MultiADD: A Multiagent Active Design Document Model to Support Group Design

Adriana Santarosa Vivacqua and Ana Cristina Bicharra Garcia

ADDLabs – Departmento de Ciência da Computação
Universidade Federal Fluminense - Praça do Valonguinho, s/n
Niterói, RJ, 24210-130, Brasil
e-mail: avivacqua@ax.apc.org, bicharra@dcc.uff.br

Abstract

In this paper we discuss the use of multiagent systems to assist the design task of engineering artifacts. We augmented the active design document approach to assist the design activity when done by a group. Task division, information flow and conflict management are the main issues when working in a group. This paper reports initial results on applying our multiagent active design document system (MultiADD) to support conflict mitigation in group design. The discussion focuses mainly on the questions related to information flow: what, when and to whom to inform while in a conflict situation. We present our model and an example taken from our implemented system, which uses this technology to support process plant design of offshore oil platform. The system has been used by the Brazilian Oil Company. In addition to speeding up the process due to the design support tool, meetings and scheduling times have been greatly reduced. Consistency among the design pieces and an increase of alternative checking have also been noticed. Most importantly we expect that the environment encourages cooperation among design participants.

Introduction

Design of complex systems generally involves a group of people developing different parts of the artifact. The group can behave in a cooperative or competitive mode depending on the social structure involving them. In this paper we focus on groups that works in the same company sharing the same overall criterion—the best for the company; in other words, cooperative design. For instance, the design of an oil process plant involves a team of about ten engineers. Each one of them is responsible for developing one or more subsystems such as the oil separation, the gas compression and the oil transference subsystems. The design subsystems must be integrated to become a process plant. There are many conflicts among the subsystems' decisions that are usually discussed and solved eventually in meetings.

In earlier days, designers worked in standalone computers, sharing their work in meetings using paper documents as the media to support their arguments. Disagreements always occur based on conflicting decisions done by different participants or misconceptions over the other participants' decisions.

The lone worker is no more. The dissemination of communication and networking technologies has created a new work method. Nowadays, workers have the means to work together over a computer network. The time required for meetings can be significantly reduced if designers are already connected by a tool which can aid them in resolving their differences and provide them with the necessary information at the correct time.

We combine the computer network availability to spread information, the active design document system to support individual design work, and multiagent systems to combine the individual design parts. In this paper we present our multiagent active design document model (MultiADD) to support group design. We illustrate our discussion presenting an implemented version of MultiADD for the domain process plant design of offshore oil platform (ADDProc) that has been used for over a year.

In the following section we discuss the main issue on group work, conflict mitigation. In the following section, we introduce background information to settle the grounds for presenting next our model. After presenting MultiADD, we illustrate it with a working example. We conclude with promising results showing the impact on the design process time.

The Conflict Mitigation Problem

Whenever a number of people have to work together, conflicts are bound to appear. Even if these people have a common goal, they often have conflicting local goals.

Usually, designers work individually in their own task, and in group to discuss their decisions and settle down their differences. In addition to the enormous amount of time spent scheduling meetings, it has been noticed [Olson et al., 92] that most of the time consumed in group design meetings is spent with clarification; i.e., designers explaining what they did and why. Consequently, we claim

that clarification is the basis of cooperative behavior.

However important information may not be presented during a design meeting either because the issue has not been thought or even questioned yet, or the issue was closed a long ago. In addition to these *timing problems*, getting the entire group's attention all the time is hardly achieved. Parallel conversation may cause relevant discussions to be overlooked. Besides, although in a cooperative environment, designers are evaluated by their individual design work. Consequently, they tend to resist to changes in their own design part.

We address these problems by creating an environment for group design that assists the individual design work, group coordination and design integration. This environment stimulates cooperation by making available the rationale for any decision done by each design participant. It saves meeting time and rework necessity by distributing relevant information to each designer at the appropriate moment. Thus, each designer receives only information that affects his work whenever needed. Design time decreases, designers get more focused in their job and meetings become objective.

Background Technologies

In this section we present the two technologies involved in our work, the Active Design Documents (ADD) model and Multiagent Systems (MAS). The ADD model is particularly useful as a design and documentation support system, while MAS provides the framework for group design and information sharing.

ADD

In the design activity, documentation is very important. However, it is expensive, extensive, generally incomplete, inconsistent or inaccessible. During the design process, most of the designer's rationale for an artifact is lost. Many design rationale tools have been developed based on hypertext technology. It's use, however, implies a high document construction time and difficult rationale retrieval. Active Design Documents [Garcia, 92] is a model—based approach that integrates design and documentation activities. It provides a design support system while documenting a project. By documentation we mean both data and rationale able to fully explain a specific design developed by a designer. As the project is developed, ADD documents the project decisions, with little or no overhead for the designer.

An ADD agent contains a domain model that enables it to generate expectations about designers' decisions. It follows, records and criticizes the designer's actions in order to generate explanations for them. Whenever the documentation agent cannot explain the designer's decisions, it requires the designer to change its domain knowledge base. As a consequence, ADD's domain model always reflects the designer's domain model.

ADD acts as an *assistant* and *apprentice* to the designer. Whenever the designer proposes a design action that differs from the apprentice's expectations, the interface will ask for the designer for justifications to explain the differences. Subsequent queries for design rationale are answered using a combination of the interface's domain knowledge and the designer-supplied justifications. It is important to notice that ADD acts as an auxiliary and not as substitute for the designer. So, the designer still plays an important part in the project. In addition, ADD offers the possibility of making design simulations in order to evaluate different solutions. Even though efficient, ADD was developed to assist a sole designer in a singular domain. To deal with group design, we expanded ADD importing Multiagent technology.

Multiagent Systems

In multiagent environments, various specialist systems work together to reach a solution. To accomplish their tasks, they need to interact and cooperate with each other. There are two main types of cooperation between agents: information exchange and service exchange [Oliveira & Qiegang, 91]. In the first, the agents distribute their results so that others may use them and in the second the agents contract one another to perform tasks. We concern ourselves exclusively with information sharing, since each designer has a specific task that cannot be performed by another.

Blackboard and Contract Net architectures [Wellman, 95] were studied to deal with the communication problems inherent to the environment. In the first, there is a central agent that controls communication, monitors conflicts and assigns tasks when necessary [Bobrow, 91]. The knowledge about the agent's needs and abilities is centralized in one sole agent. This core agent may become a bottleneck with communication growth. However, it has the advantage of knowing the agents' intersection models and of managing the information flow thus leaving the other agents to their own tasks and models (freeing them from these tasks). In the second architecture, communication is performed by all agents [Edmonds et al., 94] [Genesereth & Ketchpel, 94]. This architecture requires each agent to have knowledge about the others and negotiation abilities. Consequently, the growth in the number of agents implies an overload of information processing needs by each agent, hindering the work it was originally supposed to do [Bobrow, 91].

As in any society, there is a need to identify each agent's group behavior. Benevolent, malicious, cooperative and competitive were some of the terms coined to refer to the agent's social behavior [Rosenschein & Zlotkin, 94]. These issues have been studied in highly distributed environments (such as the Internet) where, more often than not, the real intentions of the agents are unknown.

The likeliness that conflicts will appear increases with the difference between objectives, tasks and knowledge. Conflict identification and resolution are an important part of multiagent systems. Our focus is on cooperative

multiagent systems, where human and computational agents interact to reach a common goal. The instrument to encourage cooperative behavior is the communication of the rationale of the other agent when a conflict occurs.

We use the cognitive approach to MAS that deals with complex intelligent agents in a society [Sichman, Demazeau & Boissier, 92]. In MAS, the metaphor is social behavior and the emphasis is in the actions and interactions between agents. Most research on MAS considers purely computational agents, without the interference of external users. Our approach contemplates human participation in the multiagent environment. Social interaction between agents may also influence the resolution process of a conflict [Finger, Konda & Subrahmanian, 95].

MultiADD Models

We extended ADD's model to enable cooperative work, using principles of Multiagent technology. We focus on aiding conflict resolution through communicating design rationale among users. We implemented a centralized version of MultiADD in which the core agent (the Controller) deals with the issues: *what*, *when* and *to whom* to communicate an information. This agent is able to identify the appearance of a conflict and determine the importance of each information to each designer at a given moment.

Since we are dealing with a mixed environment (one with computers and human designers), the conflict resolution process is much more complicated than in a purely computational one. The environment we have created assists them with the resolution, encouraging negotiation and discussion, but does not resolve the conflicts for them, instead letting them work it out for themselves. The next stage of our research will be to observe how conflicts are solved in this environment and build some conflict resolution schemes to deal with them.

In this section we present our Multiagent Active Design Document model (MultiADD) to support group design activities. We start presenting the domain knowledge representation used (parametric dependence networks) followed by MultiADD architecture.

Knowledge Representation

Domain knowledge is represented by a parametric dependency network, a graph on which parameters and dependencies are described. There is one type of link (depends on) and there are three types of parameters: primitive, coming from user input; derived, obtained through a formula; and decided, obtained through a rational decision making process (i.e., evaluation of alternatives through a utility function).

In our application domain, the task is subdivided in nineteen subtasks, each requiring a specific expertise to be accomplished. The domain and the expertise required to solve problems relating to the subtasks are represented in a parametric dependency network. Fortunately, the application domain is not a highly connected graph. Consequently, we can map the domain into subgraphs (representing the subsystems), were the parameters are clustered. Even so, there are still parameters connecting these subgraphs (which are usually part of both subsystems), which we consider to be the intersection between the domain areas and the source of conflicts.

For example, designing an oil separation subsystem of a process plant involves selecting and configuring a set of equipment such as oil heaters, and oil separation vessels. There is a set of criteria ruling the process and a number of input data (primitive parameters) constraining it. Figure 1 illustrates the parametric dependency net for this domain. A subsystem's complete network has around 1000 parameters.

In MultiADD, each agent has a parametric dependency net for its own domain. In addition, there is a net containing the domain intersection. Depending on the type of coordinating schema this model becomes available to one or more agents. In the centralized version of MultiADD, only the Controller agent has access to it.

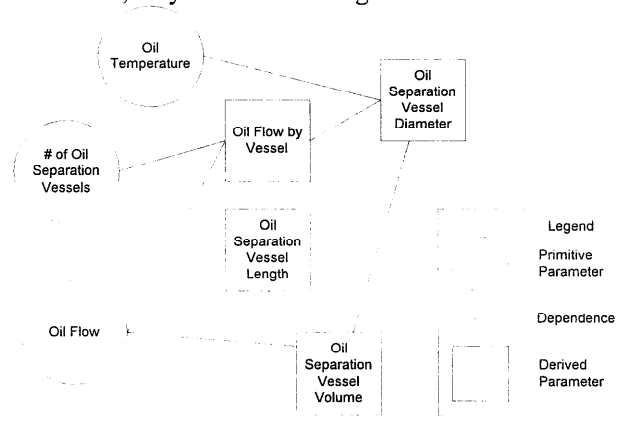

Figure 1: Partial parametric net for the Oil Separation subsystem

The net is particularly useful to assist understanding the complexity and impact of a requested change in someone's work. The instantiated net nodes represent the designer's work done. The net connectivity degree indicates the domain complexity. For instance, if a change is requested in a highly connected area that has been all instantiated, the time spent changing the design will be higher than in a place where no commitments were taken yet. Besides technical issues, the net may indicate the agents social behavior. For instance, it can track if an agent never yields indicating a non-cooperative behavior (assuming all agents equally technically prepared).

ADD Agent

In MultiADD, an agent is the tuple <assistant system, user>, called Design Team. The Design Team develops its design and documentation part to be shared and discussed by the design group following ADD's model. Each agent is

an independent ADD system with a specific knowledge base represented by a parametric dependency network We emphasize the existence of human agents because they are the main source of conflicts, as well as, the main reason for building a support system. Were we working with merely computational agents, it would be easier to predict their behavior and guide them to a solution for the problem.

Controller

In the centralized MultiADD version, Design Teams communicate changes through the *Controller* agent. The Controller's roles are to filter the information to be "heard," as well as, to select its "listeners." The Controller spreads the minimum information to keep designers up to what may interest them This piece of information may trigger designers to open each others' explanation system (a component of any ADD system) to better understand the situation.

The Controller may communicate: conflicting parameters, values, rationale and involved agents; before, after or while informing something else, at a design stage, in a specific time, always or never; to everybody, nobody, to the coordinator, or to involved agents.

The Controller deals with potential pieces of conflict information mapped to parameters and their respective domain. Anytime different values are assigned to the same parameter, a potential conflict is detected, and the involved agents may be alerted. However, not to become too intrusive the Controller also decides when to communicate the potential problems.

The Controller has a global the domain independent knowledge base (also represented in parametric nets) containing the following parameters used for its decision-making process.

- Agent's State: whether active, inactive or inexistent
- Conflicting Parameters List
- Degree of work done: how much work has already been done by the agent
- Local impact: number of parameters impacted by each parameter on the subsystem as a whole
- Parameter Exploration Degree: how many alternatives the designer has tried
- Variable State: whether a parameter is in conflict.
- Conflict Duration: how long the conflict has persisted.

There are other parameters, which we use for the purposes of group coordination and do not mention here.

Conflict Identification

Conflicts occur due to differences on points of view caused by differences on backgrounds, decision abstraction, vocabulary, understanding, goals, criteria evaluation, and personal attitudes [Oh & Sharpe, 95]. In any case, understanding is the key to solve any conflict.

Identifying a conflict is the first step to solve it. In the MultiADD environment, a Design team is always setting parameter values. Consider two Design Teams, A and B deciding over a common parameter X. There are **three** possible situations diagnosed by the Controller illustrated in Table 1.

A's value	B's value	Conflict?
80	80	NO
80	70	YES
80	--	NO

Table 1: Possible situations diagnosed by the controller

Even when an agent is not participating on a conflicting situation, it may be notified to prevent the situation from worsening.

Conflict Communication

The conflict information consists on the values proposed for the parameter by each of the subsystems plus the rationale explanations and any notes and observations the other designers might have made. We believe that the designers will act rationally and, with the understanding of the conflicts, global objectives and each other's reasoning, will be in a position to negotiate and spontaneously reach a solution.

There are three types of information to be sent:

1. A list of the Conflicting Parameters;
2. A list of the Conflicting Parameters, Values and Agents;
3. A list of the Conflicting Parameters, Values, Agents and Rationale.

At the beginning only type 1 information is communicated. When a conflict still remains, type 2 information is sent. In deadlock cases or upon request, type 3 information is provided.

All agents involved in a conflict are potential receivers. The communication moment depends on the agent's task at the time and the transmitted information importance for the agent's domain.

The Controller uses the aforementioned parameters to decide when is the right moment to send a piece of information. Some heuristic rules have been used to guide the Controller's behavior, such as:

- Agents working on a conflicting parameter's neighborhood (parameters three or fewer steps distant from the given parameter), must be warned immediately.
- Agents working on areas not affected by a conflict should not be disturbed.
- High impact changes should be avoided. The Controller warns the low impact subsystems first, trying to minimize the overall rework.
- Completed subsystems should be the last ones to be disturbed.
- High duration conflicts (defined by the number of cycles) must be informed to coordinator.
- Always send immediately to inactive agents.
- Coordinator's imposition is immediately sent to all affected subsystems and the agents must obey.

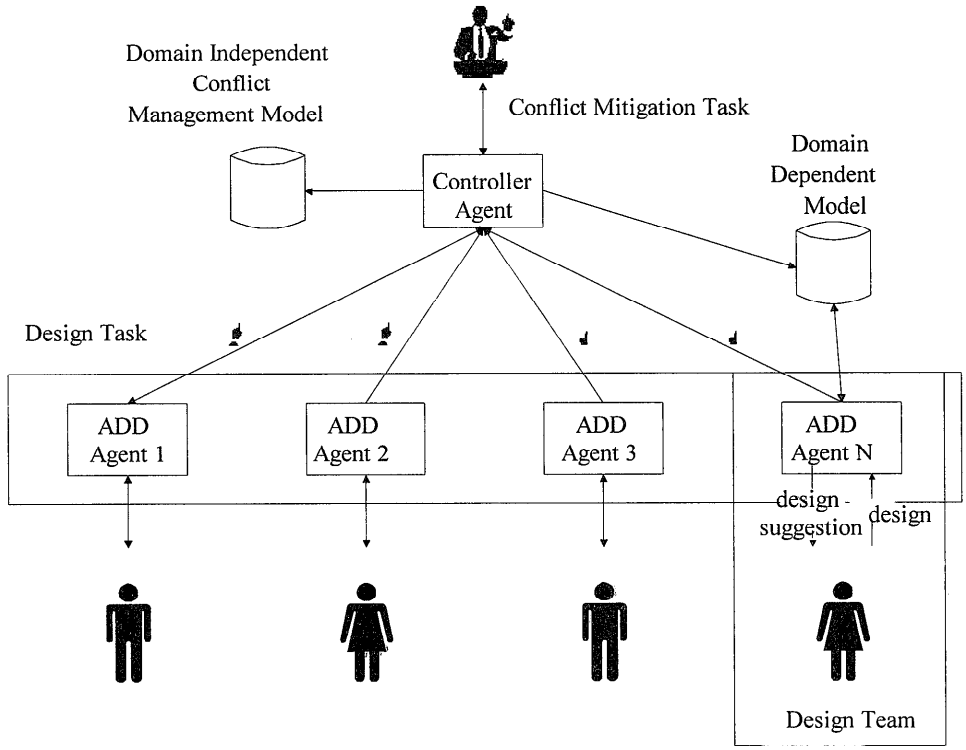

Figure 2: Centralized MultiADD Architecture

The conflict information should not be communicated only once. If a designer has already been warned but remains in conflict, he must be notified again. Thus, it becomes important to establish the time between warnings, to avoid disturbing the designer with continuous interruptions (which could lead to him dismissing the warnings). Communication is a sensitive issue and the above rules are still being refined.

ADDProc: an Example of a Centralized MultiADD System

In this section we present our system, ADDProc, which implements MultiADD's model to the domain of process plant design of offshore oil platforms. ADDProc is an intelligent design support tool for oil production processes. We have implemented a centralized version of the MultiADD system for the domain of process plant design for offshore oil platforms. We chose to implement a centralized architecture (as seen in Figure 2) because the application domain itself was centralized.

The ADDProc system was implemented using C for the decision making reasoning, Intellicorp's Prokappa for AIX for the knowledge bases, IUP and LUA libraries, from PUC-Rio for the graphical user interfaces and Oracle version 7.0 for the recording of data from previous cases. It runs on IBM RISC 6000 machines and an Intergraph server 6700 hosts the database.

The Example Application Domain

The petroleum comes from the reservoir as a mixture of oil, gas and water. This mixture needs to be separated and treated before being exported to the refineries. A process plant project contains the set of equipment that will transform the incoming petroleum in the desired oil and gas condition.

Typically, a process plant contains 19 different subsystems (such as Oil Receiving, Oil Heating, Oil Separation, Oil Treatment, Oil Transference, Gas Compression, Gas Treatment, and Water Heating subsystems). Designers of each discipline select and configure equipment in compliance with a set of requirements to achieve a final goal, which is the production and exportation of gas and oil. Each project is coordinated by a senior engineer, responsible for getting the work done within the deadlines.

To illustrate the design process of each discipline and the conflicts that appear among them, we develop an example of the Oil Heating and Separation subsystems. The oil comes from the reservoir in a low temperature. Generally, this temperature is not sufficient to allow a good separation of oil, water and gas. The Oil Heating subsystem designer is responsible for selecting equipment to increase this temperature. Often the cost-effective equipment selection provides a close final temperature, but not the one required for the Oil Separation system.

The Oil Separation system designer is responsible for

selecting a set of production separators that will actually separate oil from water and gas. The dimensions of this equipment are a function of the time the petroleum will take to pass through the equipment and its temperature. The higher the temperature or bigger the equipment, the easier the separation. Since minimum platform area is an overall requirement, the oil separation subsystem designer tries to reduce the equipment length by increasing the temperature. Of course the equipment's material resistance is another issue to be considered. The oil temperature parameter is a good example of the conflicts that may appear between design agents.

ADDProc

We mapped the 19 domains into parametric dependency graphs. Each one represents the task involved in one system. The global design model is also represented by a parametric dependency graph. Each of the agents is an independent ADD system with its specific domain knowledge and designer (forming a Design Team). Thus, each agent is, in its own, a design tool for a certain part of the system. Additionally, there is a special agent, the coordinator, which is responsible for the overall process plant design, with the authority to solve any conflict. A central Controller module takes care of the communication among the agents, verifying when conflicts happen and when they are resolved, so as to inform the agents as necessary.

Suppose the controller detected a conflict between the Heating and Separation subsystems, over the oil temperature parameter. The controller receives the updates and verifies an inconsistency between the proposed values for the oil temperature parameter. Consulting its domain dependent and independent knowledge bases, it verifies that this paramcter has been in conflict for too long (heuristically inferred by the number of cycles). In addition, this has a high impact on both the Separation system and the Separation designer's work is almost complete (90%). Moreover, the Separation designer agrees with the formal ADD's model, while the Heating designer has imposed a value over ADD's expected value. Both are high credibility agents and hierarchically equal. For these reasons, the controller first sends the conflict information to the Heating system, trying to persuade the designer to change the proposed value. If the conflict persists for more n cycles (defined as seen fit by the project's Coordinator), the controller contacts both systems trying to persuade any of them to do so. Finally, if they are still unable to reach an agreement, the controller contacts the coordinator showing the conflict. In this case, it suggests the Coordinator's support the Separation system's solution. There is a Coordination Interface where the Coordinator can study and decide upon a conflict.

Conclusions

In emphasizing the human designer, we proposed a new approach to MAS, using a "mixed" system, where human and machine work together as one. Designers develop their designed assisted by a design support system in a group support environment. Design models are created guaranteeing consistency within and among design pieces.

Initial results have shown a potential decrease on design meetings and design meeting duration. The design support system has allowed an increase on the number of alternative designs. We expect that the greatest impact will be on the overall work cycle. Time spent scheduling meeting or gathering information can be drastically reduced from days to minutes.

References

[Bobrow, 91] - Bobrow, Daniel G. - AAAI-90 Prcsidential Address - Dimensions of Interaction - *AI Magazine*, Fall 1991

[Edmonds et al. 94] - Edmonds, E.; Candy, L.; Jones, R. & Soufi, B. - Support for Collaborative Design: Agents and Emergence - *Communications of the ACM*, July 1994

[Finger, Konda & Subrahmanian, 95] - Finger, S.; Konda, S. & Subrahmanian, E. - Concurrent Design Happens at the Interfaces - *AIEDAM* 9, 88-99 1995

[Garcia, 92] - Garcia, A.C. - Active Design Documents: A New Approach for Supporting Documentation in Preliminary Routine Design - Ph.D. Dissertation - Stanford Univ., August 1992

[Genesereth & Ketchpel, 94] - Genesereth, M. & Ketchpel, S. - Software Agents - *Communications of the ACM* - July, 1994

[Klein & Lu, 90] - Klein, M. & Lu, Stephen C.Y. - Conflict Resolution in Cooperative Design - *AIENG* 4 (4), 168-180, 1990

[Oh & Sharpe, 95] - Oh, V. & Sharpe, J. - Conflict Management in Interdisciplinary Design - *AIEDAM* 9, 247-258 - 1995

[Oliveira & Qiegang, 91] - Oliveira, E. & Quiegang, L. - Towards a Generic Monitor for Cooperation in Workshop on Blackboard Systems of the AAAI, Anaheim, California, 1991

[Olson et al. 92] - Olson, G.; Olson, J.; Carter, M. & Storrosten, M. - Small Group Design Meetings: An Analysis of Collaboration - *HCI*, Vol 7, 1992

[Rosenschein & Zlotkin, 94] - Rosenschein, J. & Zlotkin, G. - Designing Conventions for Automated Negotiation - *AI Magazine*, Fall 1994

[Sichman, Demazeau & Boissier, 92] - Sichman, J., Demazeau, Y. & Boissier, O. - When can knowledge-based systems be called agents? in Anais do 9º Simpósio Brasileiro de Inteligência Artificial - SBIA'92, Rio de Janeiro, Brasil, 1992

[Wellman, 95] - Wellman, M.P. - A Computational Market Model for Distributed Configuration Design - *AIEDAM* 9, 125-133, 1995.