate, add Attributes to Estimate	Conflicting Evaluations ⇒ Modification of Evaluation Basis	Poor Evaluation Basis ⇒ Select different alternative, or describe selection in more detail			
Selector	Poor Estimation Quality ⇒ Request Modification of Estimate	Undercritical Evaluation ⇒ Request Refinement of Evaluation	Constraint Failure, Preferences ⇒ Removal and Reordering of Alternatives			No selection between several suggestions possible ⇒ Refine Rationale
Critic	Complaint about Quality of Estimation ⇒ Modify Estimation Procedure	Complaint about Quality of Evaluation ⇒ Modify Evaluation Criteria	Disqualification of Alternatives ⇒ Additional Constraints	Conflicting Criticism (e.g. too high and too low) ⇒ Agreement on Cr.	Conflict about quality of praise, too little/too much ⇒ Change praiser policy	
Praiser			Approval of Choices ⇒ Reordering of preferred Selection Method	Conflicting Statements ⇒ Re-evaluate or retract		Unwanted Change Suggestion ⇒ Receive Reason for Improvement
Suggestor	Improve Estimation ⇒ Add/Remove Item Types, Methods	Improve Evaluation ⇒ Add/Remove Attribute Types, Methods	Improve Selection Process ⇒ Intensify Search; Extend Search Space	Enable Selection ⇒ Relax Constraint		Conflicting Suggestion ⇒ Agreement on common suggestion

TABLE 3-1: Conflicts: Partners, Causes and Resolution

Let us now investigate some of the more important conflict situations. For each conflict, we describe what the conflict is about, the cause, and the strategy that could be used to solve it.

3.4.1 Conflicts between Estimator and Selector/Advisor

Description:	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 entity or value which can be evaluated better, i.e., which has that information.

3.4.2 Conflicts between Evaluator and Estimator

Description:	Evaluator cannot come up with an accurate or reliable enough output, due to non-accurate output from the estimator. The cause can be the use of a rough model or the partial use of the available information. When an evaluator has to produce a detailed evaluation, it might need detailed estimates from the estimator.
Strategy:	Switch to a more detailed model or use more information.

3.4.3 Conflicts between Evaluator and Selector/Advisor

Description:	Not enough information available for evaluation, because some information on which to base the evaluation is not available for the current choice. The selector or advisor decided on a value. The evaluator does not have enough information to perform its analysis. This could happen, for example, if the evaluator does not know about the selected material.
Strategy:	Select an entity or value which can be better evaluated, for which the evaluator has the necessary information to perform its function.

3.4.4 Conflicts between Selectors

Description	Different selections were made by the agents because of differing preferences. The can try to agree on a common value, or one agent can try to convince the other agent to accept its own proposal. This conflict is the one most commonly found in the literature, e.g., in [Lander & Lesser 1991] and [Victor 1993].
Strategy:	Agree on common value, or convince other agent to abandon choice in favor of more important/global goals.

Description	The agents made different selections because of different constraints. If the reason for the disagreement is that the agents choose from different sets of alternatives, then it is advisable to build the intersection of the two sets and to try selecting/resolving again.
Strategy:	Build intersection of both agent's possible choices.

3.4.5 Conflict between Selector and an Estimator or Evaluator

Description	Not enough information available for selection, because some criteria pertinent to the selection process are not available for that decision. The value(s) provided by the estimator or 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.

3.4.6 Conflict between Critic and an Estimator or Evaluator

Description	Agents were only using a rough model to come up with a quick answer. Critics can be used to ensure that a certain level of quality of work is performed by the 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. The critic can then request use of a more thorough approach.
Strategy:	Switch to a more detailed model.

3.4.7 Conflict between Critic and Selector

Description:	Unsatisfactory Selection, because of Constraint Violation. When a specific constraint fails for a particular selection, and the critic has reason to believe that the selector could interpret the constraint itself, then the critic can decide to request that the selector incorporate the failing constraint into the selector's design knowledge.
Strategy:	Inform Selector of constraint. The selector redoes the selection, possibly with the additional constraint.

3.5 Conflict Types

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

- *Not enough information in output.*

  This conflict type occurs when one agent needs information from another agent and the information provided is less descriptive than the first agent needs. The conflict occurs with all agents that use the output of another agent, especially the pairs Evaluator-Estimator, Evaluator-Selector, Selector-Estimator, Selector-Evaluator, and Critic-Selector.

- *Information quality not sufficient.*

  Among those situations where an agent uses another agent's output, the agent that uses the information might have quality requirements, e.g., reliability or error tolerance values. Use of high quality information will help produce high quality designs. Agent pairs where this type of conflict is found are Evaluator-Estimator, Selector-Evaluator, Critic-Estimator, and Critic-Evaluator.

- *Poor processing model.*

  This conflict type can only occur when an agent has knowledge about another agent's different processing models. An agent might have both a rough and a detailed model for devising a value. Although knowledge about these processes can be acquired through communication, it is mostly used by critics, suggestors and selectors. The selector for example might request the evaluator to use the detailed model, so that the evaluations are more usable for the selection purpose.

- *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, and also two suggestors may suggest different solution approaches.

- *Design constraint violation.*

  Violations of constraints relating to values of parameters are most commonly discovered by critics and are produced by selectors and advisors, because they are the agents making the design decisions. But a suggestion can also violate an agent's constraints and selectors can have built-in constraints that fail.

3.6 Conflict Resolution

Conflicts can be solved in several ways, depending on the conflict type. The type will affect the nature of the negotiation between the conflicting agents. These types form the basis of a taxonomy of conflict types, as introduced by Klein.

- *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. Often, the agent will not be able to produce the values. In those cases, the producing agent will have to supply default values, or the information using agent will have to make default assumptions.

- *Improve information quality.*

  Information quality is often dependent on the amount of processing done. For estimators especially, statistical information can sometimes be improved by performing a more thorough analysis of the data. This will produce a higher reliability and/or less error margin in the estimate.

- *Change to a better processing model.*

  In many cases a poor or shallow model has only been used to allow rough design. In the detailed design phase 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 and the primary criteria for searching for the common value relate to the generated design.

3.7 Summary

In this section we have presented types of SiFAs, their structure and functionality, and have presented examples of the conflicts we have discovered. Conflict types and conflict resolution types were also presented. These will form the basis of a study of basic types of negotiation. We expect SiFAs to allow us to discover more examples of conflicts and more details of conflict types as we continue our research.

4.0 SINE: A Platform for Negotiating Single Function Agents

SINE is the platform we are currently using for research into conflicts and negotiating SiFAs. We will describe the implementation goals and explain the reasons for its development. We then present the architecture that SINE uses, including the knowledge representation, the data and control flow, as well as the language and communication used. The description also covers the methods used for conflict detection and notification, the conflict resolution model and the scheduling system. Additionally, we briefly explain the design of the individual agent types.

4.1 Goals

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 during the development of the SINE platform:

- *Build a reusable SiFA system.*

  The system should allow easier development of SiFA based systems, especially in the case of parametric design systems. A large amount of 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.*

  A "universal" platform should supply some general conflict resolution rules and strategies that could be used as a background 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, which did not exist prior to SINE.

4.2 Architecture

The SINE platform is most suitable for routine parametric design problems. These are characterized by the fact that the structure of the design, 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).

SINE allows a flat, totally connected structure. This means that each agent is able to communicate with any other, and no agent has a hierarchical priority over any other agent. The flat agent topology can be extended to support agent hierarchy in future research.

For efficiency and coherency reasons, the agents design the artifact on a centralized blackboard, i.e., all the information about the design parameters is located in one place. Agents also 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 on the blackboard.

4.2.1 Agent Implementation

Every agent in SINE is an instance of some class, where the class is based on features inherited from a specific agent type class, a target class and a point of view class. Figure 4-1 shows this multiple inheritance. The three letter abbreviations in the agent blocks are the

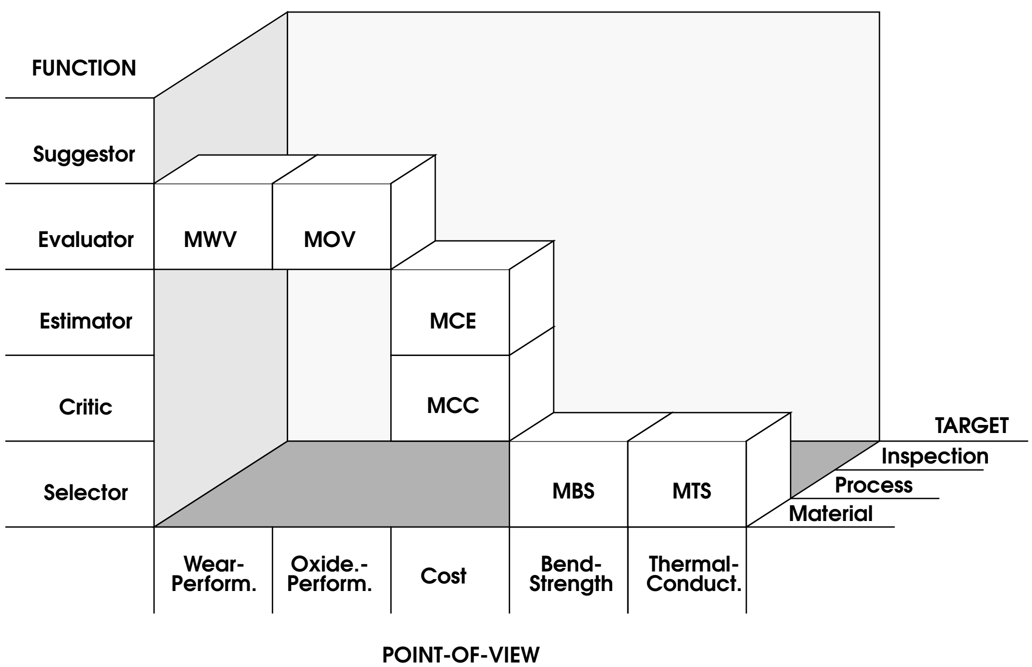

FIGURE 4-1: SINE: Multiple Inheritance for Agents

names of the agents in a test implementation. For example, MCE refers to Material Cost Estimator.

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 constraint 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 the appropriate 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. At the present time, suggestor type agents have not been implemented.

4.2.2 Knowledge Representation

The SINE system uses a combination of rules and reactive objects to represent knowledge. *Constraints* are represented as objects of a constraint class. For agents that need to make a choice between different alternatives, there is a class called *preference*. Its instances store an abstract relation and can be used to produce a (partial) ordering on a set of values.

The knowledge referring to the artifact to be designed consists of two parts. The first part is represented by general rules, constraints and preferences which can be applied to the design problem. This part is thought to change only occasionally, with the addition of new knowledge into the system, or modifications of the design task or design artifact structure.

The second part is represented by the specific requirements of the current design task. This part is volatile and has to be respecified for every new design task. In response to the requirements, agents generate design values for the parameters. These values, together with the constraints, rules and preferences, drive the agents in the design.

4.2.3 Communication

The communication mechanism is implemented through messages. The message format is based on speech acts, i.e., it captures a complete utterance in one message [Searle 1965] [Taleb-Bendiab & Oh 1993]. The utterances are similar to a human sentence, which allows a straightforward adaptation of sentences found in human discussion. The format is similar to the KQML message structure, with slots for storing the primitive, sender, receiver, time stamp, language and ontology information, as well as references to previous and future messages [Finin et al. 1993] [Kuokka et al. 1993].

The SINE communication language defines how the primitive, subject and parameters can be filled to form meaningful sentences. In the following are presented some sample speech acts:

- *Speech acts using alternatives.*

  All agents that produce values by selection, or by elimination, from a set of possible choices (e.g., Selector, Advisor, Suggestor) are able to communicate their alternatives. e.g., “Add the following alternative to your list of considered choices”; “Remove the following alternative from your list of considered choices.”

- *Speech acts using constraints.*

  Selectors restrict their set of choices with constraints. Critics use them to trigger specific messages. Other agents can use them to check their preconditions. Constraints can be failed, satisfied or unknown. e.g., “The constraint is satisfied/unsatisfied”; “Add the constraint to your list of things to consider”; “Remove the constraint from your list”.

- *Speech acts using preferences.*

  Agents that have to search a set of choices for the best option can use preferences for the purpose. e.g., “What are your preferences?”; “My preferences are...”.

- *Speech acts using proposals.*

  A value that an agent asserts for a design parameter is considered a ‘proposal’, since it can be contested by other agents through negotiation. e.g., “I agree with your proposal for design-value x”; “I don’t agree with your proposal for design-value x”; “Give me another/all possible proposals(s)”; “Here is another proposal / are all other proposals”.

4.2.4 Conflict Detection and Notification

The agents themselves are responsible for detecting conflicts. *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. *Value-Usage Conflicts,* that indicate the inability of an agent to process the given information, can be detected through a failure indication from a constraint object when an agent checks its preconditions before execution. *Value-Criticism Conflicts* can also be detected through constraint failure.

4.2.5 Conflict Resolution

After posting the conflict information in the central database, the initiating agent starts the conflict resolution. How exactly the agent solves its conflict, what questions it asks during negotiation and what messages it sends, is up to the developer of the agent, as long as the agent uses the SINE language for the communication. It is not necessary that all agents have the same conflict resolution knowledge.

One of the methods we have implemented is to solve the problem by having the initiating agent iterate through a set of queries, in order to determine the current state of its conflict partner. The agent then proceeds by requesting the other agent to produce a new value or, if there is strong indication that this is the only value that the partner can produce, tries to adapt to the given value itself.

The agents are also responsible for recognizing conflict resolution or that no solution is possible. In most cases, one of the two agents will eventually make a proposal that the other agent agrees to. If no satisfactory proposal can be made from either side, the conflict cannot be solved. In that case, another design parameter might have to be revised in order to relax some constraints or a different strategy or plan might have to be used. Postponing the conflict resolution may also help, as design parameters may get revised in the meantime. More details about conflict resolution will be presented in a future paper.

4.2.6 Agent Scheduling and Control Flow

The SINE platform has a scheduling mechanism which maintains a list of agents that should be run next. The agenda manager is the top level controller of the system. An entry is proposed by an agent sending a message to the agenda manager. When no agent proposes an entry, the agenda manager assumes that the design is either finished or stuck unrecoverably, and it terminates, returning control to the user. In the case of a conflict, the agenda system is inactive. Instead, the communication mechanism that passes the information back and forth between the agents ensures that each agent receives control to be able to evaluate and react to the message.

4.3 Implementation

The SINE system was developed using GNU C and the CLIPS (C Language Integrated Production System) language [Giarratano & Riley 1994]. CLIPS is a forward chaining rule-based inference engine. For this project, we used the UNIX–X/Windows version of CLIPS. Development was done on a Digital Equipment Corporation Alpha™ 3000-300.

5.0 Evaluation and Future Work

From the analysis of conflict occurrences, conflict types, and the conflict resolution strategies, as well as from the experiments carried out so far, we have gathered some feedback about the SINE platform and about the use of single function agents. The experience helped us to single out some of the promising directions to pursue with the single function agent paradigm. At present the use of SiFAs has been restricted to several experiments of limited size. Work has focussed on discovering conflicts and conflict types.

5.1 Evaluation

The evaluation of SiFAs is proceeding in two directions, conceptual and practical. Evaluation of the usefulness of the SiFA concept as a research tool is being revealed by its ability to focus our attention on precise differences between types of conflicts. The results reported here show the richness of the analysis it makes possible, and new types of conflicts are still being discovered. At present a similar analysis of the impact of conflict types on negotiation remains to be done.

From a practical point of view, the SINE platform has been used to implement some of the functionality of the I3D+ system [Dunskus 1994]. The design of the system was much simpler, more flexible and maintainable than the previous implementation in plain CLIPS. The overall system functionality was the same. However, conflict resolution traces showed a different path due to the new mechanisms implemented. I3D+ used global and local goals to decide which agents should win in a conflict. SINE is situation driven, agents try to solve conflict situations by compromising or adapting as well as possible. Negotiation depends on the way agents are designed to interact and negotiate rather than on some rigid global system.

Another implementation used the platform to produce advice to a sailboat designer. This system required a larger number of agents, points–of–view and targets. The solutions generated by using the SINE platform were good although the design problem was of a less structured type. This provided a strong argument for using conflict resolution as a means to adapt to such problems.

The more flexible negotiation, in comparison to ID3+, led to better design decisions. Test runs in which vital information was removed demonstrated the flexibility of the agents in situations with underspecified data. SINE can therefore reduce some of the brittleness of design expert systems.

As part of the analysis of the results the performance of the platform was evaluated by a design expert. The factors primarily analyzed were the paths to achieve solutions and the outputs of the system. The main criteria were the relevance of the results and the explanatory power of the steps in achieving a solution. The expert's feedback was very positive. Their expertise was useful in determining the dependency of the conflict resolution process on the agent's knowledge and on the type of agents involved.

Overall, developing SINE and implementing design expert systems with the platform has shown that SiFAs are easy to use and quick to design systems with. The small granularity of the agents 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. It remains to be seen whether these benefits scale up.

5.2 Future Work

Although a working platform is available and a considerable amount of research has been done in the area of SiFAs, there are still a large number of topics that can be further investigated as well as functionality that waits to be incorporated or improved. The topics mentioned below are being investigated as part of the most recent research efforts on the SIFA model.

5.2.1 Conflict Resolution Hierarchy

A hierarchy of conflict situations that can occur in SiFAs is being built. The nodes high in the hierarchy denote very general types of conflicts such as incompatible suggestions, and nodes lower in the hierarchy are for more specific conflicts such as a conflict between two selectors suggesting different values for the same parameter. The conflicts represented in the leaf nodes are the most specific ones. The paths from all the leaf nodes where an agent function appears to the root of the hierarchy specify the part of the hierarchy that agents with that function need to know about. So each agent needs to have limited conflict resolution knowledge depending on its function.

This hierarchy is also enriched by attaching negotiation strategies to the nodes. A strategy is defined by the information exchange between participants in the conflict. In higher level conflict types, the messages on which this exchange is based can contain variables, which have to be instantiated. The values are assigned to these variables depending on the classification of the conflict in the lower level nodes of the hierarchy. In some situations a variable may be an entire message that needs to be instantiated according to the current context.

5.2.2 Grouping Knowledge Around Parameters

Building a conflict resolution hierarchy requires studying conflict types in more detail. To do this we need to know what pieces of knowledge that are generated by the system may potentially be conflicting. This requires that the values, estimates, evaluations, criticisms, and praises that refer to each parameter be explicitly represented as a group of entities around the parameter. Agents will be allowed to express their opinion not only about the design parameter value itself, but also about the knowledge entities stored with it. The targets of the agents will be extended from the parameter values to these entities. It will be thus possible to criticize an evaluation, or to praise a criticism.

5.2.3 Learning with Single Function Agents

Learning is one of the directions that we considered important to explore in conjunction with the SiFA paradigm. Among the several possible approaches, we chose to focus on learning resulting from the negotiation process.

There are several important potential benefits of learning during negotiation:

- The negotiation skills and strategies of the agents will be improved.
- Negotiation is a resource consuming process and learning can help reduce the amount of resources needed.
- Learning is expected to lead to a reduced number of conflicts.
- A more efficient negotiation process may contribute to an improved design solution.
- Knowledge acquired during negotiation can help increase the functional capability of each agent.

We hope that single function agents will help explore some learning mechanisms at a reduced scale and to single out those aspects which prove to be relevant. At this moment, with work in progress, the most promising approaches are the following.

Learning conflict patterns: Agents are supposed to learn the situations which frequently lead to conflicting situations and the types of conflicts associated with them. Keeping design histories would be one way to achieve this, where situation analysis is done in a case-based manner. However, by defining a set of attributes to characterize conflicts, it would be possible to take an inductive classification approach [Gennari et al. 1989]. The knowledge acquired helps anticipate conflicts and base proposals on expectations.

Learning negotiation plans: Keeping negotiation histories represents an opportunity to generate plans from previous negotiation cases. This can be done by compiling negotiation sequences using an explanation based approach [Mitchell et al. 1986]. We expect to find out how much these plans are dependent on the type of agents involved and on the type of conflict to which the plan responds.

Building agent models: Developing a model of the negotiation partner is crucial for performing negotiation. The issue which becomes relevant in the single function agent context is the dependency of the agent model on the type of agent. Agent specificity should lead to the formation and use of expectations, and to the tuning of the negotiation depending on the negotiation partner.

There are two main questions guiding our learning research, and we expect that any results in this area would help considerably to show how agents can improve their cooperation skills: What factors are dependent on the agent type in the learning process? Which types of learning have the major impact on the cooperation between agents and on the resulting design?

5.2.4 Agent Grouping/Hierarchy

For systems with a large number of parameters, some grouping of agents into subsystems that are responsible for certain design areas might be useful. These groups may be formed by grouping together agents that have a significant amount of interaction. Once we have a model of negotiation between agents, it will be possible to figure out which agents should be grouped together by measuring the interaction between agents, for example in terms of exchanged messages. Meta-agents, such as managers, could help administrate the activation of the groups and the negotiation between groups.

5.2.5 History Keeping and Using Design Rationale

As agents make decisions and communicate, they provide knowledge about themselves and gain knowledge about others. Design rationale can be used for design revision and negotiation. Associating rationale to decisions will imply the communication of richer information among agents. Design rationale will enhance both design and negotiation histories. Besides allowing for a more informed response to an agent's decision, it also represents a relevant opportunity for learning.

5.3 Conclusion

The SiFA approach to investigating negotiating multi-agent design systems offers a new way of studying the ingredients of such systems. It provides ways of discovering and testing the components of negotiation, patterns of communication, functional primitives of design, and the types of knowledge needed. By adopting a methodology that restricts 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.

6.0 References

Bond & Gasser 1988

A. H. Bond and L. Gasser (eds.) *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann Publishers Inc., San Mateo, California, 1988.

Brown 1992 a

D. C. Brown. “Design”. *Encyclopedia of AI*, 2nd edition, S. Shapiro (ed.), J. Wiley, New York, NY, 1992.

Brown 1992 b

D. C. Brown. “The Reusability of DSPL Systems”. *Preprints of the Workshop on Reusable Design Systems: Understanding the Role of Knowledge. Second International Conference on Artificial Intelligence 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.

Bussmann & Mueller 1993

S. Bussmann & H. J. Mueller. Bargaining Agents. *Progress in Artificial Intelligence (EPIA93)*, Filgueiras & Damas (eds.), Springer Verlag Lecture Notes in AI, Vol. 727.

Davis & Smith 1981

R. Davis & R. G. Smith. “Negotiation as a Metaphor for Distributed Problem Solving”, *Artificial Intelligence 20*, pp. 63 - 100, 1983. North Holland. Reprinted in *Readings in Distributed Artificial Intelligence*, (eds.) A. H. Bond & L. Gasser, Morgan Kaufmann Publishers, Inc. San Mateo, California, 1988.

Douglas et al. 1993

R. E. Douglas Jr., D. C. Brown & D. C. Zenger. “A Concurrent Engineering Demonstration & Training System for Engineers and Managers”. *International Journal of CAD/CAM and Computer Graphics,* special issue on “AI and Computer Graphics”, (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.* Masters Thesis, Worcester Polytechnic Institute, 1994.

Finin et al. 1993

T. Finin, J. Weber, G. Wiederhold, M. Genesereth, R. Fritzon, D. McKay, J. McGuire, R. Pelavin, S. Shapiro & C. Beck. *DRAFT Specification of the KQML Agent-Communication Language.* The DARPA Knowledge Sharing Initiative External Interfaces Working Group, 1993. (More information can be received via ftp from ksl.stanford.edu).

Gennari et al. 1989

J.H. Gennari, P. Langley & D. Fisher. “Models of Incremental Concept Formation”, *Artificial Intelligence,* Vol. 40, No. 1-3, 1989, pp.11-61.

Giarratano & Riley 1994

J. C. Giarratano & G. Riley. *Expert Systems: Principles and Programming,* 2nd ed. PWS Publishing CO, Boston, MA.1994.

Huhns 1987

M. N. Huhns (ed.). *Distributed Artificial Intelligence Vol. 1*, Pitman Publishing, London, and Morgan Kaufmann Publishers, San Mateo, CA, 1987.

Huhns & Gasser 1989

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

Klein 1991

M. Klein. “Supporting Conflict Resolution in Cooperative Design Systems” in *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21, No. 6, November/December 1991.

Kuokka et al. 1993

D. R. Kuokka, J. G. McGuire, R. N. Pelavin, H. C. Weber. H. M Tenenbaum, T. Gruber and G. Olsen. “SHADE: Technology for Knowledge-Based Collaborative Engineering” in *AAAI Workshop on Collaborative Design 1993*, as well as *Concurrent Engineering: Research and Applications*, Vol. 1, 1993, pp. 137-146.

Lander & Lesser 1991

S. E. Lander & V. R. Lesser. “Customizing Distributed Search Among Agents with Heterogeneous Knowledge” in *Proceedings of the 5th International Symposium on AI Applications in Manufacturing & Robotics*, Cancun, Mexico. Dec 1991.

Mitchell et al. 1986

T. M. Mitchell, R. Keller & S. Kedar-Cabelli. “Explanation-Based Generalization”, *Machine Learning,* Vol. 1, No. 1, 1986, pp.47-80.

Searle 1965

J. Searle. “What is a Speech Act”, *Philosophy in America.* Max Black (ed.), George Allen & Unwin Ltd., London, 1965. Reprinted in *Readings in the Philosophy of Language.* J. F. Rosenberg & C. Travis (eds.), Prentice-Hall Inc., New Jersey, 1971.

Sycara 1990

K. P. Sycara. “Cooperative Negotiation in Concurrent Engineering Design”, *Cooperative Engineering Design,* Springer Verlag Publications, 1990.

Taleb-Bendiab & Oh 1993

A. Taleb-Bendiab & V. Oh. “Speech-Act based Communication Protocol to support Multi-Agent Cooperative Design Systems”. *Proceedings of the 1993 AI in Engineering Conference*, p. 107, Computational Mechanics Inc., 1993.

Victor 1993

S. K. Victor. *Negotiation Between Distributed Agents in a Concurrent Engineering System.* Masters Thesis, Worcester Polytechnic Institute, 1993.

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”, *International Conference on Artificial Engineering,* Toulouse, France, July 1993.

Werkman & Barone 1991

K. J. Werkman & M. Barone. “Evaluating Alternative Connection Designs Through Multiagent Negotiation”, *Computer Aided Cooperative Product Development*, D. Sriram, R. Logcher, S. Fukuda (eds.), Lecture Notes Series, No. 492, Springer Verlag, 1992, pp. 298-333.