Conflicts and Negotiation in Single Function Agent Based Design Systems

Ilan Berker and David C. Brown
Computer Science Department
Worcester Polytechnic Institute
Worcester, MA 01906
berker@cs.wpi.edu, dcb@cs.wpi.edu
(508) 831-5618

January 1996

Abstract

A basic ingredient of Concurrent Engineering (CE) is that all aspects of the product's life-cycle should be considered as early as possible during design. A key idea that has been adopted for the implementation of CE is the use of a team, in which each member is an expert on some aspect of the product in a phase of its life-cycle. In our work, we use a team of software agents. Our goal is to study in detail the interactions, conflicts and conflict resolution that is possible with such a multi-agent system. The approach is to use Single Function Agents (SiFAs), each of which can perform a very specialized task, from a single point of view. The ability to represent expertise with different points of view is very important in design. These different points of view usually cause conflicts among agents, and these conflicts need to be resolved in order for the design process to be successful. Therefore, agents need to be capable of detecting and resolving these conflicts. This paper presents a model of conflicts and negotiations in the SiFA framework. A hierarchy of possible conflicts is proposed and the steps of the negotiation process are discussed.

Short Title: Conflicts and Negotiation in SiFA-based Design Systems

Keywords: Conflict, Negotiation, SiFA, Parametric, Critics, Selection, Design

1 Introduction

A basic ingredient of Concurrent Engineering is that all aspects of the product's life-cycle should be considered as early as possible during design. For example, both Packaging and Recycling should be considered at product design time. This idea was developed in response to a "linear" approach that considered design, then manufacturing, then assembly, etcetera. This approach delays the discovery of the possibly negative impact that design decisions may have on the later phases of the life-cycle.

By considering these later phases of the life-cycle during the design process, it is possible to quickly discover and compensate for problems that might have occurred at a later time in a linear approach. This converts problem discovery from a linear pattern to a more parallel, or "concurrent" one, thus saving time and effort.

In order for early consideration of later stages of the life cycle to be given at design time, it is necessary for a great deal of knowledge to be available during the design process. A wide variety of types of expertise are required to cover all the phases. Consequently, a key idea for the implementation of Concurrent Engineering is the use of a team, in which each member is an expert on some aspect of the product in some phase of its life-cycle. Members of such a team will have different goals and different knowledge, and varying degrees of specialization.

1.1 Single Function Agents

Our work uses software agents instead of human agents. There is not intended to be a 1-1 correspondence between software agents and human agents. If any correspondence were to exist it would be that a group of software agents correspond to the reasoning of a single human agent.

Our goal is to study *in detail* the interactions, conflicts and conflict resolutions that are *possible* with such a multi-agent system. This work is not meant to literally simulate a real CE team, but rather to try to discover what is at the *core* of a team's decision-making. We are concerned with the "primitives" of knowledge and reasoning for design, redesign, and conflict resolution.

Other work on multi-agent systems that do design has tended to use large agents with powerful capabilities and large quantities of knowledge (e.g., for example [12]). With so much power and knowledge it is hard to discover what is *essential* in the system. Although research with large agents is very valuable, it has other goals.

In our approach we have stripped down each agent to an extremely simple form, in order to more closely study the behavior of the team, the functionality required, the interactions that are possible and necessary, the amount and types of knowledge needed, and the conflicts that occur. In this way we can build a better understanding of design agent conflict management.

Taking the multi-agent paradigm to such an extreme results in Single Function Agents (SiFAs), where each agent performs a single function in the design process and therefore contains knowledge that is very specialized. As with a human team, each SiFA will have different goals and different knowledge. As each SiFA does far less than a normal team member, many more agents are required. Current SiFA research focuses on parametric design.

SIFAs are not the same as the large-grained agents in systems that try to incorporate legacy systems, such as constraint solvers, or data-base retrieval engines [10], as although those have restricted functionality they are usually very general, and may not be knowledge-based.

1.2 Conflicts and Negotiation

The use of agents brings with it the possibility of conflicts. Although it may be possible to build a system where all conflicts are resolved during development time, in most design problems this is either impossible or extremely difficult.

There are many advantages of run time conflict resolution. Such interaction supports concurrent engineering. Conflicts and their resolution cause the behavior of the system to emerge from the requirements of the particular problem at hand. The system is more flexible and does not suffer from the brittleness problem of traditional expert systems.

One way of resolving run time conflicts is for the agents to negotiate. This is the most general and flexible way of resolving conflicts. With SiFAs, many more conflicts become visible, due to the small size of each agent — conflicts can't appear and be resolved within an agent. This is very important if we wish to understand the essence of conflict management.

1.3 COSINE: A Wine Glass Designer

One goal of this research was to build a single function agent based design system to demonstrate the capabilities and restrictions of negotiation in the SiFA paradigm. Consequently, a simple wine glass designer called COSINE was implemented. The wine glass domain is simple enough to enable building a prototype in a short amount of time, yet it is complicated enough to demonstrate all the power of a SiFA system plus the richness of the possible conflict situations. It is important to note that the wine glass domain and the COSINE system are for purely illustrative purposes. Other SIFA-based systems have been (and are being) built [6].

At present, each SiFA acts on a single parameter. In COSINE, the parameters of the design are cup radius, cup thickness, stem length, stem radius, base radius, and base thickness. There are many constraints among these parameters. For example, the ratio between the cup radius and the base radius has to be within certain limits so that the cup is stable. Also, the cup radius has to be within certain limits so that it holds a reasonable amount of liquid. Similar constraints exist involving other parameters.

2 Selected Previous Work

2.1 Agents and Multi-Agent Systems

In recent years there has been increased interest in agent based systems. The concept of an agent has become important both in Artificial Intelligence (AI) and in other fields of Computer Science [20]. Agents also have a direct influence on Distributed Artificial Intelligence (DAI) research because of their potentially concurrent nature [1]. Many difficult tasks, including design, have parallel decompositions that result in easier subtasks than serial decompositions. Parallel decompositions allow opportunistic collaboration among the subtasks.

2.2 Conflict Resolution

Klein [7] suggests a model of conflict resolution having a hierarchy of conflicts with the most abstract conflicts at the top and most concrete conflicts at the leaves. There is a corresponding hierarchy of resolution strategies, resulting in domain dependent and domain independent conflicts and their associated resolution strategies. The nodes higher in the hierarchy represent the domain independent conflicts while the lower nodes become more domain dependent.

Klein's agents all have their own design knowledge and conflict resolution knowledge [8] [9]. The agents have differing design expertise, but their conflict resolution knowledge is identical.

Other recent work on Conflict Management can be found in an AI EDAM special issue [14].

2.3 Negotiation

Negotiation is a common approach to conflict resolution in the design domain. Sycara defines negotiation to be the process by which resolution of inconsistencies is achieved in order to arrive at a coherent set of design decisions [16] [15]. The negotiation process proceeds with generation of a proposal, generation of a counter proposal based on feedback from dissenting agents, and then communication of justifications and supporting evidence [16].

In Sycara's model there are four conflict situations where negotiation is used. These are where different agents make conflicting recommendations for a parameter value, a value proposed by one agent makes it impossible for another agent to offer consistent values for other attributes, a decision of one agent adversely affects optimality of other agents, and where alternate approaches achieve similar functional results.

Lander and Lesser [12] use the negotiated search paradigm for conflict resolution among heterogeneous and reusable expert agents in their TEAM framework. This paradigm allows agents to be both logically and implementationally heterogeneous.

Werkman's Designer Fabricator Interpreter (DFI) is a system where agents with different points of view cooperatively evaluate different suggestions for a design parameter [19]. Unlike

SiFA systems, DFI uses an arbitrator as a means of central control through which agents communicate.

Polat et al [13] describe a problem-solving environment that supports multi-agent conflict detection and resolution. They include a flowchart that describes the general conflict resolution process between agents. This has many similarities with our approach, discussed in section 6.6.

2.4 Single Function Agents

Although there has been a lot of work done on multi-agent systems, Single Function Agents (SiFAs) are a relatively new way of building multi-agent systems. So far, three systems have been developed.

The first, I3D [17], was a system that integrated part design and manufacturing plan production for Powder Processing applications. The agents were carefully sequenced, and all possible conflicts that might occur were anticipated in advance and removed during development of the system.

The second system built was I3D+ [18]. It had agenda-based scheduling of the agents and allowed conflicts about the values of parameters to occur among agents. The conflicts were classified into six types depending on the relation between agents' local goals and the global goal.

SNEAKERS [3] was built to train users in Concurrent Engineering. The user interacted with agents that had different functions and points of view.

Based on these systems, SINE was developed as a platform to build multi-agent design systems using SiFAs [2]. It was possible to simulate the negotiation behavior of I3D+ using SINE.

3 The Single Function Agent Paradigm

3.1 What is a SiFA?

A SiFA is an agent that performs a single *function* on a single *target* from a single *point of view* [4].

The function performed by a SiFA determines its type. At present only a limited number of functions have been used for design problems. We conjecture that a set of agents with these functions is sufficient for most design problem solving activities, although we admit that this is far from clear for more conceptual design activities.

These agents types are:

Selector: Selects a value for a parameter by picking a value from a list of possible values. These are usually suitable for discrete valued parameters.

Advisor: Produces a value for a parameter by some means other than picking a value from a list. Advisors are more suitable for real valued parameters. Advisors and selectors are the agents where most of the design knowledge is stored.

Estimator: Produces estimates of values for a parameter. Unlike selectors, estimators can work with insufficient information, so the values they produce are just estimates of what the final value should be.

Evaluator: Evaluates the value of a parameter, producing a value of goodness for that value, usually represented as a percentage.

Critic: Criticizes values of parameters by pointing out constraints or quality requirements that are not met by the current values.

Praiser: Praises values of parameters by pointing out why the values are desirable.

Suggestor: Suggests what to do to remove an existing conflict or to avoid a conflict altogether.

The target of a SiFA is a single parameter of the design. In the case of wine glass design, the targets include cup radius, and stem length.

The point of view of an agent is some aspect of the design that the agent considers while doing its work. Usually, the point of view of the agent is a goal that the agent is trying to satisfy or optimize. Examples of points of view for agents in wine glass design are cost (as in all design, the cost should be minimized), style (the glass should look nice), stability (the cup should not fall over because of a very large cup and a small base), and volume (the cup should hold a reasonable amount of liquid).

In the SiFA paradigm, an agent is characterized by a single function, a single target, and a single point of view. In contrast to agents that perform general tasks, such as an equation solver, SiFAs are very specialized knowledge based agents. A naming convention for individual SiFAs is to specify the target first, then the point of view and lastly, the function. So a selector whose target is the cup radius and is trying to maximize stability of the cup is a *cup radius stability selector*.

SiFAs implement the basis of Concurrent Engineering, i.e., early consideration of aspects of the product's life-cycle, by allowing agents, such as selectors and critics, to represent the points of view of these aspects at any decision making point in the design. Hence a decision about a length might be made from the points of view of packaging and of maintainance, and criticized from the point of view of cost.

3.2 Communication Between SiFAs

Analogous to team members talking to each other, SiFAs communicate by sending each other messages. Each agent can communicate directly with any other agent without the need for an arbitrator. The communication language used is based on KQML (Knowledge Query and Manipulation Language) [11]. Even though agents exchange messages directly, the current state of the design is stored centrally and is accessible to all agents in the system.

3.3 The Control Mechanism

The SIFA research makes very little commitment to the system architecture to be used, as this is not currently the primary focus of the work. However, for current implementations an agenda mechanism is used to schedule agents. Agents are polled by the agenda and each agent indicates what it wants to do. Based on the replies, the agenda orders the agents according to the importance of the tasks they will carry out; e.g., conflict detection and resolution is more important than doing design. Then the agenda gives control to that agent. When the agent's work is finished, the cycle is repeated.

4 Extensions to the SiFA Paradigm

4.1 The Parameter Block

In previous work with SiFAs, the critiques, praises, or evaluations they produced were not stored separately. This prevented some potential, realistic actions of agents. This work extends the SiFA model to represent all value, estimate, criticism, praise, and evaluation *entities* separately, as *first class objects*. Being a first class object means that critiques, praises, and evaluations have the same status as the value of a design parameter. They are all directly accessible and they can all be the target of an agent. This is done by having value, estimate, criticism, praise, and evaluation entities organized in a structure called the *parameter block*, shown in figure 1. There is a block for each parameter. These blocks are stored centrally.

* * * FIGURE The parameter block

The root of the parameter block is the name of a parameter. The first *level of reference* has two entities. These are the parameter's value and estimated value. In the current model there can be only one value and one estimate at any time.

The second level of reference has evaluations, criticisms, and praises of the value and the estimate. These entities are said to reference the value or the estimate. There can be multiple criticisms, praises, and evaluations of the same value or estimate.

The third level of reference has evaluations, criticisms, and praises of second level evaluations, criticisms, and praises. These entities refer to the second level entities which refer to the first level entities. These *chains of reference* uniquely determine what each entity refers to. So at the third level, there exist entities such as the evaluation of the criticism of the value of the parameter. The third level entities typically contain meta-level information about the design. Once again, there can be multiple evaluations, criticisms, and praises of the same second level entity.

It is also possible to imagine this structure growing into fourth, or even fifth level references. Although what the contents of such entities might be is not immediately obvious, the model allows such information to be represented if it is meaningful in any design problem.

It is clear that any complete model of the formation, detection and resolution of conflicts between agents with different points of view needs to be able to represent and reason about the information in at least the first three levels of the parameter block.

4.2 Targets of Agents

When all entities are represented as first class objects, they can all be the target of an agent. This means that the target of a critic is no longer just the name of a parameter, but more specifically, it is a criticism entity, such as the criticism of the evaluation of the estimate. Each agent can act on, or modify, the contents of its target. So a critic can store a criticism in the critique entity that is its target (Figure 2), and an evaluator can store an evaluation in the evaluation entity that is its target (not shown).

* * * FIGURE Estimate evaluation critics

There can be more than one criticism of the evaluation entity, from different points of view. In that case there would be one agent for each of these criticisms, shown as Critic1 and Critic2 in figure 2. These critic agents would have different points of view and different targets, as they are acting on two different entities. It is also possible for two critics to have different points of view but the same target as is the case with Critic2 and Critic3, if it is not necessary to store both criticisms produced.

Only one value and one estimate are possible at any time in a parameter block, unlike evaluations, criticisms, and praises. There may be more than one selector or estimator agent, with different points of view, that have the same estimate or value as their target. This is also shown in figure 2. Note that this may lead to a conflict.

4.3 Knowledge in a SiFA

The knowledge that has to be present in a SiFA is determined by the tasks the agent has to carry out (shown in figure 3). Each agent has to have knowledge about how to perform these tasks.

* * * FIGURE The knowledge contained in a SiFA

The types of knowledge can be roughly divided into three categories. The first one contains design and redesign knowledge. The design knowledge allows the agent to carry out its main function which may be one of selection, estimation, evaluation, criticism or praise. The redesign knowledge is used when the agent has to produce a value, estimate, evaluation, criticism or praise after it has already produced one. This is necessary when the first entity produced causes a conflict in the system and the agent is asked for another during negotiation. This situation is explained in section 6.

The second category contains conflict indication, detection, and classification knowledge that is used to carry out the respective tasks of conflict handling. These typically involve checking if some constraints which are internal to the agent, referred to as *internal constraints* are violated or not. These tasks are explained in detail in section 6. The knowledge in this category also includes the knowledge of the types of conflicts in which the particular agent can be involved.

The third category consists of negotiation strategy selection, refinement, and execution knowledge. This category of knowledge enables the agent to negotiate with other agents in conflict situations. Negotiation strategies and the associated tasks are also explained in section 6.

5 Hierarchy of Conflicts

The conflict hierarchy for SiFAs is shown in figure 4. The nodes represent conflict types. Each node is a specialization of its parent node and therefore inherits all of its properties. The root node, labeled *conflict*, encompasses all conflict types. Since advisors and selectors are involved in similar conflicts, we use the term *selector* to refer to both types in this hierarchy.

* * * FIGURE The SiFA conflict hierarchy

It would be possible to change the order of the criteria used at each level to specialize the conflicts. However, the hierarchy presented here does the best job of grouping common aspects of conflicts into a single, more general node. The hierarchy is structured mainly by the way the conflicts manifest themselves, since in the SiFA paradigm this also provides information about the underlying cause of the conflict. The following sections describe the hierarchy of conflicts in detail.

5.1 Criticizing an Entity

This type of conflict occurs when a critic is not satisfied with the entity it is watching over. It produces a criticism and initiates a conflict. The entity being criticized is the one that is the critic's target. This situation is shown in figure 5. These conflicts are referred to as *criticism conflicts.*

* * * FIGURE Criticizing an entity

For this type of conflict a value, estimate, evaluation, criticism, or praise entity may be criticized. The next level of specialization of the hierarchy differentiates conflicts into three based on the type of entity being criticized.

5.1.1 Value/Estimate Criticism

In this group of conflicts, the entity being criticized is a value or an estimate. There are similarities between these two entities because there can be at most one value or estimate of a parameter at any time. Currently, the criticism of a value or estimate may be one of *too high*, *too low*, *wrong units* or *too imprecise*.

This group can further be broken down into two subgroups where the criticized entity is a value, and where it is an estimate, resulting in conflicts between a critic and a selector, and between a critic and an estimator, respectively. Both types of conflict are initiated by the critic.

5.1.2 Evaluation Criticism

An evaluation may be criticized because it is *too imprecise*, *too high*, or *too low*. All conflicts of this type are between an evaluator and a critic. The conflict is initiated by the critic.

5.1.3 Critique/Praise Criticism

Just as for any other entity, it is possible to have criticisms of either criticism or praise. Such a criticism may be one of *too negative*, *too positive*, *too imprecise*, or *not substantiated*. SiFAs are probably unique in their ability to express and act on this sort of meta-level knowledge. Such conflict-based interactions provide information that may enable learning to take place in the CE team. This is an area of current research [6].

This conflict type may be divided into two groups, criticizing a critique, and criticizing a praise, resulting in conflicts between two critics, and between a critic and a praiser respectively.

5.2 Incompatible Suggestions

If there are two or more agents with the same target, or if there is a relationship between the targets of two agents, then there is a possibility for conflict among these agents. If the results of these agents are not compatible in some way, there is conflict. These conflicts are also referred to as *incompatibility conflicts*.

Similar to the left hand side of the hierarchy, the next level of specialization is based on the entity that is the subject of the conflict. Here, the subject of the conflict is not what the criticism refers to, but it is the target of the agents suggesting incompatible values, estimates, evaluations, etc. Analogous to the right side of the hierarchy, the conflicts are divided into three groups.

5.2.1 Value/Estimate Incompatibility

Incompatible suggestions for values and estimates can occur in two ways. The first is when there are two agents with the same target and the suggested values, or estimates, are incompatible. This will cause a conflict because only one value (or estimate) may be stored in the target entity. This situation is shown in figure 6. Both agents have to be the same type, selector or estimator. So the conflict is between two selectors or two evaluators. The points of view of the conflicting agents have to be different.

* * * FIGURE Incompatible suggestions for the same value/estimate entity

The second kind of conflict can occur between two entities across parameters. This happens when the value of one parameter is not compatible with the value of the other parameter as shown in figure 7. The two value entities have to be in different parameter blocks since each parameter can have only one value entity. This type of conflict is possible only between two selectors. As a simple example, consider the case in the wine glass design where the a base radius selector generates a value for the base radius and then checks the internal constraint that the base radius has to be greater than the stem radius. If this constraint is violated, then the base radius selector will be in conflict with the stem radius selector.

* * * FIGURE Incompatible suggestions for different value entities

5.2.2 Evaluation Incompatibility

Incompatible suggestions for an evaluation entity occurs when two evaluators with different points of view have the same evaluation entity as their target, as shown in figure 8. This kind of conflict will occur only between two evaluators.

* * * FIGURE Incompatible suggestions for an evaluation

5.3 Critique/Praise Incompatibility

Two kinds of conflicts are possible in this group. One kind is when two critics, or two praisers, with different points of view have the same target. This situation, shown in figure 9, is very similar to incompatible evaluations. Both agents have to be the same type, so the conflict occurs between two critics or two praisers.

* * * FIGURE Incompatible suggestions for the same critique/praise entity

The other kind happens when there is an incompatibility between a criticism and a praise of the same entity. In this situation, shown in figure 10, the targets of the agents in conflict are different but the targets refer to the same entity. In other words, the criticism and the praise of an entity are not compatible. This conflict occurs between a critic and a praiser and can be initiated by either of them. The agents in conflict have to have targets in the same parameter block at the same level of reference, since the incompatible critic and praise refer to the same entity. This conflict is not meaningful across parameters.

* * * FIGURE Incompatible suggestions for different critique/praise entities

5.4 Domain Independence

Although this hierarchy is very fine grained, it is highly domain independent. There isn't anything that is particular to any domain such as mechanical or electrical design in the hierarchy. It applies equally to all design problems that can be solved by SiFAs. This property allows us to investigate what type of knowledge is needed for each type of conflict, to form a more precise picture than already exists for multi-agent systems in a CE setting.

The hierarchy could be expanded for another level from the leaf nodes by specializing based on the points of view of agents participating in the conflict. If some points of view can be grouped together into classes that are present in all domains, such as manufacturability, durability, and cost, then the additional level of the hierarchy would again be domain independent. Any further specialization, for example, one that involves parameters, such as stem length in the wine glass design, would certainly be highly domain dependent.

The conflict taxonomy is structured by how the conflict occurs. An alternative is to structure conflict hierarchies by the fundamental underlying causes of the conflicts. Heuristic classification approaches derive much of their power from structuring the diagnostic taxonomies in this way. Currently we believe that SiFAs are fine-grained and specialized enough to be very close to the cause of each conflict. Interaction with the conflicting agent should refine the agents' knowledge of the cause. Further work is required to fully confirm this belief. This issue is important as a precise classification of each conflict should provide indexing for retrieval of an appropriate remedy for that conflict.

6 SiFA Negotiations

The system has to go through several steps during a conflict. These are Indication of a possible conflict, Detection of the conflict, Selection of a negotiation strategy to use, and Refinement of that strategy. The actual negotiation is simply the execution of the selected strategy. All of these steps put together is called the *negotiation process*. The result of the process is either a solution to the conflict or the signaling of a failure.

Some of the steps in the negotiation process are very trivial in the SiFA paradigm since the conflicts are very well defined and the number of different kinds of conflicts an agent can be in is very limited. All of the steps are explained in the following sections.

6.1 Conflict Indication

Agents need to be able to notice situations where there is the possibility of a conflict. This act of realizing a potential conflict is called *conflict indication.* Conflict indication is a very simple task for the agents in the SiFA paradigm. Each time an agent acts on its target by modifying it, the state of the overall system changes, therefore there is potential for a conflict. How a possible conflict is indicated depends on the general conflict categories to which it belongs.

6.1.1 Indication of a Criticism Conflict

Conflicts which are caused by the criticism of an entity, the ones on the left hand side of the conflict hierarchy, are noticed by the critic producing the criticism. Each time a critic produces a criticism, there is a conflict indication.

The production of a criticism may also cause an incompatibility conflict if the target entity holding the criticism is already storing a critique. This is discussed next.

6.1.2 Indication of an Incompatibility Conflict

Whenever an agent acts on its target by producing a value, estimate, evaluation, praise, or criticism, it checks to see if there is the possibility of a conflict.

For an agent to indicate a same entity conflict, all of the three conditions given below have to be satisfied:

- The target entity already contains a value, estimate, evaluation, critique, or praise,
- The owner of the entity (the agent who put in the old contents) is not itself (the agent acting on the target now), and
- The old content of the entity is not identical to the new one the agent is attempting to store in the entity.

If all these conditions are true, the agent notices the possibility of a conflict and "indicates" it. The indication of a conflict does not mean that there is necessarily going to be a conflict. This is the work of the conflict detection knowledge.

6.2 Conflict Detection

The agent that indicates a conflict is responsible for determining whether there actually is a conflict. Suppose there is a 0.01% difference in the values proposed by two different selectors for the value entity of a parameter. Unless that value is extremely sensitive, this small difference does not constitute a conflict. Thus indication of a conflict does not necessarily mean that there is actually a conflict.

Conflict detection, for human or SiFA, is a knowledge based task and it is not always trivial. How detection works in the SiFA paradigm depends on whether the conflict is a criticism conflict or an incompatibility conflict.

6.2.1 Detection of a Criticism Conflict

As soon as a critic produces a criticism it has to decide if that actually signals a conflict or not. This is usually an easy task since the critic knows the reason why it produced that particular criticism. If the criticism was due to the violation of a hard constraint that tests some physical properties or user requirements, then there is probably a conflict. On the other hand, if the criticism just reflects a preference, then there is no conflict. The criticism stays on the blackboard and the agents whose targets were criticized may take it into consideration if they wish.

6.2.2 Detection of an Incompatibility Conflict

Detecting an incompatibility conflict requires more knowledge and is usually a more complicated task than detecting criticism conflicts. There are three possible cases.

Same Entity Conflicts: Once the agent notices that what is already stored in its target entity is not identical to what it wants to store, and indicates a conflict, it has to decide if the two are compatible. If they are, then there is no conflict although they may not be identical.

For example, suppose a design parameter can take on continuous numeric values, and its current value has been set to 5.00 by selector A. If later, selector B calculates a value for the same parameter from a different point of view and comes up with 5.01, it indicates a possible conflict, but has to do more work in order to decide if there is indeed a conflict. Selector B needs to know how sensitive the value of the parameter is. This sensitivity also depends on selector B's point of view. If allowing up to 1% changes in the value of the parameter is acceptable by selector B, then there is no conflict. Selector B can leave the value as 5.00. Similar situations can occur with other agent types and entities.

For agents to be able to detect incompatibility conflicts, they need appropriate knowledge. Target entities may be numeric or symbolic. Value entities would be numeric for parameters such as length, but symbolic for parameters such as color or material. Estimates would generally be of the same type as the value of the parameter. Evaluations may be symbolic such as good, average, bad, or numeric if the quality is measured as a percentage. Critiques and praises may contain both numeric and symbolic information.

If the target entity is numeric, then the agent should have either a percentage or a fixed value by which the numeric value, estimate or evaluation it produces may be relaxed. For example, a selector wants the value to be 50, but can relax 10%, so anything between 45 and 55 is fine.

If the target entity is symbolic, there are two possibilities. If there is an ordering to the symbols, such as bad, average, good, then the agents can have a constraint such as "anything better than average is acceptable". If there is no ordering to the symbols than the agents need to have a list of symbols that they find acceptable.

Different Entities Within a Parameter Block: Detecting this kind of conflict is the same as detecting conflicts involving one entity as far as the knowledge based reasoning is concerned. The difference is that the agent is not comparing something that is already in its target with what it wants to store, but is comparing it to some other entity in the parameter block.

This means that agents need to know what entities have the potential to be incompatible with their targets. This is not a big problem since critics and praisers are the only kind of agents that can get into such conflicts because of incompatible criticisms and praises of the same entity.

Different Entities Across Parameter Blocks: This case is very similar to detection of conflict involving entities within a parameter block. The only difference is that only selectors can have such conflicts. This means that selectors need to know what values are related to the value they are producing and check for the constraints between these values.

6.3 Conflict Classification

After an agent detects a conflict, it has to decide what kind of conflict it is. This is a very simple task in the SiFA paradigm, as each agent can be involved in a very limited set of conflicts. Also, the detection process already gives the agent enough information to immediately classify the conflict as one of the leaf nodes of the conflict hierarchy.

If a critic detects a conflict after producing a criticism because some hard constraint is violated, then it knows that it is involved in a criticism conflict. It also knows if the conflict is about a value, estimate, evaluation, criticism or praise. The type of the agent with which it is in conflict follows immediately from the entity type — e.g., a value entity means that the other agent is a selector. Finally, it can easily figure out which particular agent it is in conflict with by inspecting the *owner* field of the entity that is the subject of the conflict. Similar reasoning allows the agents to deal with incompatibility conflicts as well.

6.4 Negotiation Strategy Selection

A *negotiation strategy* is a body of knowledge, usually representable by a set of rules, that allows the agent to carry out a negotiation. After a conflict has been detected and classified, the SiFA that detected the conflict has to start negotiating, as the *initiator*, with the conflicting agent, referred to as the *partner*. In order to do that, it has to select a negotiation strategy. The partner also has to select a strategy.

In the context of SiFAs, since the conflicts an agent can be involved in are very specific, the number of strategies an agent can have for any conflict are also limited. As the number of conflicts an agent can be involved in is also small, the total number of strategies it needs is small as well. Usually a SiFA has a single strategy to deal with a certain conflict, but more than one strategy for a single conflict type is also possible.

If an agent has a single negotiation strategy for the conflict it is in, then the strategy selection task is trivial. If the agent has more than one strategy for a particular conflict, then it needs a criteria to select its negotiation strategy. This criteria may be the point of view of its partner or the particular value, estimate, evaluation, criticism, or praise that is the the subject of the conflict. For example, it might choose its negotiation strategy by using a rule such as "if the difference is less than 10% use strategy 1 else use strategy 2".

The partner also needs to select a strategy. The partner can classify the conflict based on the first message it receives from the initiator. This message identifies the initiator, its target, and what the conflict is about. The partner is aware of its own target, so when it receives the first message in the negotiation, it is able to classify the conflict. Then it can select a negotiation strategy in a fashion similar to the initiator.

It is also possible for agents to switch from one negotiation mode to another during the execution of the negotiation strategy, based on the interaction with the other agent.

6.5 Negotiation Strategy Refinement

The negotiation strategy selected may still be a generalized strategy that has to be instantiated by giving values to some variables of the strategy. The main reason why an agent would have generalized strategies is to save space. The agent may need a few different strategies for a type of conflict, and if these strategies are very similar except for a few minor points, then it is better to have only one generalized strategy rather than storing all these similar strategies separately.

Another reason why a generalized strategy may be useful is if the agent does not have enough information to select a strategy before the negotiation starts. In this case, the agent can start negotiation using the generalized strategy and then instantiate it during the negotiation as more information becomes available.

6.6 Negotiation Strategy Execution

After conflict indication, detection, strategy selection and refinement, the agents are finally ready to negotiate. The negotiation consists of the initiator's and the partner's execution of their respective strategies. Examples of simple strategies, and their potential for supporting learning, are described in [6].

It is possible for these strategies to clash, so that the agents will not be able to reach an agreement. In such a case either the conflict will not be resolved, or one or both agents will change their strategies during the negotiation. The change of strategy is explained in section 6.7. At present we have not studied the case where no resolution is possible.

6.6.1 Negotiation Graphs

It is possible to represent a negotiation between two SiFAs using *negotiation graphs.* A negotiation graph shows all possible sequences of messages sent and the actions performed by the agents involved in a particular kind of conflict.

The start node of the graph shows the message sent by the initiator of the conflict. A path through a negotiation graph starting at the start node and ending at a node without any outgoing arcs is a full transcript of a possible negotiation. Whenever there is more than one outgoing arc from a node, this means that the negotiation can proceed in any one of those ways. Which one of those arcs is taken depends on the state of the particular agents involved in the negotiation.

Figure 11 shows a negotiation graph for a conflict between a critic and a selector. What this graph indicates is that the critic asks the selector for alternatives and the selector either provides more alternatives or asks the critic to relax its constraint. If the selector proposes an alternative value, the critic may still disagree, and if the critic relaxes its constraint, the relaxed constraint may still be violated. So the negotiation process is a cycle of proposals of alternatives and constraint relaxations. The negotiation either ends in success if the critic is satisfied with the value, or in failure if the critic cannot relax its constraint any more and the selector can give no other alternative values.

* * * FIGURE Selector-Critic negotiation graph

It should be noted that the negotiation graph does not give any information about the order of constraint relaxations by the critic or of the alternative proposals by the selector. This is dependent on the respective agents and the strategies they use. The selector may use one of the following strategies — *greedy*, *maximally cooperative*, or a mixture of the two. For example:

- When asked to supply an alternative value, decline and ask for a relaxation immediately,
- When asked to supply an alternative, give alternatives until there are no more possible alternatives, only then ask for a constraint relaxation,
- Supply alternatives and ask for relaxations in an interleaved manner, possibly giving one alternative and asking for a relaxation before giving another alternative.

Similarly, the critic may use one of these strategies:

- Wait for a "no more alternatives" message from the selector before relaxing a constraint,
- Relax a constraints before asking for alternatives,
- Relax constraints and ask for alternatives in an interleaved manner, possibly doing one relaxation and asking for one alternative before doing another relaxation.

Figure 12 shows another negotiation graph, this time for a conflict between two selectors with the same target. As for the selector-critic conflict, the selectors can ask each other for alternative values or constraint relaxations in any order that is dictated by their respective strategies. The left and right halves of the graph are identical except that the roles of selector1 and selector2 are reversed.

* * * FIGURE Selector-Selector negotiation graph

Negotiation graphs are helpful when building SiFA negotiation knowledge, since the graphs show all messages that the agent will need to respond to and all possible answers that it can give to a specific message, for each conflict type that it can be involved in. Since the conflict situations in the SiFA paradigm are very well specified, the negotiation graphs are fairly small and easy to build. The drawback of using negotiation graphs is that they get very complicated very quickly if we allow the agents to use a larger vocabulary so that they can talk about things other than alternatives and relaxations.

6.7 Changing Negotiation Modes

Agents have the ability to change their negotiation strategy during negotiation. What prompts an agent to change its mode of negotiation is the interaction with its partner or its own internal state.

An example of changing modes triggered by the interaction is when a selector changes its negotiation strategy depending on whether it receives a message asking for an alternative value or a message asking for a constraint relaxation.

An example of changing modes prompted by an agent's internal state is when a selector changes strategy when it runs out of alternatives. At that point it can go into a *don't care* mode meaning that it does not care what the value of the parameter is and the other agent can have any value it likes. This is possible if the agent's point of view is not very critical, such as style or colorfulness as opposed to strength or cost.

6.8 Emergent Behavior

The negotiation graphs do not fully determine how a negotiation will proceed since there are multiple paths through a negotiation graph. The particular path followed in a negotiation depends on the context in which the conflict occurred, the particular strategies of the involved agents, their internal knowledge, and their internal states. There is also the possibility of changing negotiation modes during negotiation.

This means that the the negotiation behavior of the agents cannot be predicted accurately in advance and the systems behavior as a whole will emerge at run-time as a function of the negotiation capabilities of the agents in the system. Furthermore, if each agent is built without a priori knowledge of the full negotiation graphs, then it is not possible to predict even the general structure of the negotiations before run-time. This is further complicated by the possibility of negotiation being affected by praise or critiques of the values being discussed (see [6]).

7 Evaluator-Critic Criticism Conflict Example

This is a conflict between the cup radius value evaluator from the volume point of view and its critic, as shown in figure 13. This example is taken from COSINE, the SiFA based wine glass designer.

* * * FIGURE Criticism of an evaluation

COSINE was developed using CLIPS (C Language Integrated Production System) version 6.02 which is an expert system building tool. Detailed information about CLIPS is available in [5]. The development platform for COSINE was a DEC Alpha 3000.

The agents implemented in the cup radius block in COSINE are shown in figure 14. Since the evaluation of the value is a second level entity, the criticism of that evaluation is a third level entity.

* * * FIGURE Cup radius parameter block in COSINE

Once there is a value for the cup radius, the preconditions of the evaluator are satisfied, so it provides the evaluation *good* from the volume point of view. The reason for this result is that the difference between the volume of the cup and the ideal value for volume according to this evaluator is less than a preset threshold.

The critic of the evaluation has a constraint which says that the evaluation should be more precise and numeric, not symbolic. As this constraint is violated, the critic produces a criticism and initiates a conflict. When the evaluator is asked for a more precise evaluation, it supplies an evaluation represented as a percentage of the optimal quality that can be achieved from its point of view. This percentage represents how close the current volume is to the ideal volume.

Here is the excerpt from COSINE:

```
Cup Radius Value Volume Evaluator: Evaluation asserted.
   Cup radius value is good from a volume pov.
Cup Radius Value Evaluation Precision Critic: Conflict detected
   with Cup Radius Value Volume Evaluator.
Cup Radius Value Evaluation Precision Critic: Criticism asserted.
   Cup radius value evaluation is too imprecise.
Cup Radius Value Evaluation Precision Critic: Ask Cup Radius
   Value Volume Evaluator to offer a more precise evaluation.
Cup Radius Value Volume Evaluator:
   Changing to precision evaluation mode.
Cup Radius Value Volume Evaluator: Evaluation asserted.
   Cup radius value's quality is 88 percent.
Cup Radius Value Evaluation Precision Critic: Criticism retracted.
   Cup radius value evaluation is not imprecise any more.
Cup Radius Value Evaluation Precision Critic: Conflict with Cup
   Radius Value Volume Evaluator resolved.
```

8 Evaluation

The goals of this work, and our achievement of them, can be demonstrated by the following questions and answers:

- *Have the conflicts, interactions and knowledge that underlie CE teams been isolated and investigated?*

  By using a set of primitive, design-oriented functions, and by allocating one primary function to each type of agent, the SiFA approach has allowed the discovery of a rich set of basic conflicts, as well as the knowledge required to handle them. Such an analysis could have only been done with the SiFA approach. However, much more work remains to be done.

- *Have all conflict types among all possible pairs of agents been identified and negotiation behaviors proposed for those conflicts?*

  The proposed conflict hierarchy, we believe, covers all meaningful SiFA conflicts. Although negotiation graphs for all conflict types have not been presented, COSINE has demonstrated all conflict types and their resolution with simple negotiation strategies. However, there are many more possible negotiation behaviors to be explored.

- *Do the classification of conflicts and the corresponding resolution strategies allow inheritance from abstract to concrete classes in the manner proposed by Klein's model?*

  The structure of the conflict hierarchy assures that the nodes higher in the hierarchy have all the general aspects of the nodes below them. The hierarchy is an is-a hierarchy, so a leaf node inherits all aspects of its parent node. At this time the resolution strategies have not been fully associated with the conflict hierarchy, although this appears to be quite possible.

- *Has the proposed model been successfully implemented?*

  COSINE implements the proposed SiFA model successfully. Although COSINE does not take advantage of the possibility of using inheritance in defining the agents (especially for their negotiation knowledge) [4] the implementation is simple enough to allow for automatic generation of the code by a knowledge acquisition tool that provides a graphical front end to the designer of a SiFA-based system.

- *Does the model, and implementation, allow for the integration of future improvements such as the ability of agents to learn during negotiation?*

  The model does not have any restrictive features, neither does the implementation. The model does not make any claims about learning and can definitely be extended to include it [6]. The implementation would allow history keeping, and learning simple constants, constraints, preferences, etc, but learning new methods, that are currently implemented as rules, would be difficult.

- *Are the negotiation behaviors of agents understandable to humans?*

  The sample runs of COSINE are very easy to follow. The agents are operating in an interleaved fashion, and it is possible to have more than one conflict detection and negotiation going on at once if the conflicts do not involve any related entities. However, for ease of understanding, one conflict is resolved before a second conflict is detected and its negotiation started.

- *Have any patterns of communication among certain pairs of agent types been identified?*

  Since the negotiation behavior of the agents are still very simple, there are very definite observed patterns of communication between the agents. This is a result of the extremely restricted vocabulary and strategies currently available to the agents, as opposed to patterns that emerge by the nature of the conflicts. We conjecture that similar patterns will be observed even if the agents are given more powerful negotiation strategies and a broader vocabulary.

Considerable progress has been made in meeting the goals of this work. The work on SiFA's is still new and the model keeps evolving. Work has concentrated on analysis of their properties. A large-scale implementation of a pure SiFA-based system that is capable of solving design problems does not yet exist. Consequently, little attention has been given so far to the performance of large SiFA-based systems.

9 Future Research

There are many directions that require further research in order to understand the full extent of both the power and the shortcomings of SiFA-based design systems.

One direction is to formalize the meaning and use of the parameter block's higher level entities, where meta-level knowledge about the design resides. It is not very clear what, or in what form, the information should be stored in the third level criticisms, praises, and evaluations in order to help the design process.

Although COSINE successfully resolves the agent conflicts among its agents, the negotiation behavior of the agents (their strategies) is very primitive. The simplicity of the negotiation process among the agents is one of the promises of the SiFA paradigm. However, more work is needed to construct and test negotiation strategies. A library of negotiation strategies for selectors, critics, evaluators, estimators, and praisers on all levels of reference would allow rapid (perhaps automated) construction of SiFAs with powerful negotiation techniques.

However, in order to construct such strategies, the language used among agents needs to be extended beyond the current vocabulary of alternatives and relaxations. Note that agents probably do not need to "understand" every other agent, just the ones it can conflict with.

Another direction is to provide SiFAs with history keeping capabilities. Even very simple history keeping would improve the system drastically. For example, if an agent keeps a record of its proposals that led to a conflict, it can avoid those in the future [6]. There are many other opportunities to improve the system with history keeping.

History keeping, criticisms and praises, traces of negotiations, and analysis of conflict situations, all facilitate learning. Things that might be learned include dependencies between design parameters, preconditions and postconditions for rules, negotiation strategies, and knowledge about other agents. Learning should improve the quality of the final design, reduce the time required to do the design, and provide better conflict anticipation, avoidance, and resolution. There will be opportunities to study the interactions between agents as they learn together. Agents learn from interactions, but their learning changes the way they interact.

10 Conclusions

10.1 Main Contributions

The first contribution of this work has been to extend the SiFA paradigm with the concept of a parameter block. The parameter block enables knowledge about the design to be presented explicitly, and related. It also allows multiple levels of meta-knowledge to be represented, through levels of reference. This allows comments about the decisions and comments of other agents.

Another contribution is the analysis of conflicts in the SiFA framework. Since SiFAs provide the building blocks of a multi-agent design system, the conflicts between them represent the primitives of conflict situations in any design. The conflict hierarchy enables general methods for handling the various tasks involved in the negotiation process, i.e., a method for detecting incompatibility conflicts will work for a whole group of conflicts under the incompatibility conflicts branch of the hierarchy. The negotiation model defined for the SiFA paradigm is another useful contribution.

10.2 Discussion

The SiFA-based model provides a very powerful tool with which to build and study design systems. The model allows a designer to build a system that will consider many different points of view while designing an artifact, without having to predict all possible conflicts between these points of view and resolve them while building the system. The separation of conflict knowledge from the design knowledge in the agents also has very important consequences in terms of the maintainability and the understandability of a system.

The main disadvantage of the SiFA model from a practical perspective is that the fine grained agent size is less efficient than using larger agents. However, we anticipate that the understanding gained from SiFA research will show how functionality should be grouped, and learning added, to produce more efficient systems.

The SiFA framework provides a new perspective to study design systems, concurrent engineering, multi-agent systems, and conflict management. SiFAs reflect a Concurrent

Engineering approach by allowing agents to represent the point of view of any life-cycle aspect at any decision making point in the design.

The major gain in building SiFA based design systems will be to learn more about the essence of conflict management and negotiation. SiFAs have proved to be very helpful in gaining insight into many design related research issues and they have revealed new research paths that should be explored further.

References

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

[2] D. C. Brown, B. V. Dunskus & D. L. Grecu. Using Single Function Agents to Investigate Negotiation. *AAAI Workshop on Models of Conflict Management in Cooperative Problem Solving*, 1994.

[3] R. E. Douglas, D. C. Brown & D. C. Zenger. A Concurrent Engineering Demonstration and Training System for Engineers and Managers. *International Journal of CADCAM and Computer Graphics, special issue on "AI and Computer Graphics"*, (Ed.) I. Costea, Hermes, Vol. 8, No. 3, 1993, pp.263-301.

[4] B. V. Dunskus, D. L. Grecu, D. C. Brown & I. Berker. Using Single Function Agents to Investigate Conflicts. *(AI EDAM): Artificial Intelligence in Engineering Design, Analysis and Manufacturing*, Special Issue: Conflict Management in Design, (Ed.) I. Smith, Cambridge UP, 1995.

[5] J. C. Giarratano & G. Riley. *CLIPS Reference Manual.* Volumes 1 & 2. NASA Lyndon B. Johnson Space Center Information Center Information Systems Directorate, Software Technology Branch, Version 6.0, June 2nd 1993.

[6] D. L. Grecu & D. C. Brown. Learning by Single Function Agents During Spring Design. *Proc. AI in Design Conference, AID'96*, Stanford, CA, August 1996.

[7] M. Klein. Supporting Conflict Resolution in Cooperative Design Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21, No. 6, November/December 1991, pp. 1379-1390.

[8] M. Klein & S. C.-Y. Lu. Conflict Resolution in Cooperative Design. *The International Journal for AI in Engineering.* No.4, 1990, pp. 168-180.

[9] M. Klein & S. C.-Y. Lu. Insights into Cooperative Group Design: Experience with the LAN Designer System. *Proceedings of the 6th International AI in Engineering Conference*, G. Rzevski & R. A. Adey (Eds.), Oxford, UK, Elsevier, July 1991, pp. 143-162.

[10] D. R. Kuokka & L. T. Harada. Communication Infrastructure for Concurrent Engineering. *(AI EDAM): Artificial Intelligence in Engineering Design, Analysis and Manufacturing*, Special Issue: Conflict Management in Design, (Ed.) I. Smith, Cambridge UP, 1995.

[11] T. Finin, J. Weber, G. Wiederhold, M. Genesereth, R. Fritzson, J. McGuire, D. McKay, S. Shapiro, R. Pelavin & C. Beck. *Specification of the KQML Agent Communication Language*, Official Document of the DARPA Knowledge Sharing Initiative's External Interfaces Working Group, 1993.

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

[13] F. Polat, Shashi Shekar, & H. A. Guvenir. A Negotiation Platform for Cooperating Multi-agent Systems. *Concurrent Engineering: Research and Applications*, Vol. 1, No. 3, Academic Press, September 1993, pp. 179-187.

[14] *(AI EDAM): Artificial Intelligence in Engineering Design, Analysis and Manufacturing*, Special Issue: Conflict Management in Design, (Ed.) I. Smith, Cambridge UP, 1995.

[15] K. P. Sycara. Resolving Goal Conflicts via Negotiation. *Proceedings of the AAAI-88*, Volume 1. St. Paul, MN, 1988, pp. 245-250.

[16] K. P. Sycara. Cooperative Negotiation in Concurrent Engineering Design. *Cooperative Engineering Design*, Springer Verlag Publications, 1990, pp. 269-297.

[17] 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. *Applications of AI in Engineering VIII. Vol. 1: Design, Methods and Techniques.* G. Rzevski, J. Pastor & R. A. Adey (Eds.), Proceedings of the 8th International AI in Engineering Conference, Toulouse, France. Elsevier, June 1993, pp. 83-96.

[18] S. K. Victor & D. C. Brown. Designing with Negotiation Using Single Function Agents. *Applications of Artificial Intelligence in Engineering IX.* G. Rzevski, R. A. Adey & D. W. Russel (Eds.), Proceedings of the 9th International AI in Engineering Conference, Pennsylvania, USA. Computational Mechanics Publications, July 1994, pp. 173-179.

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

[20] M. Wooldridge & N. R. Jennings. Intelligent Agents: Theory and Practice. Submitted to *Knowledge Engineering Review*, 1994.

11 Figures

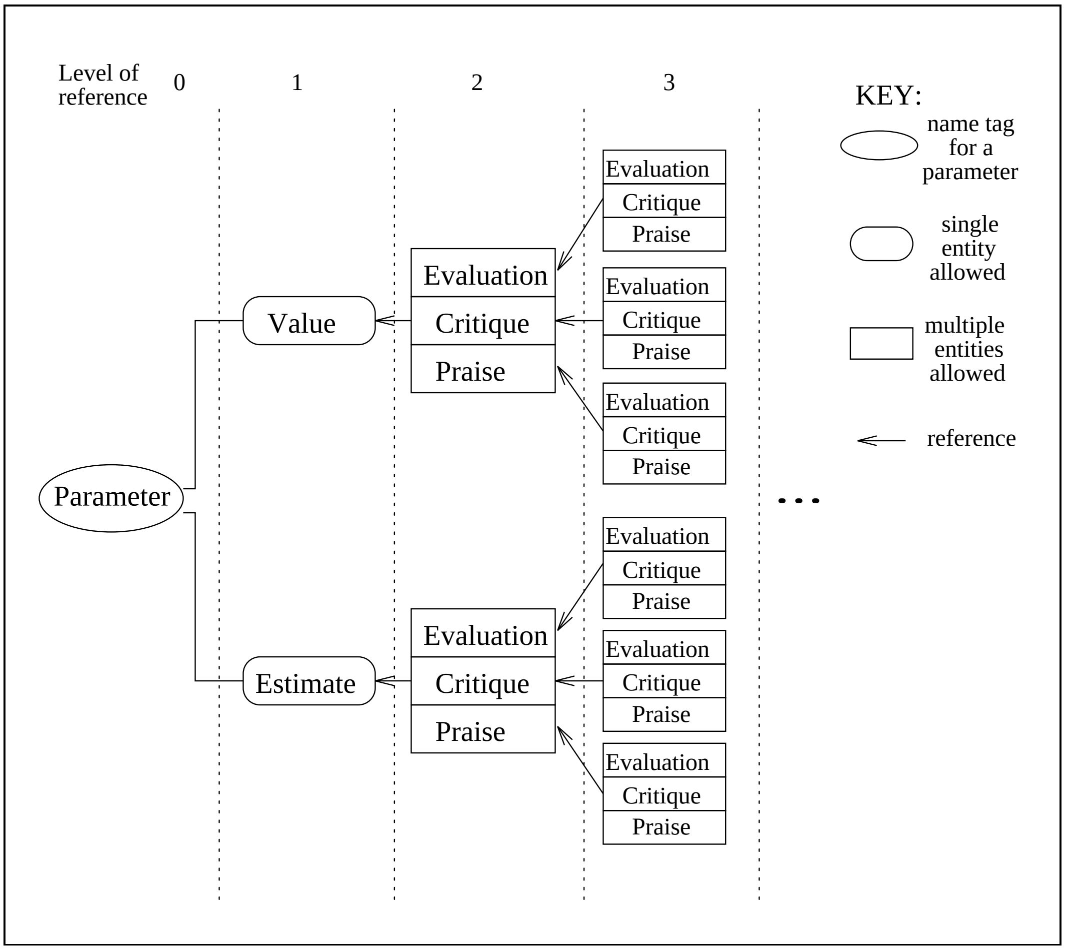

Figure 1: The parameter block

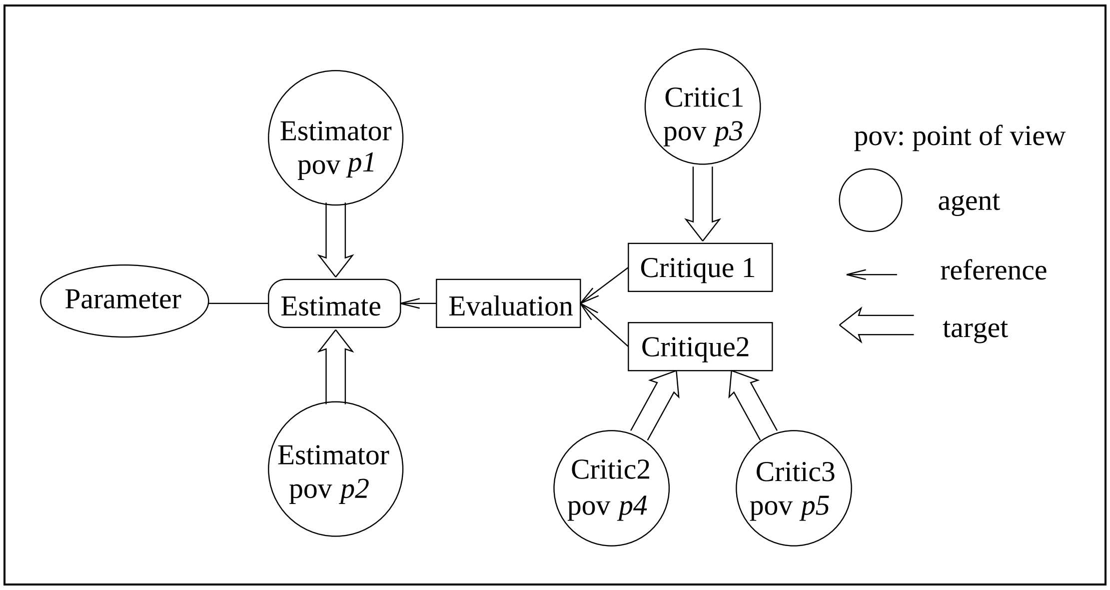

Figure 2: Estimate evaluation critics

Figure 3: The knowledge contained in a SiFA

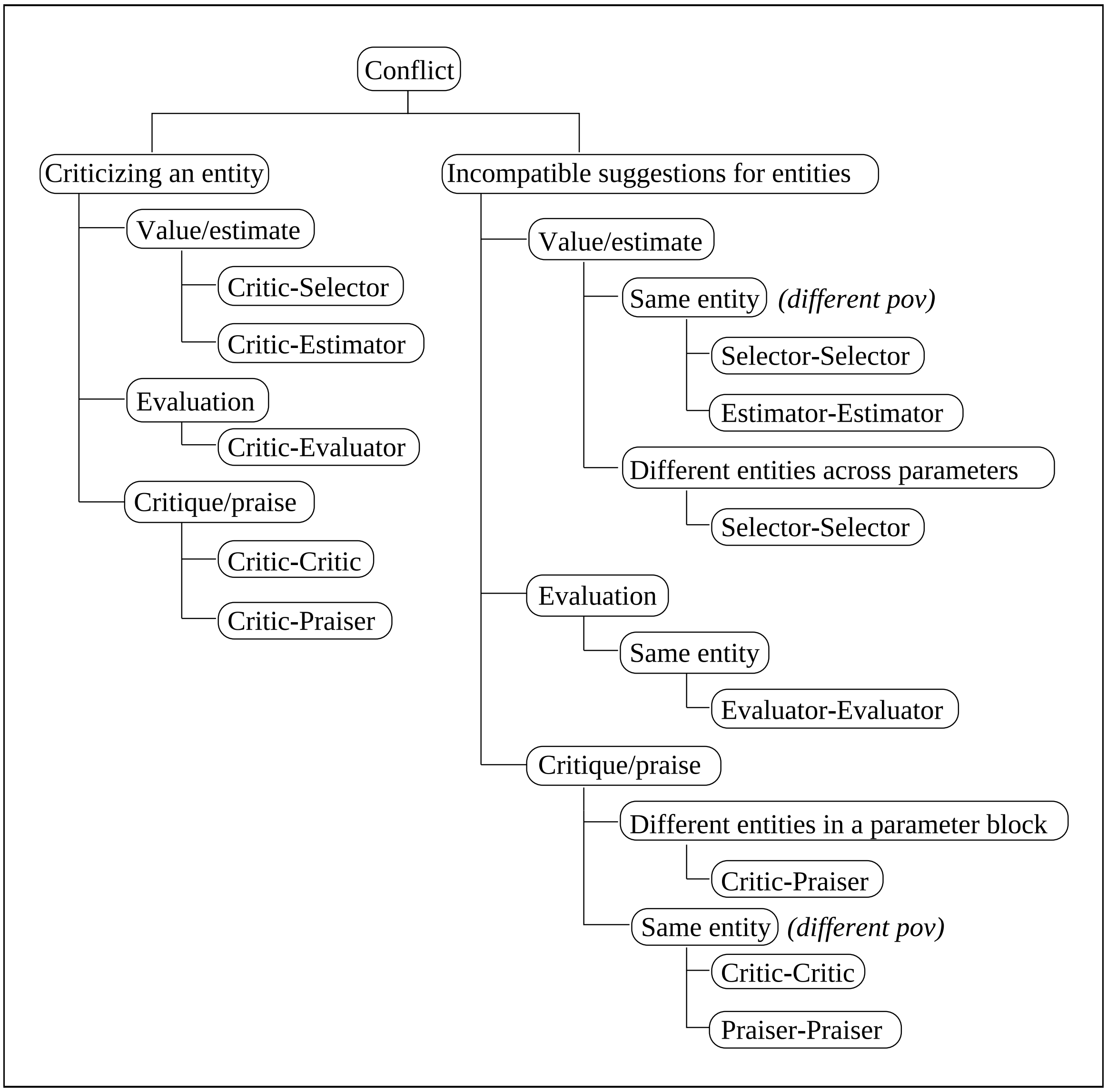

Figure 4: The SiFA conflict hierarchy

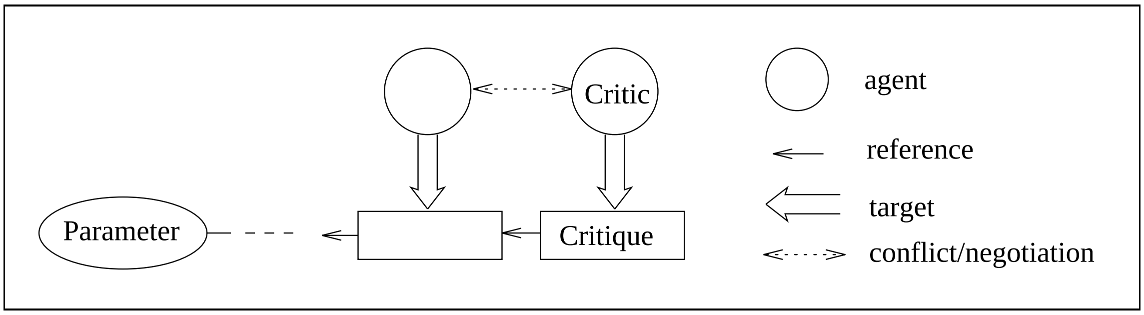

Figure 5: Criticizing an entity

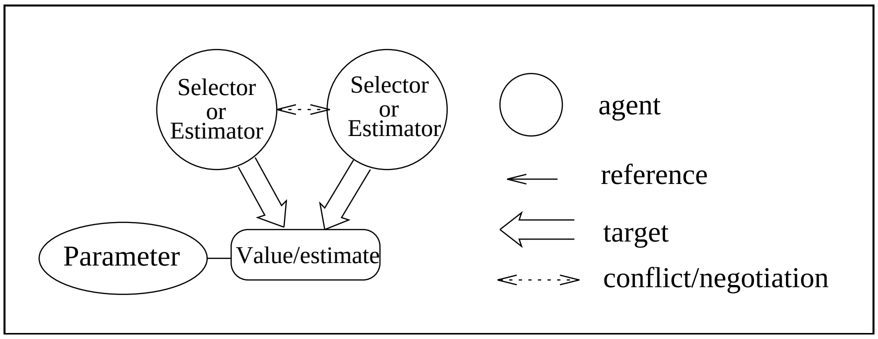

Figure 6: Incompatible suggestions for the same value/estimate entity

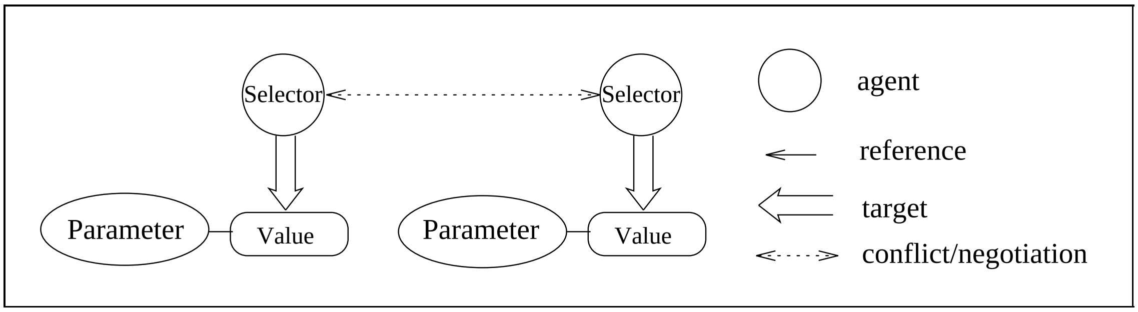

Figure 7: Incompatible suggestions for different value entities

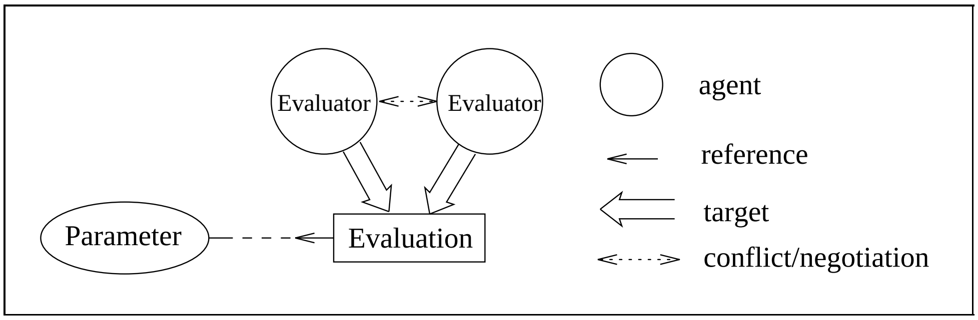

Figure 8: Incompatible suggestions for an evaluation

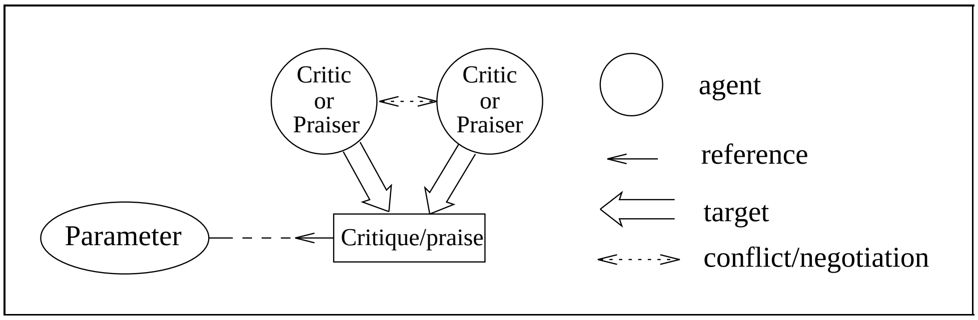

Figure 9: Incompatible suggestions for the same critique/praise entity

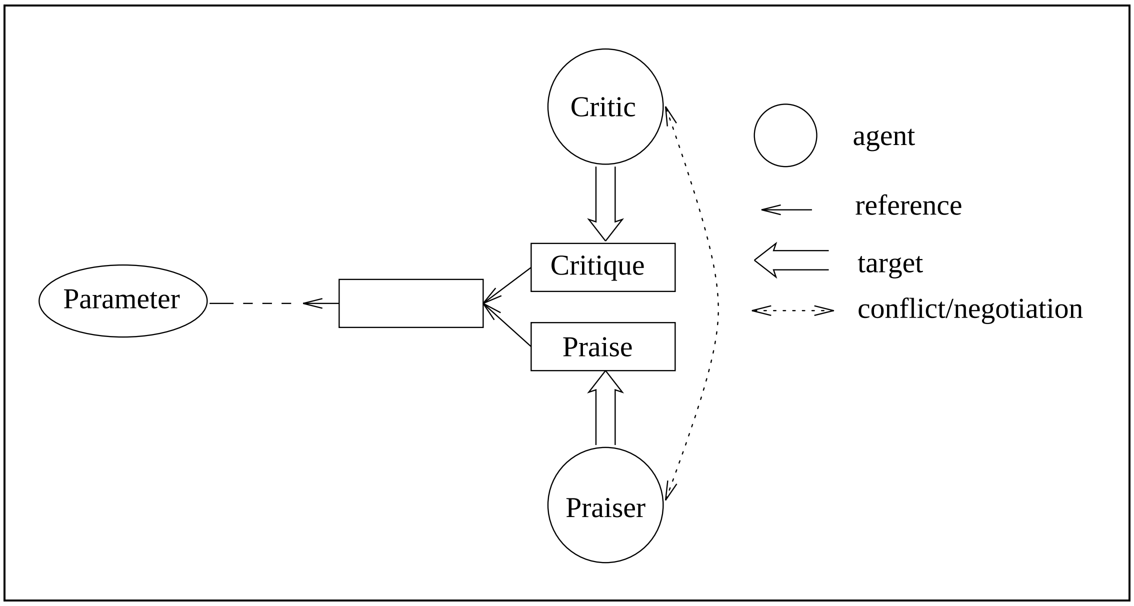

Figure 10: Incompatible suggestions for different critique/praise entities

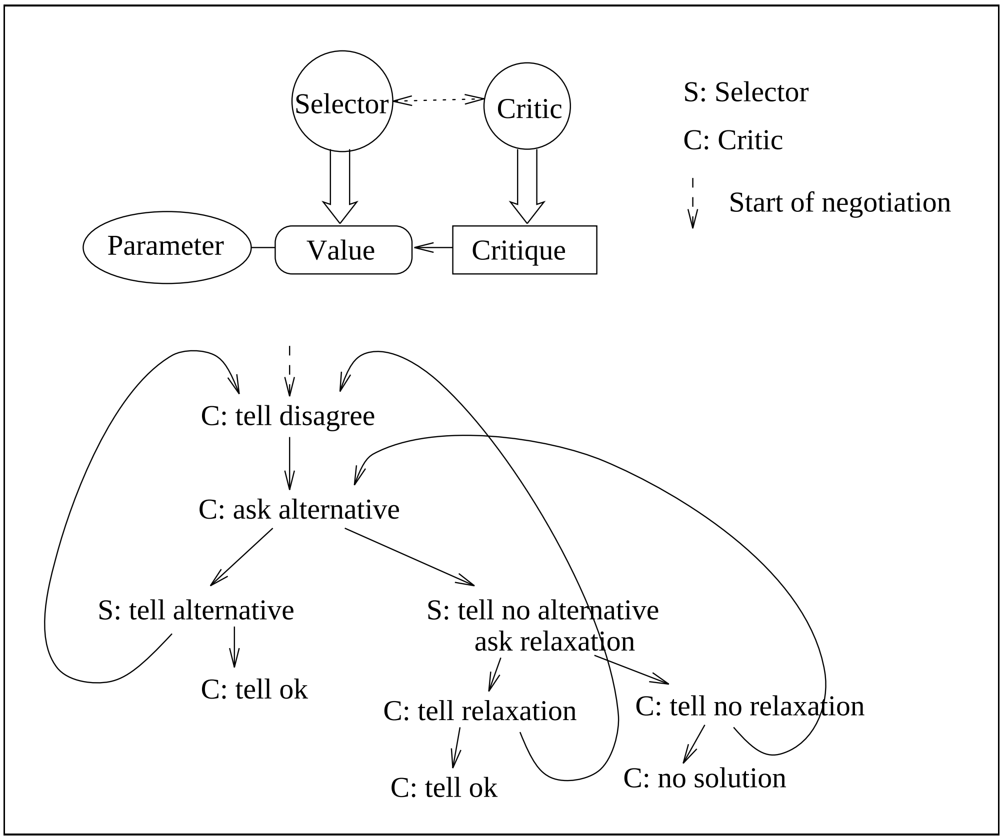

Figure 11: Selector-Critic negotiation graph

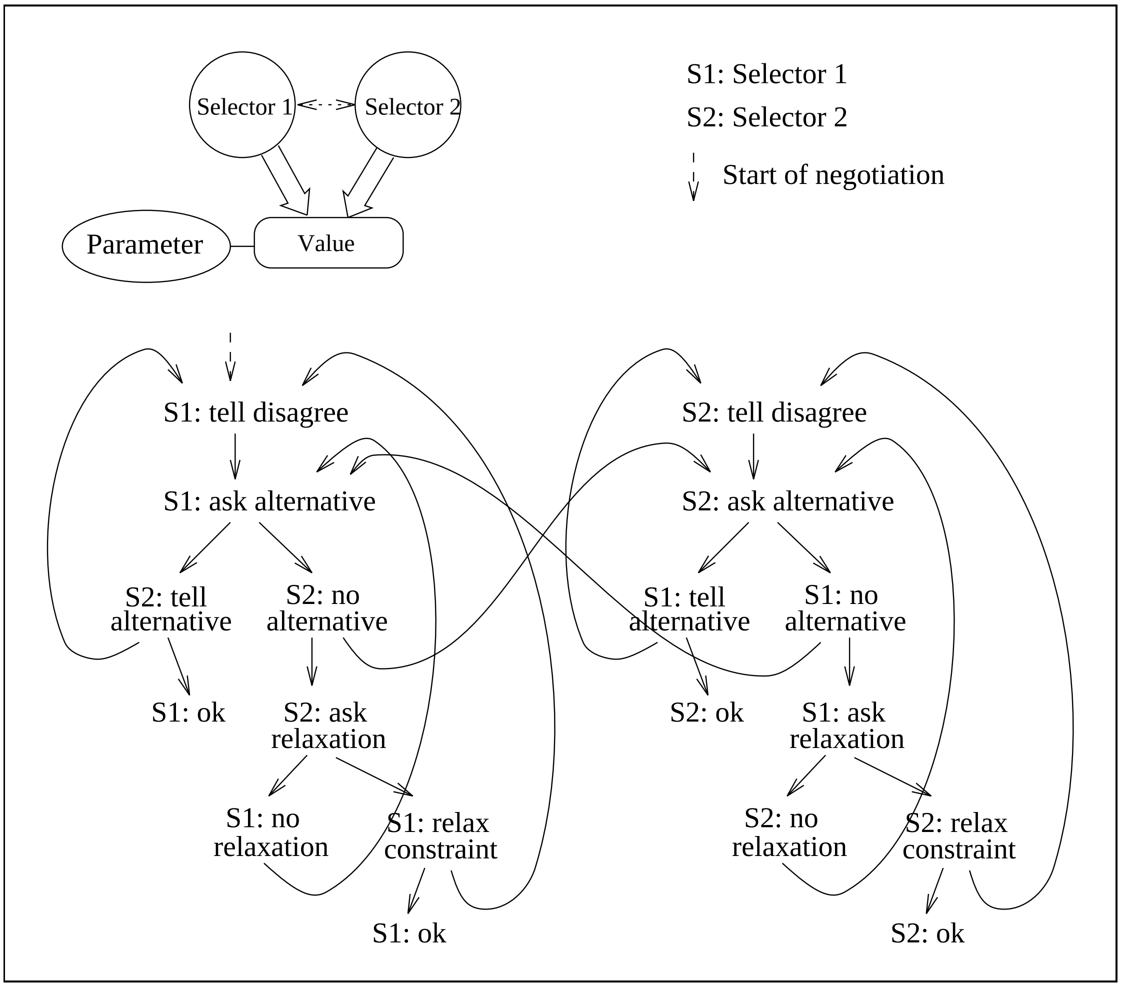

Figure 12: Selector-Selector negotiation graph

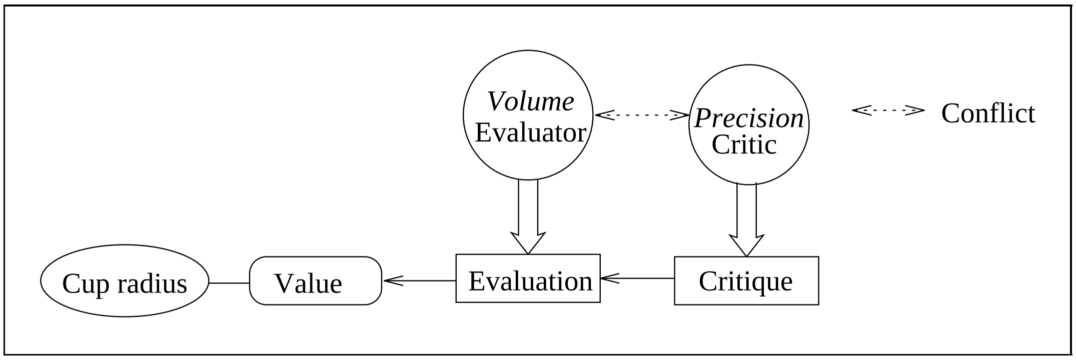

Figure 13: Criticism of an evaluation

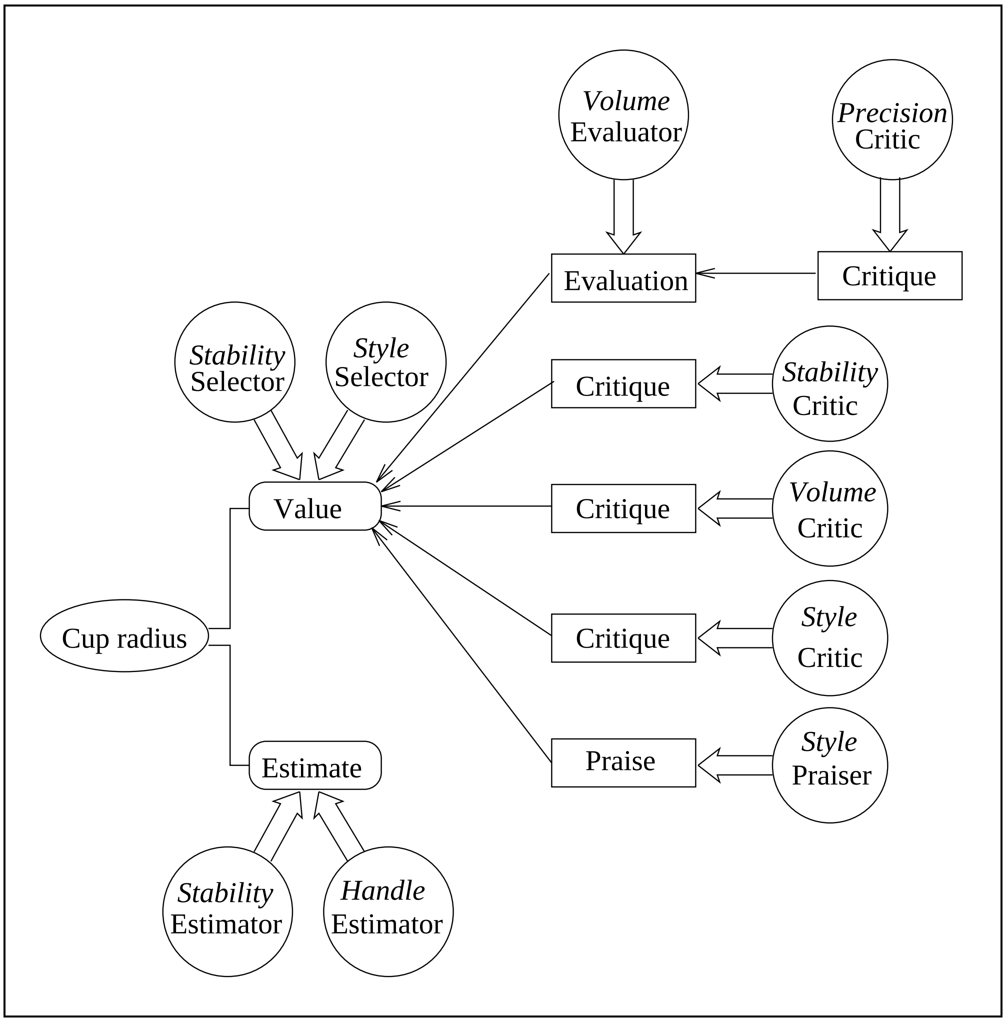

Figure 14: Cup radius parameter block in COSINE

12 Bios

Ilan Berker has a B.S. in Computer Engineering from Bosphorus University, Istanbul and an M.S. degree in Computer Science from Worcester Polytechnic Institute, Massachusetts. His research at WPI focused on conflict management and negotiation in agent-based design systems. His other interests include machine learning and design rationale in multi-agent systems. Ilan Berker currently works as a software design engineer for Microsoft Corporation.

David C. Brown is a Professor of Computer Science at Worcester Polytechnic Institute, and has B.Sc., M.Sc., M.S. and Ph.D. degrees in Computer Science. He is a member of the ACM, IEEE Computer Society, and the AAAI. He is on the Editorial Boards of the Journals "AI in Engineering, Design, Analysis and Manufacturing" and "Concurrent Engineering: Research and Application". He is a member of the Advisory Committee for the AI in Engineering, the AI in Design Conferences, and for the IFIP WG 5.2 Workshops and Conferences. His current research interests include computational models of engineering design, and the applications of Artificial Intelligence to Engineering and Manufacturing. He is the author, with B. Chandrasekaran, of the book "Design Problem Solving: Knowledge Structures and Control Strategies", Pitman Publishing, Ltd., and a co-editor of "Intelligent Computer Aided Design", Elsevier Science Publishers B.V. (North-Holland).