From: AAAI Technical Report WS-97-09. Compilation copyright © 1997, AAAI (www.aaai.org). All rights reserved.

Multiagent Active Design Documents in Group Design

Adriana Santarosa Vivacqua and Ana Cristina B. Garcia

ADDLabs - Departamento de Ciência da Computação
Universidade Federal Fluminense - Praça do Valonguinho, s/n
Niterói, RJ, 24210-030, BRAZIL

e-mail: avivacqua@ax.apc.org, bicharra@inf.puc-rio.br

Abstract

Design of engineering artifacts is a complex task, which is usually taken up by a group of designers. Usually, the problem is subdivided and distributed among them, and they must work together, sharing information and decisions so as to accomplish the final goal. The mere existence of a common goal should make this a cooperative process. However, the fact that designers have different tasks and local goals hinders the cooperative process. In this paper we discuss issues related to the incentive of cooperation and coordination of designers working in a group. How can we make designers cooperate among each other and how can we make the whole design process more efficient? We have created a collaborative design environment in which designers work on their individual portions of the design and share information among themselves. Cooperation is encouraged by showing the designers each others' rationales. The model has been implemented for the domain of process plant design for off-shore oil platforms and is currently being used by engineers at the Brazilian Oil Company.

Introduction

Advances in communication technology have created the necessary environment to allow people to work together. It became possible to allocate expertise to develop a project from people geographically apart, without moving them from their own workplaces.

Group design; i. e., design projects developed by a group of people, is a real domain example impacted by this new technological setting. This activity is essentially cooperative. Designers contribute with their own expertise to a part of the whole, competing when conflicting local optima occur, but yielding to reach a set of global goals. Actually, these global goals are the ones that make them share information, cooperate with each other and act as a group.

Whenever a group of people is put together, conflicts are bound to appear. Therefore, a control and coordination structure is needed to ensure that the work gets done. Moreover, a cooperative behavior stimulates conflict resolution. Cooperation can be achieved through common understanding of the issues impacting the whole and the portions of a design.

In this paper we discuss the use of a Multiagent Active Design Document System (MultiADD) applied to an engineering group design domain: a process plant design of offshore oil platform. Next, the issues on conflict mitigation is discussed, followed a presentation of the MultiADD architecture. Afterwards, an example from the implemented version of MultiADD (ADDProc) illustrates the way the system works. Finally, a comparison with other multiagent systems is presented.

The conflict mitigation and coordination problem

In our work we focus on conflict mitigation in cooperative environments. Designers develop their work mainly looking at their objectives, which may lead to conflicting situations. However. there is a set of common goals leading them to negotiate when conflicts occur.

In engineering domains, designers work in different parts of a project that are later assembled together to form the desired artifact. The existence of a common goal should make it a cooperative process, however, that is not always the case. As designers are evaluated by their individual work, they tend to concentrate on their individual parts, searching for their local optima and overlooking the global optimum. However, this situation can be reverted through understanding of each others work. Design rationale sharing is a way to provide this deeper comprehension.

The Cairo project at MIT [7], and Network-Hydra [1] are examples of work on group design support systems. They provide an environment where people describe and share their decisions' rationale. Like in hypertext design rationale tools, the users are responsible for describing and recording their own rationales, which become available to other participants. They are also responsible for retrieving relevant information to their work.

Copyright © American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

Technological Background

To create a collaborative design environment, we have combined Active Design Documents (ADD) and Multiagent Technologies. In this section we present ADD and Multiagent Systems and in the next MultiADD, our collaborative design environment.

ADD

The Active Design Document (ADD) model proposed by Garcia [2] is an environment in which designers develop and document a project. The documentation consists of 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 follows, records and criticizes the designer's actions in order to generate explanations for them. It is able to generate expectations about designers' decisions and whenever it cannot explain these decisions, it asks the designer to change its domain knowledge base. In this fashion, ADD's domain model always reflects the designer's domain model.

The three macro elements contained in ADD's architecture are: *interfaces, design knowledge base* and *reasoning components*. These are explained below. The design process consists of proposing values for design parameters, adjusting initial requirements and generating more requirements or design parameters.

- *interfaces*: allow user interaction with the system. Permits the user to and provide a solution for a case, obtain a design explanation and change ADD's model (Design, Explanation and Knowledge Acquisition Interface, respectively).
- *design knowledge base*: contains knowledge about the domain, the decision making process and previous cases. This information is the basis for creating ADD's decision expectation.
- *reasoning components*: the anticipator, reconciler, knowledge elicitor, rationale generator and control are responsible for generating design decisions and comparing those with the designer's, querying the user for additional knowledge when needed, preparing design reports, and controlling the documentation process.

ADD uses an *apprentice* metaphor: whenever the designer proposes a design action that differs from the apprentice's expectations, it will ask the designer for justifications to explain the differences. Subsequent queries for design rationale are answered using a combination of the domain knowledge and the designer-supplied justifications. It is important to notice that ADD acts as an auxiliary and not as substitute to the designer, with the designer still playing an important part in the project. 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. They must exchange information and services [4] among themselves. "Blackboard" and "Contract Net" architectures [8] were conceived to deal with the communication problems inherent to the environment. Our initial implementation of the system is centralized and we intend to decentralize it as a second step in our research.

As in any society, there is a need to identify each agent's group behavior (benevolent, malicious, cooperative and competitive where some of the terms coined to refer to the agent's social behavior) [3] [6]. These items have been studied in highly distributed environments where, usually, 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 is 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.

Although much has already been studied, most research in MAS deals with purely computational agents, without the interference of external users. Our approach takes into account the existence of the user (designer). We consider an agent as being the tuple <assistant system, user>, which we call a Design Team (we'll come back to that later). It has already been noted that the social interaction between agents may also influence the resolution process of a conflict [2]. This becomes particularly important in environments such as ours, where the user is an active part of the system.

MultiADD

One of the main issues when dealing with cooperative work is conflict management. Because it involves a group of people, there is a high propensity to conflict appearance. Normally, these conflicts are discussed in group meetings, which involve negotiations among designers. A study by Olson [5] has shown mostly, design meetings are consumed by clarification discussions, i. e., time is spent justifying each others decisions. Besides, it is often the case a quarrel due to missing information. Worse than discussing the issue in a early stage, is to miss the issue leading to a later rework. These timing issues is worsened with time spent scheduling meetings and designers listening to a great amount of irrelevant information. Any group design assistant tool must to deal with the issues summarized below:

- what to inform;
- when to inform;
- to whom to inform a piece of information; and
- how to assist coordination.

These issues are discussed in the following sections.

Design Teams

The domain models for each of the ADD agents is represented by a parametric dependency net. The net is particularly useful to us because we can extract information from it for later use, track the designer's work and explain it when necessary.

In order to emphasize the human designer's part in the process, we have coined the term *Design Team*. A Design Team is comprised of an ADD agent and a designer. The human designer is important to us because he is the one making the final decisions, solving the conflicts and getting the design done. The ADD agent assists and documents the process, but the designer has the last word on everything being done. In this fashion, in a design team the designer is a specialist on a particular discipline and responsible for the design of one of the subsystems of the process plant. The ADD agent is specifically built for the discipline in hand.

Each ADD agent consists of a case base specific for a discipline and a specialist system that works with the designer's and the case base's values. An explanation is generated for each parameter, in case the designer wishes to refer to it (or another designer wishes to understand the rationale of the first one). The knowledge acquisition module is activated when the designer takes an action that the ADD agent cannot understand or when he wishes to alter the initial ADD agent's model.

While working, the Design Team must interact with other Design Teams working on the same project because their domains are overlapping and more than one designer may manipulate the same variable. Due to these interactions, the Design Team gets involved in many conflict situations (involving these common parameters). Normally, the designers would meet to discuss their differences. However, much time is spent in the scheduling and meetings which needn't be. Moreover, studies have indicated that most of the meetings' time is spent with clarification, i.e. designers explaining their rationale to each other.

By volunteering the information when its needed, while the designers are working, we cut down on the number of meetings and much of the time spent in them. By providing the necessary information at the correct time, we give designers the means to verify its impacts on the other subsystems. In this way, they can evaluate their work as t is being done, try different alternatives and seek to resolve the conflicts as they appear.

Besides the domain knowledge, which is shared with the group, each Design Team has information about itself which is also distributed. This knowledge includes (but is not limited to):

- domain complexity: represent the number of nodes affected by a particular parameter
- decision introspection: number of alternative values studied for the conflicting parameter
- design completeness: percentage of design already done

This information is taken from each subsystem's parametric net and used to decide when is the best time to send some information to a Design Team and what weight the information of a particular Design Team carries. For example, we assume that if a designer has worked extensively on a parameter (high decision introspection), worked for a long time or has a finished the greater part of his design (high design completeness), it has given his work reasonable thought and his opinions should carry more weight than those of a Team which hasn't worked as much.

In order to share this information, there is a need for an extra module, to take care of the communication issues involved. Therefore, we created a module which we have called *controller* to manage the information exchange needs generated by the existence of a number of different agents working together.This is presented in the next section.

Controller

The Controller is a special module which can have one or more Design Teams attached to it. figure a shows the Design Team and its communication with the Controller. It manages the information entering and leaving the subsystems. Its main roles are to identify conflicts and distribute information and orders among the agents.

The MultiADD system can either be centralized (having only one controller module) or decentralized (having one controller attached to each Design Team). the different architectures indicate the agents' hierarchical power and depend on the information visibility (sometimes it might be more advantageous to have groups of Design Teams with certain common characteristics put together).

In the case of a centralized architecture, the Controller receives Design Team requests from time to time (acting upon request) and, in addition, it gathers information from all the agents with which it is connected from time to time (acts on a schedule). Besides that, it keeps track of the time being spent on the project as a whole and by each Design Team and gathers information necessary to project coordination

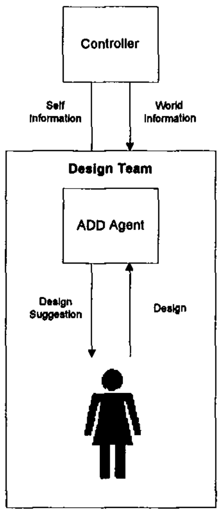

Figure A: Design Team and Controller

The Controller has a global model (also represented by a parametric net) of how the subsystems intersect, on which it bases its decisions. In addition to the information sent by the Design Team, the Controller uses the following information:

- who are the players; i.e. which agents are involved in each project
- which parameters have conflicting values
- domain intersection model: Design Teams affected by each conflict information
- for how long (real time or in number of cycles) a conflict has remained
- for how long each Design Team has participated each conflict
- the project's schedule
- the behavior of each Design Team related to the overall conflicts. This information comes from the observation of each Design Team's behavior (which designers try to negotiate when in conflict).

Based on this information and a set of heuristic rules gathered from the application domain, the controller defines:

1. What information to send;
2. When to send it;
3. To Whom to send it.

As it receives the values of the different Design Teams, the controller determines which are in conflict (for each conflict variable). These are potential receivers of conflict information. After that, based on how much a parameter affects a subsystem, how much work the Design Team has already done and the conflict's status (how long it has persisted, how many Design Teams are involved, how different are the proposed solutions), it decides to whom to send the conflict information.

Based on what the designer is working on at the moment (at what point in the parametric net he is), the influence of the parameter over the net (especially over what has already been done and might have to be redone if changes occur), how much time has already been spent on this particular conflict and how much is still left to the project's deadline, it decides when to inform a Design Team of a conflict situation.

The contents of the conflict information might be: the parameter values proposed by each Design Team and their rationale explanations (which are automatically generated by each ADD agent), plus notes or observations made by the designer. Given the necessary understanding of each other's reasoning, the designers will act rationally and spontaneously engage in a negotiation process, seeking to find a solution to their problems.

In different situations, different types of conflict information are given. The conflict information may consist of:

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 and Agents and Rationale.

At an initial moment, type 1 information is communicated. As time passes, type 2 information is sent. In deadlock cases or upon request, type 3 information is provided. figure b presents the MultiADD model.

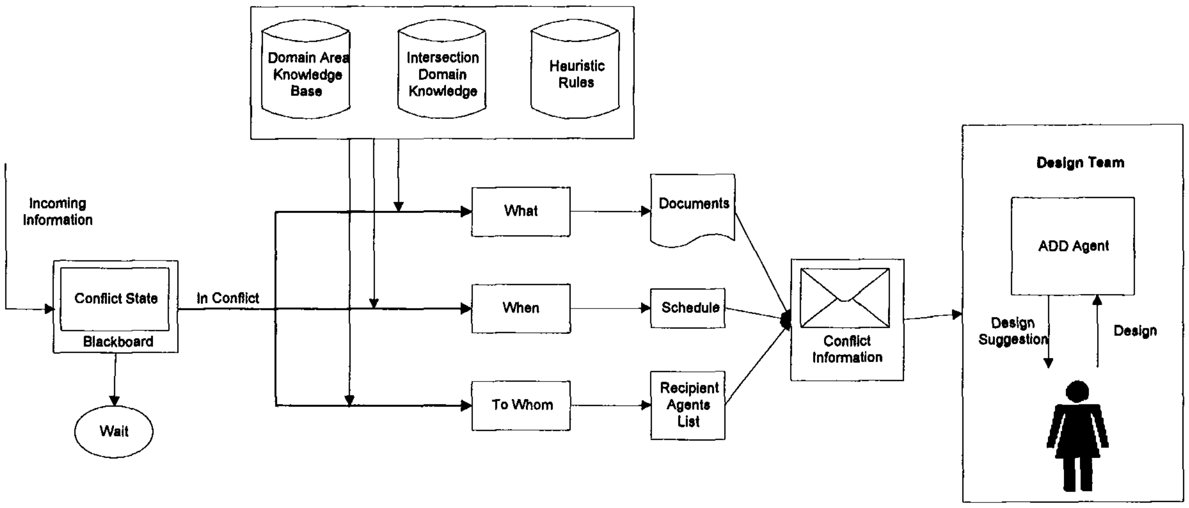

Figure B: The MultiADD model

Coordinator

A central part of any group activity is coordination. Whenever a number of persons are put together, there is a need for a certain amount of coordination. The parts need to be put together in a coherent fashion and the designers must work in an orderly way. To ensure the group will attempt to the global optimum, a restricted budget and schedule, coordination is necessary.

The Project Coordinator (PC) represents an individual or a group of individuals responsible for overseeing the whole project and assuring the deadlines and budgets.

Coordination may be effected by one sole individual (centralized), by groups of individuals (by committees) or by everyone (democratic). These different coordination modes have the same information needs. It is a matter of making the information visible to the right person at the right moment (being non-intrusive without omission). In the event of committees or democratic configurations, there is a need for a discussion structure. The members of this privileged group need to discuss to reach an agreement regarding the conflict. Actions taken by the Coordinator includes: suggest a value, impose a value or abstain from it.

The Coordinator has access to the Controller information through the Conflict Analysis Interface. He acesses the Design Team explanations through the Explanation Interfaces. Design Team rationale and Conflict History provides the grounds for understanding conflicts, consequently solving them. Since MultiADD works in a ADD style, a suggestion to conflict resolution is provided, although the underlying group behavior asks for human creative action.

An example: ADDProc

In this section we present our first implementation of the MultiADD model, ADDProc, dealing with the domain of process plant design for offshore oil platforms. ADDProc is an intelligent design tool (based on the ADD architecture) for oil production processes, as described below.

Process plant design for offshore oil platforms

Design of oil processing plants involves several specialists in different domains. Each designer is in charge of one or more disciplines for the same project. A typical process plant contains 19 different subsystems including, Oil Receiving, Oil Heating, Oil Separation, Oil Treatment, Oil Transference, Gas Compression, Gas Treatment, and Water Heating. Designers of each discipline select systems and to achieve the final goal: the production of gas and oil. Each project is coordinated by one more senior engineer, who is responsible for handling conflicts that may emerge from the individual subsystem's design and guaranteeing the deadlines.

The oil comes from the reservoir as a mixture of oil, gas and water which needs to be separated and treated before being exported. Each of the subsystems manipulates the Flow, Pressure, Temperature and Composition of this gas, water and oil mixture flowing through the platform.

As the oil flows through various subsystems, conflicts will certainly appear. For instance, the oil comes from the reservoir in a low temperature, which is generally 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, in turn, must select 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. However, minimum platform area is an overall requirement, and this designer will try to reduce the equipment's length by increasing the temperature.

ADDProc

We implemented a centralized version of MultiADD, where each of the subsystems is an independent Design Team with its specific domain knowledge and designer. There is one agent which is hierarchically superior to the others, being responsible for the overall process plant design and having the authority to solve any conflict (the coordinator). One controller module takes care of the communication among the agents, verifying when conflicts happen and when they are resolved, and informing the agents as necessary.

Expanding on the previous section's example, suppose that the controller has detected a conflict between the Heating and Separation subsystems, over the oil temperature parameter.

Upon receival of the above requests for update, the controller verifies an inconsistency between the proposed values for the oil temperature parameter. Consulting its knowledge bases, it verifies that this parameter has been in conflict for too long (heuristically inferred by the number of cycles). In addition, this has a high impact on 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 will first send the conflict information to the Heating system, trying to persuade the designer to change the proposed value. If the conflict persists for much longer, the controller contacts both systems trying to persuade any of them. Finally, if they still don't reach an agreement, the controller contacts the coordinator showing the conflict. It suggests the Coordinator's support to the Separation designer's solution.

Conclusions

Traditional approaches to MAS have considered only the computational agent. We have emphasized the human designer, taking a new approach and creating a "mixed" system, where human and machine work together as one. Thus, designers develop their work assisted by a computational agent, in a cooperative design environment. Therefore, design models are guaranteed consistency within and among design pieces.

From the implemented system, we have already noted a potential decrease on the number of design meetings and design meeting duration. The ADD agents favor an increase on the number of alternative designs tried and more thorough evaluation of these alternatives. However, we expect the greatest impact to be on the work cycle. Time spent scheduling meetings or gathering information can be drastically reduced from days to minutes.

References

[1] Fischer, Gerhard; Grudin, Jonathan & Lemke, Andreas - Supporting Indirect Collaborative Design With Integrated Knowledge-Based Design Environments - *Human Computer Interaction, 1992, Volume 7, pp281-314*

[2] Garcia, Ana Cristina B. & Howard, H.Craig - Active Design Documents: From Information Archives to Design Model Construction or Making Documents Useful - AIENG, 93

[3] Jin, Yan; Levitt, Raymond; Christiansen, Tore & Kunz, John - The Virtual Design Team: Modelling Organizational Behavior of Concurrent Design Teams - *AIEDAM, 9, 145-158, 1995*

[4] Oliveira, Eugénio & Qiegang, Long - Towards a Generic Monitor for Cooperation - *Workshop on Blackboard Systems of the AAAI, Anaheim, California, 1991*

[5] Olson, Gary M; Olson, Judith S.; Carter, Mark R. & Storrosten, Marianne - Small Group Design Meetings: An Analysis of Collaboration - *Human-Computer Interaction, Volume 7, 1992*

[6] Peña-Mora, Feniosky, Sriram, Ram D. & Logcher, Robert - Conflict Mitigation System for Collaborative Engineering - *AIEDAM, 9, 101-104, 1995*

[7] Peña-Mora, Feniosky & Hussein, Karim - Change Negotiation Meetings in a Distributed Collaborative Engineering Environment - *Information Technology in Civil and Structural Engineering Design, Scotland, August 1996*

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