Guiding Agent Learning in Design

Dan L. Grecu & David C. Brown
Worcester Polytechnic Institute, Worcester, MA 01609, USA

Key words: machine learning, multi-agent systems, expectations

Abstract: In this paper we discuss the need for learning in multi-agent design systems, and the variety of forms it might take. We propose a particular method of guiding learning in these systems, describe an architecture for its implementation, and discuss how the learning should be evaluated.

1. INTRODUCTION

This paper is concerned with the use of Machine Learning techniques in Multi-Agent Design Systems (MADS). It is clear that designers who are attempting to solve large and complex design problems require computational support, and that some aspects of these designs might be fully automatable. MADS have been proposed as a viable approach to building such design systems. They offer a variety of advantages, including extensibility, the potential for parallel activity, and the capability of acting in a distributed manner (Lander 1998).

In this paper we discuss the need for learning in MADS, and the variety of forms it might take. We propose a particular method of guiding learning in a MADS, and describe an architecture for its implementation.

2. LEARNING IN A MULTI-AGENT DESIGN SYSTEM

2.1 Opportunities for learning

Multi-agent design systems (MADS) are similar to organisations: their performance can be improved. Two characteristics of performance in organisations are *effectiveness* and *efficiency* (Etzioni 1964). Effectiveness is the degree to which the goals of the system are attained, i.e., the quality of the solution. Efficiency refers to the amount of resources used to produce the result.

In a MADS, effectiveness is affected by the amount, distribution and use of the knowledge in the system, as well as by the accuracy and completeness of information exchange. Efficiency is affected by the details of the process used, including the number of conflicts, the amount of communication, the reasoning approaches selected, and the quality of conflict resolution strategies.

It would be nice if systems that were both highly efficient and highly effective could be developed from scratch. However, this is rare. As design systems move from addressing small and 'neat' design problems towards real-world design tasks, it becomes harder at development time to address the range of efficiency and effectiveness issues that the system will encounter at run-time.

Ideally, we would like to have a system where design agents base their decisions on all the knowledge that is available in the design system, and where they know the possible consequences of every potential decision. The utilities associated with these consequences would drive the decision selection, and would allow agents to precisely respond to design goals.

The truth is that agents have only limited information about how other agents operate, about their knowledge, and their internal reasoning strategies. Within a multi-agent system, it is not possible to anticipate all possible interactions between agents *a priori*. Furthermore, agents typically see only the part of the design covered by their domain competence.

As a result, agents support their decisions based on *the knowledge they have*, and not on *the knowledge that is available in the system*. Furthermore, an agent may *sometimes* know *some of the consequences* of a decision it has made, but it cannot know or compute all the consequences of its decisions. Decisions can be made based on *heuristic criteria*, and consequences can be *evaluated*. The difference between the ideal and the real setting opens opportunities for improvement through learning.

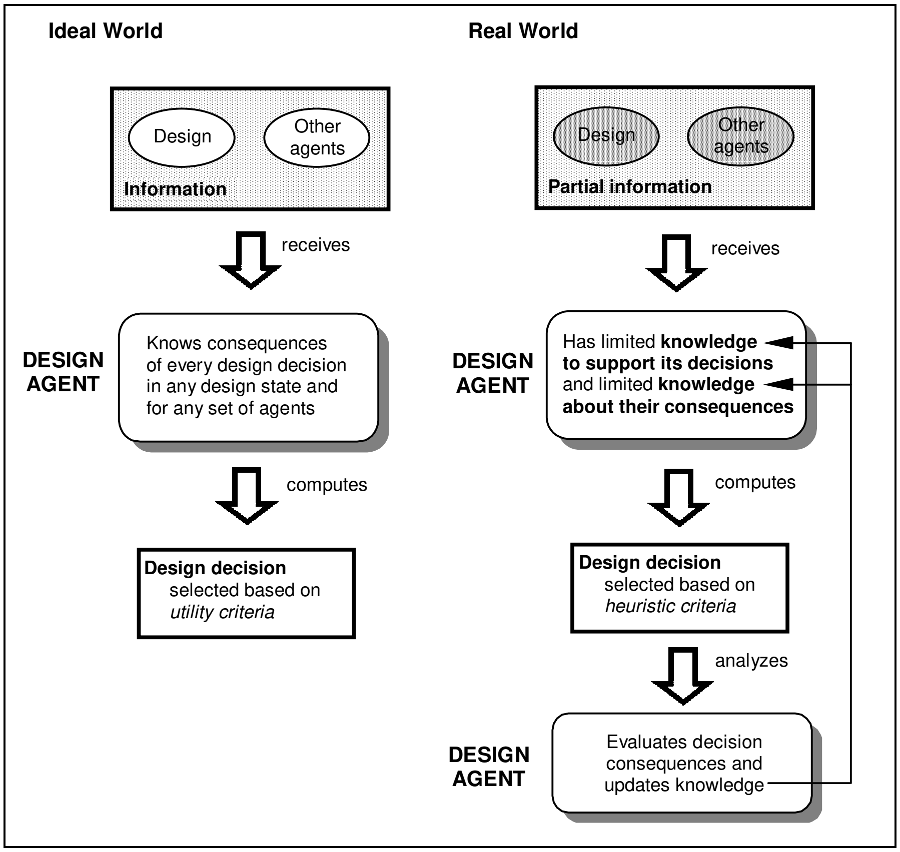

Figure 1. Agent decision-making in design

2.2 Dimensions of learning

In a multi-agent design system there is no single place where learning might occur, and no single time for it to occur. Such a system is complex, with many types of knowledge and reasoning, many types of interaction, and many roles for agents to play. This complexity provides a myriad of opportunities for learning (Grecu and Brown 1998a).

There are several “dimensions” of learning in a design context. For example, the *triggers* of learning might include failure, success, expectation violations or a perceived need to improve some aspect of the system. Different types of information may support learning. Some of it might be available while designing (e.g., critiques), and some after a design is produced (e.g., feedback about the design's quality). Some of it is communicated directly to the learner (e.g., another agent's design decision), while some is collected and available for retrieval (e.g., design traces).

Perhaps the most variety is found in what gets learned, i.e., the targets of the learning process. These include constraints, dependencies between design parameters, plans, preferences, and the consequences of design decisions.

3. FLEXIBLE AGENT LEARNING

Human problem-solvers do not persist with the same learning task forever. They start by identifying areas where they need to change their approach to the problems they face. Then they use learning in an attempt to acquire knowledge that will allow them to be better at the task. In time, the acquired knowledge will lead the problem-solvers to different decisions in situations similar to the ones that triggered the learning process in the first place.

The temporary nature of the learning process is highlighted by two observations: First, the improved results achieved by the problem solver will presumably reduce the need for learning in a specific area, and other areas of concern in the problem-solving process may become more important for learning. Second, the learning environment changes, as the learned knowledge is used. Therefore, the learner will only have a finite opportunity to learn from a specific context.

These observations establish several requirements and constraints for learning design agents. We will refer to them interchangeably as learners and agents, to stress their role as both design problem-solvers and adaptive entities.

First, an agent should be able to determine *on its own* that a specific need for learning has occurred. For instance, a design agent may decide that it needs information about how particular design and manufacturing choices influence the cost of a product, to reduce the number of times products are rejected based on cost criteria. While the types of learning needs may be pre-defined by the implementer of a MADS, for example, "The agent needs to learn in response to conflicts", the occurrence of a learning need at run-time has to be determined by the agent itself.

Second, given that the learning scenario is no longer pre-scripted, design agents are themselves responsible for identifying the 'ingredients' of a learning process: *the learning target*, e.g., the cost of a component; *the sources that will provide information to support the learning process*, such as design and design process parameters, dependencies, and constraints; and, *the learning strategy*, e.g., inductive learning or explanation-based learning.

Finally, an agent needs to recognize when a learning process should be stopped. This may be because no further improvements are achieved through the learning process as it is, or because the information acquired through learning does not produce reliable or desired results.

In summary, agents, like humans, have to display flexibility in learning if they are to prove efficient in constantly changing environments. The ability to determine the need for learning, the parameters of learning, where and what to learn, and the possibility of refocusing on new learning tasks are essential for multi-agent design systems. This is especially true when attempting to scale up to new problems: sooner or later the system will have to operate with knowledge and contexts that were not anticipated when the design system was developed.

In the following we will propose to use expectations as a basis for flexible learning in design. We will show how expectations can be acquired dynamically, to respond precisely to the requirements we have enumerated above. Before we focus on the learning process itself, let us first define expectations in the particular context of design, and how expectations are used in design.

4. EXPECTATIONS IN DESIGN

4.1 The observable world of a design agent

Design agents have knowledge about the problem domain in which they operate and about the agent environment in which they find themselves. The very idea of using agents in problem-solving suggests that agents are *specialized*. In mechanical design, for example, an agent's *domain* can be restricted to materials and their physical properties, to assembly operations, or to product marketing. The agent's *tasks* can range from making decisions about the design, to critiquing design aspects, or to evaluating design parts.

Within its 'society' an agent may know about the roles or specializations of the agents with which it interacts, about when to act, how to communicate, and how to solve conflicts with other agents. However, a realistic approach which considers the resources available to an agent would not be able to anticipate or to compute the behavior of all the other agents in the system (Cherniak 1986; Russell and Wefald 1991).

We define the *observable world* of an agent as the collection of features, in the design domain and in the agent environment, that the agent can 'perceive'. The observable world of an agent is delimited epistemologically and by the physical nature of the access to information.

Epistemologically, the observable world of an agent is constrained by the agent's specialization and by its limited ability to infer new knowledge. An agent that is specialized in material selection may have no use for information about the impact of color on product aesthetics or on marketing. Therefore, color aspects of the designed product may not be part of the set of concepts used by the material selection agent. Alternatively, even if the agent may have the knowledge to reason about how colour impacts aesthetic aspects, it may be time constrained in executing a task that has a relatively low priority, or requires extensive processing.

Physical access to information depends on two main factors. If the design agents are physically distributed over a network, relevant design or process information, such as agent behavior, that may be typically monitored if the agents were co-located, may become available only through communication with other agents. Second, even if the agents are located on the same machine, the notion of agency implies an encapsulation of information that makes it invisible outside of an agent, unless deliberately exposed through posting or communication.

4.2 Defining expectations

Expectations express the belief that an event will happen. More precisely, not only *that* an event might happen, but also *the circumstances* or *conditions* under which the event will happen.

Expectations are typically created because limited resources prevent the holder of the expectation from establishing a proven causal relationship between the set of conditions and the ensuing situation. The considerations outlined in discussing the observable world of the agent – time, the factual information about the conditions that predict the situation of interest, or the knowledge needed to establish the causal connection between the conditions and the situation to which the expectation refers – provide the reasons for these limited resources. If agents were omniscient and had unlimited computational power, expectations would not be needed, since events, values and outcomes would be computable in advance.

In our multi-agent design paradigm, expectations represent the knowledge of agents that events will occur in a pre-defined way: design parameters will be within specific ranges, responses from other agents will arrive within a given amount of time, or decisions will lead to given outcomes. Figure 2 shows an example of an expectation, expressed as a rule. The conditions for the cost expectation include conditions related to the design and to the design agents.

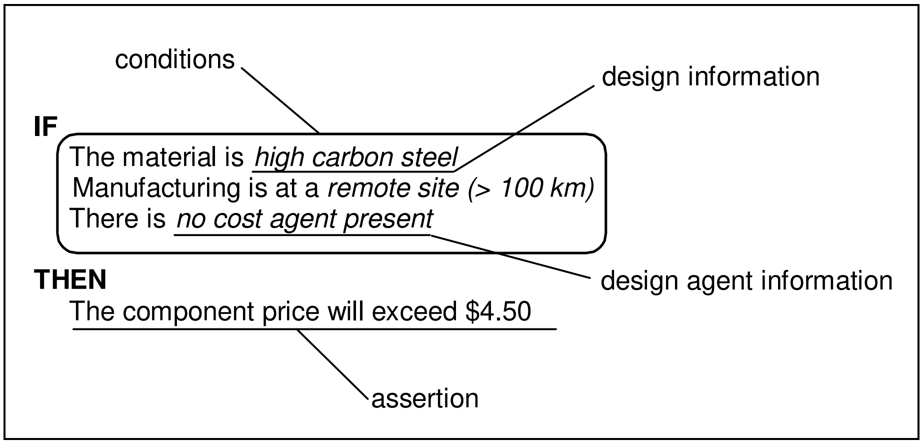

Figure 2. Design expectation example

Expectations have an *empirical character*, in that often there is no deductive connection between the conditions that are observed and the situation that is asserted. The absence of the deductive connection is either due to the absence of the knowledge that allows to make the necessary inferences, or to the unavailability of the resources to carry out the computation.

Expectations are a *tentative* form of knowledge, which has to be acquired dynamically. Consequently, they have to be set up based on the learning requirements described in section 3, they have to be monitored and updated, and eventually they have to be validated or rejected.

4.3 Expectation-based design decision making

In our approach to multi-agent design, expectations are used when a design decision is being proposed, and when its consequences are evaluated (figure 3).

Using an expectation while taking a design decision is necessary when a piece of information that is a precondition for that decision cannot be or has not been inferred. For example, an agent may need to know whether the material that will be used in a component is resistant to corrosion. However, it may be the case that the agent holds an expectation that provides the required information, such as:

IF
The material is steel
The component is chrome plated during manufacturing
THEN
The component is corrosion resistant

If the conditions for the expectation hold, then the agent can use the assertion as a basis for the decision it is about to take.

Once a decision has been selected, the design agent will evaluate its consequences. Again, expectations will bridge the gap of unavailable knowledge. An agent may have expectations about the cost range of its solutions, and use them to verify that at that point the solution it is about to propose will not drive the overall cost above a given limit. If the expectation indicates that the constraint will not be satisfied the agent will revise its design decision, to prevent a later constraint violation or a conflict.

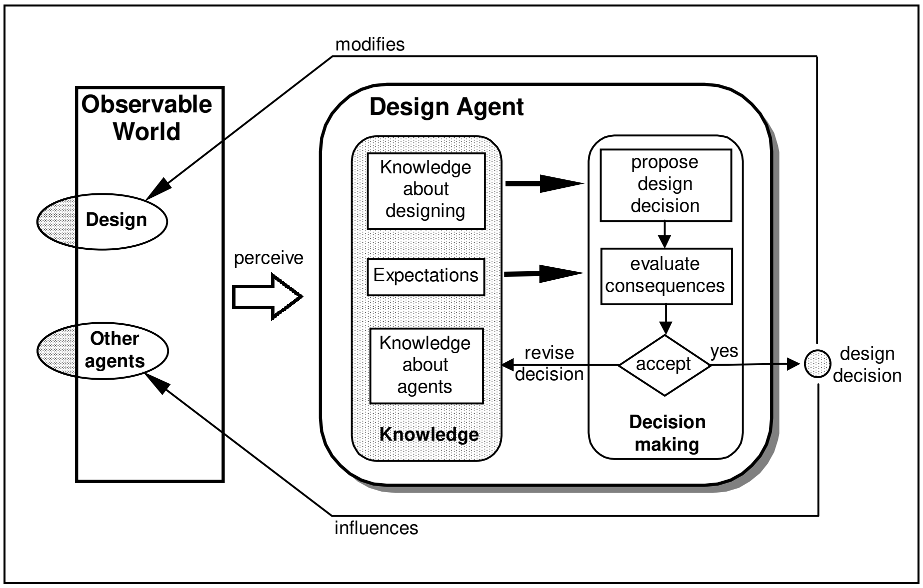

Figure 3. Using expectations in design decision-making

5. EXPECTATIONS AS A BASIS FOR AGENT LEARNING

In this section we will describe how agents acquire expectations. An agent initiates an expectation learning process when it determines that it needs information about a design or a design process element. As described in the previous section, the need may occur when the agent tries to determine whether the conditions for a specific decision are met, or when it tries to evaluate the consequences of a decision. The need for information defines the assertion of the future expectation, that is the event or parameter that the expectation will predict.

Once the agent decides to acquire an expectation, it proceeds through two major phases: learning an initial expectation and monitoring/updating the expectation.

5.1 Learning expectations

5.1.1 The learning process

To learn an expectation an agent identifies the conditions that predict the event on which the expectation focuses, i.e., the assertion of the expectation. This amounts to a causal reasoning process, in which the design agent searches its observable world for conditions that might influence the assertion. Recent research in understanding the mechanisms that underlie causal reasoning has identified two major stages within this process: the use of causal mechanisms to delimit a set of candidate of conditions, and the use of covariational principles to extract from the candidate conditions the subset that is relevant for predicting the assertion (Koslowski 1996).

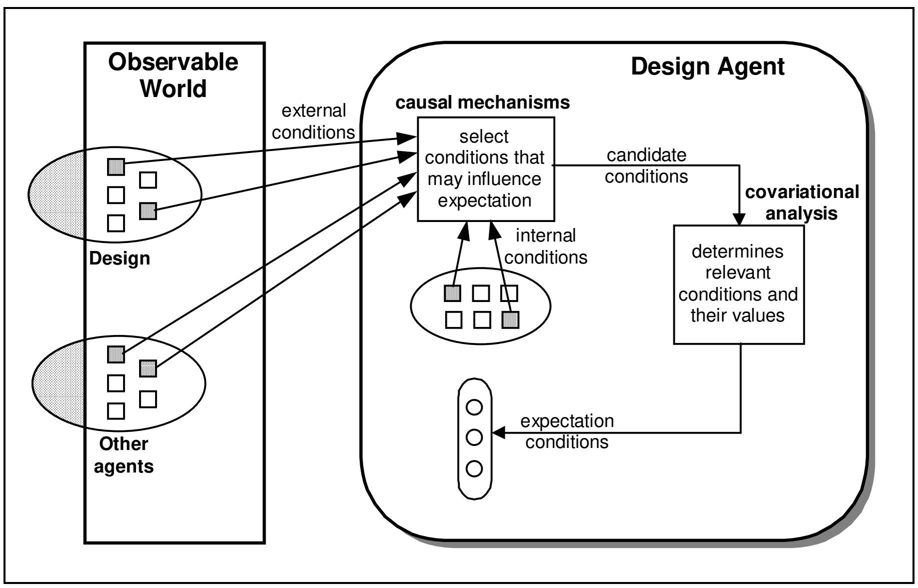

Figure 4. Learning expectations

Accordingly, a design agent implements a two-stage expectation learning (figure 4). In a first stage, the agent uses **causal mechanisms** to select from the external world and from its own domain specific knowledge candidate conditions that, in some combination, might affect the expectation assertion. Subsequently, these conditions are submitted to a **covariational analysis** to construct a subset of relevant conditions that are the ones that indeed impact the assertion.

To illustrate this process, figure 5 provides an example of a spring design agent that, in the context of deciding the diameter of a spring, needs to anticipate the range of the cost of this component. This knowledge need triggers the learning of an expectation, which will make an assertion on the cost of the component. The agent's causal mechanisms will first set-up a list of possible factors (the expectation's conditions) that may impact the cost. In doing so, the agent selects its own choice of material – an internal design condition, the range of stress and the manufacturing site – external design conditions determined by other agents, and the presence of a cost critique agent – an external condition that refers to other agents.

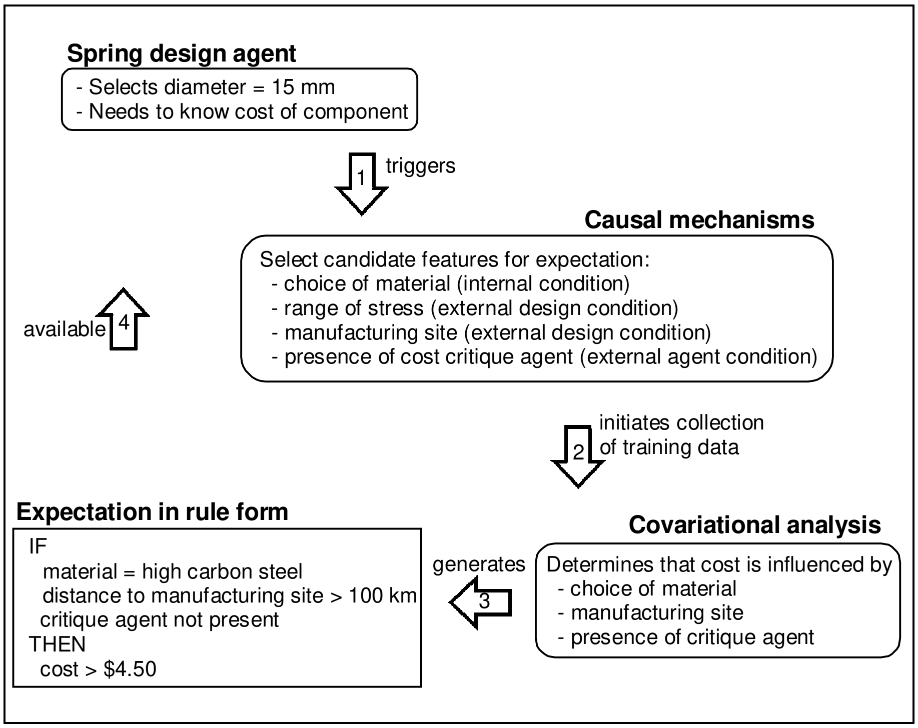

Figure 5. Expectation learning example

Once the candidate conditions are selected, the agent will collect training data for the covariational analysis. The training data is obtained from subsequent design sessions in which the design agent records values for the expectation conditions, as well as for the expectation assertion, i.e., the cost. Based on the training data, the agent eliminates redundant and irrelevant conditions, and generates an expectation in rule form, that now becomes available to the design agent.

5.1.2 Causal mechanisms and covariational analysis

The causal mechanisms involved in the first stage of the expectation learning process play a fundamental role in focusing the learning process. A pure covariational process would be simply overwhelmed by the number of influence factors it would have to consider. It has been argued that people rely only on statistical associations to identify causes and explain events, and deviations from this behavior were regarded as cognitive biases (Tversky and Kahneman 1974). A significant body of evidence indicates that this is the case only when any other evidence or information is lacking. However, domain experts tend to go through a causal attribution stage in which they use domain specific knowledge to reason about possible causes for an event (Hilton 1990; Koslowski 1996; Shultz et al. 1986).

Design agents being domain experts have access to knowledge that allows them to hypothesize possible causes for an event. Dependencies between design parameters, either explicitly represented, or represented implicitly as constraints represent one source of causal attribution. Actions or attributes of agents that include in their domain of expertise the parameter to which the expectation assertion refers are another important source. A task that computes the parameter present in an expectation assertion can also provide causal information, and even more so when the task was divided into sub-tasks.

The covariation analysis is an inductive learning stage in which expectations are seen as concepts. The expectation conditions are the concept features, while the ranges for the expectation assertion, such as the component cost in the previous example, represent the concept classes. The inductive learning algorithm attempts to learn a representation for the concept. The features of the resulting concept description are the *relevant* conditions that the agent has identified as influencing the occurrence of specific assertion ranges, i.e., classes.

To achieve this learning goal, agents use wrappers for relevant condition selection (figure 6). Wrappers (Kohavi and John 1998; Liu and Setiono 1998) apply an induction algorithm to a training data set. The experiments are run by eliminating different sets of features from the training data instances. Specifically, wrappers eliminate conditions from the candidate condition set. The wrapper method proposes a subset of features that are relevant for the identification of a given class. Features are considered relevant if their “values vary systemically with category membership” (Gennari, Langley, and Fisher 1989).

The learner performs a search in the space of subsets of features to identify one subset that allows for both a reduced description and good prediction of the new expectation that is being learned. The wrapper maintains several subsets of candidate features. An accuracy testing component determines the performance of each subset, and eliminates or adds new subsets of features, by providing information to a feature selector.

Wrappers have the major advantage of being able to work with different learning algorithms, as long as the algorithms have the same interface. Therefore, the approach provides flexibility in choosing and testing different learning algorithms without affecting the agent or the multi-agent system. They have also been proven to be effective in pruning large initial sets of features (Kohavi and John 1998). Therefore, even if the agent does not have a strong set of causal mechanisms for setting up a new expectation, and producing a small set of candidate conditions, the wrapper technique can partially compensate for this weakness.

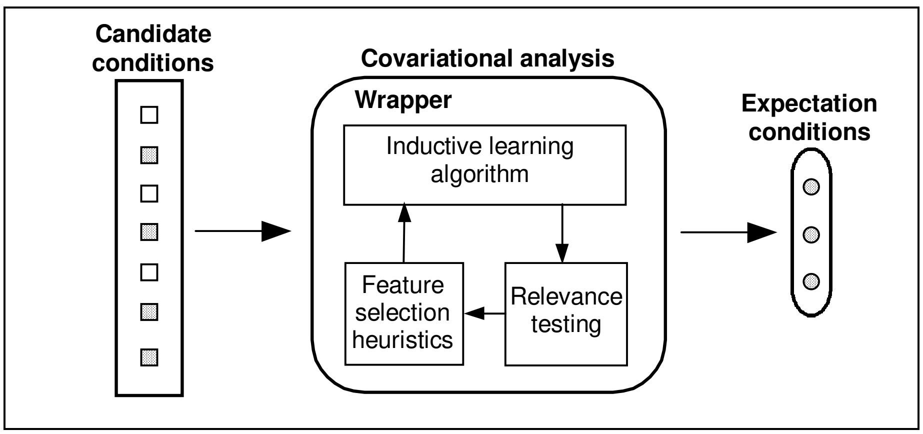

Figure 6. Selection of relevant expectation conditions

5.2 Monitoring expectation validity

Given the fact that expectations are set up empirically, design agents need to validate them before using them (figure 7). During the validation process an expectation is used to make predictions wherever the expectation assertion is needed. The value that was predicted by the expectation is then compared with the final value resulting from the design process. If the expectation is violated, that is, if the resulting value does not match the predicted assertion, the agent needs to review the expectation.

Reviewing an expectation implies that the agent will re-initiate the training stage of the learning process. It is assumed that the new data will not enable the agent to change its causal mechanisms, although the new evidence might allow an agent to use knowledge based reasoning to pinpoint a particular condition that needs to be changed. For retraining, the agent will start to collect additional training instances about the use of the expectation. Whenever the expectation is supposed to be used the values of the conditions will be recorded together with the value of the assertion resulting from the design. The retraining will generate an updated expectation.

The overall review process is can be repeated for a pre-defined number of times. If the expectation does not reach a stable status the agent will drop the expectation.

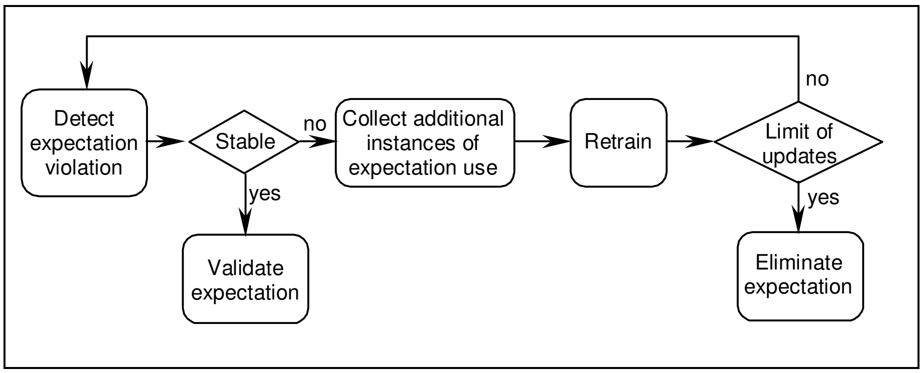

Figure 7. Monitoring expectation validity

Several causes can prevent an expectation from being accepted. The causal mechanisms can lack sufficient coverage to include important discriminating conditions in the candidate set. Another possibility stems from the fact that several expectation learning processes can proceed simultaneously in several of the agents, thus changing their decisions and their behavior. If one of the changing elements associated with an agent is included among the conditions of an expectation that is developed by another agent, it is likely that this expectations will take a longer time to 'stabilize', or may lead to it being eliminated.

6. THE AGENT ARCHITECTURE

The multi-agent architecture that we use models a group of designers. Agents act as design specialists and as group members. There are no agent hierarchies or relations between the agents that create rigid 'links' between them. However, the types of interactions between agents are predetermined, and they essentially represent the rules that create the group behavior. The interactions result dynamically, at run-time, and originate in the problem the system attempts to solve. The agents have complete autonomy in organizing their actions, with regard to the decisions they take as design specialists, or to their interactions with the rest of the group.

The agent model evolved from the Single Function Agent (SiFA) paradigm (Dunskus et al. 1995), and includes specialized, knowledge-based design agents with precise functionality. Each agent has a predefined function in the design process. The agent types we see as most important are:

– *Designers* – agents that are responsible for taking design decisions, such as selecting values for design parameters, or creating links between design components in a configuration process.
– *Critics* – agents that criticize design aspects, such as design parameter values, or weak properties of component configurations. Beyond revealing undesirable properties of the design, critics may point out constraints or quality requirements that are not met by the design aspect on which they focus.
– *Praisers* – are meant to praise design aspects that rate particularly highly from a given point of view. Positive evaluations are important when designers have to decide which parts of the design need to be revised and which ones should preferably remain unchanged.
– *Estimators* – produce estimates of design aspects, such as parameters, or component types, that are needed in design decisions, but are unavailable at that point in the design process. The unavailability is often caused by cyclic dependencies and design constraints, where computations cannot be ordered such that all the needed elements are computed in previous design steps.

The agent function types are not necessarily limited to the ones previously described. The final application domain and the scale of the multi-agent system are the factors that ultimately decide the agent types to be included in the system.

All design agents have a restricted area of influence called a “target”. The target represents the design elements that are the object of an agent’s functionality. In parametric design problems an agent’s target can be as narrow as a single design parameter. Several agents, of various functionalities, can have overlapping targets. For example, a component material can be decided by a designer agent, and can be criticized by two different critics.

Design agents can be classified on a third dimension – their domain of specialization. Agents typically group knowledge and heuristics that allow them to reason in a particular domain. For example, two critics that target the material of a component can have different domains of specialization, such as cost or reliability.

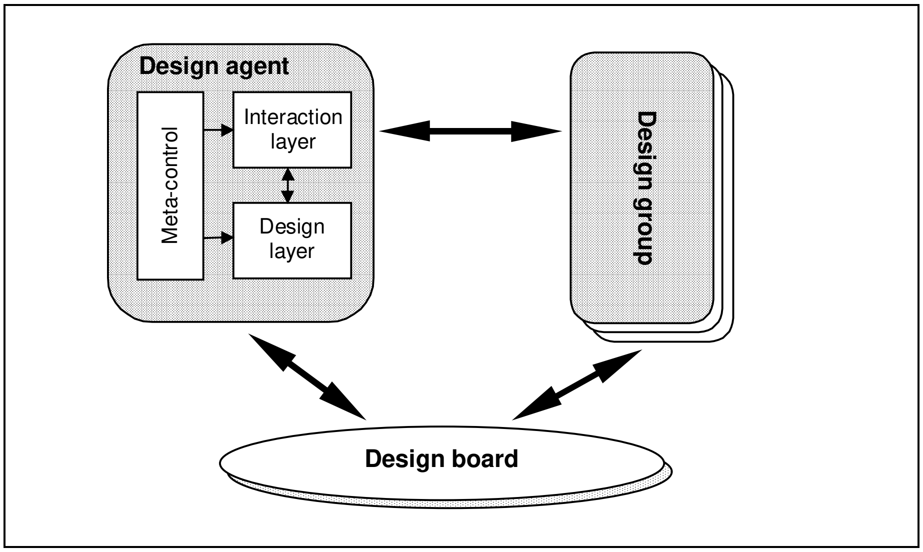

Figure 8. High-level representation of the multi-agent design system architecture

The description given so far covers one of the two main components that make up a design agent – the design layer. The knowledge incorporated at this level gives the agent the ability to function as an independent design specialist. A second layer – the interaction layer – allows the agent to be part of the design group. The interaction layer includes the knowledge that is necessary to communicate, coordinate and reach agreements with other group members (figure 8):

- *Coordination:* Agents act on a task-centered basis. An agent takes on, or 'assumes' a task if it decides it can achieve the requirements of the task. Once an agent has acquired a task, the scheduling of the task and its execution are entirely decided by that agent. An agent can delay the execution of a task if it does not have the necessary resources to proceed (parameter values, critiques of a given decision, etc.). Agent coordination is generated by the computational needs that arise during task execution. The coordination module searches for and acquires the information needed for the agent's computations. The availability of such information can be determined from the current design state or by querying other agents.
- *Conflict resolution:* Conflicts occur mainly due to constraint violations. Previous design decisions may have left no choices for subsequent design aspects that depend on these decisions. The agents that have over-

constrained the design and the agent that cannot proceed as a result of these constraints have to agree on a set of values that allows them all to accomplish their respective tasks. Previous work on single function agents has looked extensively into conflict classification (Dunskus et al. 1995), conflict detection (Berker and Brown 1996), and conflict resolution (Brown, Dunskus, and Grecu 1994). The conflict resolution strategies that were successfully tested in the SiFA environment are reimplemented in the current agent model.

– *Communication* is implemented using a speech act framework with KQML (Finin et al. 1992). Messages include performatives defining the permissible actions that an agent can attempt in communicating with another agent (e.g., 'ask', 'evaluate', 'reply' etc.). The communication is direct, i.e., it doesn't use any intermediary facilitation or mediation agents.

The multi-agent design system is implemented in the CLIPS rule-based environment (Giarratano and Riley 1998). The machine-learning components are based on source code for wrapper techniques and inductive learning included in the MLC++ machine learning library (SGI 1996).

7. THE EVALUATION METHODOLOGY

Given the flexibility in learning that is made possible in our approach, and the potential complexity of the MADS, it is clear that careful attention needs to be given to the evaluation of the impact of learning on the design system (Grecu and Brown 1998b). A number of issues need to be considered for the evaluation.

The first issue focuses on the sets of features in the design environment that are 'perceived' by the agents, that is, the observable world of each agent. The learning result will significantly depend on the features available to the agents. The features in the design environment that are visible to an agent will determine the subset that eventually gets selected by the learning component to represent new expectations.

The causal mechanisms for selecting candidate conditions represent a second major validation topic. Causal mechanisms strongly bias the learning process. A large set of initial candidate conditions will transfer part of the learning bias toward the covariational analysis, and relies on the latter's ability to filter out conditions in a domain-independent way. On the other hand, a restrictive, low-level set of causal mechanisms, that cope with very narrow situations, may forfeit the chance to develop an expectation, given that critical information may be omitted from the very beginning.

The covariational analysis has three factors that influence the final definition of the expectation. The learning algorithm used has to suit the types of features selected by the causal mechanisms, and its selection depends of the size of the training set that is being provided. Second, the candidate conditions can be generated and refined using several techniques, such as hill-climbing. And, finally, various criteria of relevance can be used, including the accuracy and the size of the feature subset. All of the three factors need an estimate and an analysis of adequacy for the class of design problems that are being approached.

The expectation validation process itself has several parameters open to validation. The amount of training instances that are acquired before the expectation is revised determines the 'granularity' of the expectation revision. It has to be coordinated with the number of times expectation revisions are accepted, such that the decision to eliminate an expectation corresponds to the learning limits of the agent with respect to the design environment, and not to weak stabilization criteria.

Although not directly included in the learning process itself, but nevertheless of considerable impact, are the criteria that determine whether and when an agent decides to acquire an expectation. This directly influences the number of learning processes that will be active. It is also critical in preventing an agent from acquiring irrelevant knowledge that would impede on its decision making rather than enhancing it.

A problem directly related to the number of expectations that are simultaneously learned is the potential interference between learning processes, as described in section 5.3. The focus here lies on the differences between a set of expectations learned and revised sequentially, and the same set of expectations learned concurrently, with a lower rate of stabilization.

Finally, learning needs to be evaluated with respect to the objectives of the design problem. The evaluation has to answer whether design aspects have been improved, and/or whether the design process has become more efficient, for example, by generating fewer conflicts, or less backtracking cases.

8. CONCLUSIONS

There are several important aspects of the approach to guiding learning in MADS that have been presented here. The first, and most general idea, is that learning is distributed throughout the MADS, and that learning can occur independently and concurrently.

The second is that the learning activity is temporary, in the sense that learning is active at a particular location in the MADS until the learning result has been validated or eliminated.

The third important aspect is that learning processes shift around the system at run time in response to the agents' information needs. These patterns of activating learning processes are affected by the design problems approached and by their requirements. In addition, given the same design problems, it is clear that the learned expectations will be different depending on the agents that are present in the system.

The manipulation of the expectation learning parameters discussed in the last two sections offers the possibility of experimentation with learning in a MADS. The use of wrappers strongly supports the flexibility of the learning methodology, and allows Machine Learning experiments to be carried out to determine the effect of different algorithms.

We believe that learning in MADS is an important and necessary area of investigation that will ensure the effectiveness and efficiency of future design systems. The approach we have described for guiding learning in multi-agent design systems provides the flexibility needed to take advantage of the power of learning and to dynamically target it to areas that need it in the MADS. We are experimenting with our approach and details of its performance will be described in future articles.

9. REFERENCES

Berker, I., and D.C. Brown (1996). Conflict and negotiations in single function agent based design systems. *Concurrent Engineering: Research and Applications*. Special issue on Multi-Agent Systems in Concurrent Engineering, D.C. Brown, S. Lander, and C. Petrie (eds.) (1):17-3.

Brown, D.C., B. Dunskus, and D. Grecu (1994). Using Single Function Agents to Investigate Negotiations. AAAI-94 Workshop on Models of Conflict Management in Cooperative Problem Solving, Seattle, WA.

Cherniak, C. (1986). *Minimal Rationality*. Cambridge, MA: The MIT Press.

Dunskus, B.V., D.L. Grecu, D.C. Brown, and I. Berker (1995). Using Single Function Agents to Investigate Conflict. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*. Special issue on Conflict Management in Design, I. Smith (ed.). 9:299-312.

Etzioni, A. (1964). *Modern Organizations*. Englewood Cliffs, NJ: Prentice Hall.

Finin, T., J. Weber, G. Wiederhold, M. Genesereth, R. Fritzson, J. McGuire, D. McKay, C. Shapiro, W. Pelavin, and S. Beck. (1992). Specification of the KQML Agent Communication Language: Enterprise Integratrion Technologies, Inc.

Gennari, J.H., P. Langley, and D. Fisher. (1989). Models of incremental concept formation. *Artificial Intelligence* 40:11-61.

Giarratano, J.C., and G. Riley. 1998. *CLIPS Reference Manual*: PWS Publishing Co.

Grecu, D.L., and D.C. Brown (1998). Dimensions of Learning in Design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*. Special issue on Machine Learning in Design, A.H.B. Duffy, D.C. Brown, and A.K. Goel (eds.) 12 (April):117-122.

Grecu, D.L., and D.C. Brown (1998). Evaluating the Impact of Distributed Learning in Real-World Design Problems. Workshop on Machine Learning in Design (5th International Conference on Artificial Intelligence in Design), Lisbon, Portugal.

Hilton, D.J. (1990). Conversational Processes and Causal Explanation. *Psychological Bulletin* 107 (1):65-81.

Kohavi, R., and G.H. John (1998). Wrappers for Feature Subset Selection. *Artificial Intelligence* 97 (1-2):273-324.

Koslowski, B. (1996). *Theory and Evidence: The Development of Scientific Reasoning*. Edited by L. Gleitman, S. Carey, E. Newport and E. Spelke, *Learning, Development, and Conceptual Change*. Cambridge, MA and London, UK: The MIT Press.

Lander, S. (1998). Issues in Multiagent Design Systems. *IEEE Expert*:18-26.

Liu, H., and R. Setiono (1998). Incremental Feature Selection. *Applied Intelligence*.

Russell, S., and E. Wefald (1991). *Do the Right Thing - Studies in Limited Rationality*. Cambridge, MA: The MIT Press.

SGI (1996). MLC++ Utilities: Silicon Graphics.

Shultz, T.R., G.W. Fisher, C.C. Pratt, and S. Rulf (1986). Selection of Causal Rules. *Child Development* 57:143-152.

Tversky, A., and D. Kahneman (1974). Judgment under uncertainty: heuristics and biases. *Science* 185:1124-1131.