Semiotic Framework for Multi-user Interfaces
Raquel Oliveira Prates, Clarisse Sieckenius de Souza, Ana Cristina Bicharra Garcia

Semiotic approaches to user interface designs have shown that system interfaces are messages sent from the designers to the users. It is through the system's interface that the designers tell the users the problems the system is able to solve and how the users should interact with it to solve these problems. In a multi-users environment this message is more complex, since the designers must also tell the users how to interact in the group. To create such message is not an easy task and designers should be provided with a good designing tool. In this article we present a semiotic framework that is the first step in the direction of the construction of a multi-users interface design environment. The framework has three dimensions: action, communication and observation and provides support for the analysis of multi-users system's interface and understanding of interactions in the group. It can also be a helpful guide to designers of multi-users interfaces.

Introduction
People have always worked in groups and now, as networks link computers worldwide, the software industry is faced with the challenge of supporting extensive collaborative work [15]. Among the various approaches we can take to understand and achieve the design of quality software tools, there is the semiotic approach.

In the semiotic approach [8,7,6] a system’s interface is a one-shot message sent from the designer to the user. Since this “message” is also capable of exchanging messages with the user, it can be conceived of a metacommunication artifact [6]. The message being conveyed by the system’s interface is the answer to two fundamental questions: (1) “What problem(s) is the system able to solve?” and (2) “How can users interact with the system to implement these solutions?”. In a multi-users environment this message is more complex, since it must also make users understand how they are supposed to interact with other group members. In this case the system’s interface is not only a metacommunication artifact, but also a communication embedding artifact.

Creating these complex metacommunication and communication embedding artifacts is not an easy task, and multi-users interface designers should be provided with a good designing environment. The ideal designing environment contains descriptive and diagnostic tools that can guide designers without constraining their creativity, and its fundamental components are a group of widgets and interactive structures (structures composed by widgets with an interactive purpose). To support the designers’ definition of the components to be used in a specific interface there are rules of formation of complex interactive structures, rules of cohesion and consistency between selected interactive sub-structures, rules of correspondence between interactive structures and the message it conveys in group action domain, and finally a mechanism for users to extend the interface language. All of these are “per se” a research agenda in computer human interaction.

In this article we present a framework that is a first step in the direction of the construction of the multi-users interface design environment. The framework has dimensions that distinguish forming components of interactive structures to be used in the design of multi-user interfaces. Moreover, this semiotically-based framework provides support for the analysis of multi-users system’s interface and understanding of interactions among group members. This also makes it a helpful guide to designers of multi-users interfaces.
In Section 2 we review semiotic engineering concepts and a semiotic approach to system design of multi-users interfaces. In Section 3 we present our framework and its basic dimensions: communication, observation and action. In Section 4 we explain how members’ properties and behavior, as well as group properties can be described by the framework and how they combine to define group interactions. In Section 6 we analyze two multi-user systems’ interface according to our framework: ADDProc and TeamRooms. At last in Section 7 we present our concluding remarks, showing that different issues analyzed separately by other authors can be explained using the framework presented.

**Semiotic Engineering**

Semiotic Engineering is based on communication among people. When a person wants to communicate with another person he/she expresses his/her ideas in a symbolic system both transmitter and receiver are aware of and sends it through a channel to the receiver. The receiver gets the message and decodes it [22,16]. A message is formed by one or more signs. Pierce [25] defined a sign as being something that stands for something else to someone, based on some codifying rule. For instance, the word “dog” or a drawing of a dog are signs of the animal dog. In order to understand what the sender intended to transmit, the receiver (or interpreter) generates a mental image of the message, called interpretant. This interpretant is also a representation of the object referred to by the message; in other words, it is also a sign and can generate other interpretants. This process is called unlimited semiosis [11]. To illustrate this process imagine that a person is shown a picture of a dog and thinks of the word “dog”. The word “dog” is an interpretant of the picture of a dog and is also a sign of the animal dog. The word “dog” can generate a mental image of a barking dog, that makes the person think of a dog-collar and so on. Figure 1 shows a diagram of the communication process between two people.

![Figure 1: Communication Process.](image)

In a semiotic engineering framework [8,7,6], an interface system is a one-way message sent from the designer to the user, conveyed through a computer media. The system’s interface is a message that can also exchange messages with the user; hence it is a metacommunication artifact. This metacommunication artifact carries to the user the answer to two questions:

1. What are the problems the system is able to solve.
2. How can the user interact with the system to solve these problems.

When the application, instead of being a single-user application, is a multi-user application, there are other questions to be answered by the designer.

3. What is the group’s hierarchical structure?
4. What is the distribution of tasks among members of the group?
5. How should each member interact with other members of the group?
6. What are the other effects of local and global actions of members?
7. What is the communication language among members?
8. What are the domain dependent contents?

The answers to these questions define how the users are supposed to interact as a group and must also be conveyed by the interface. Hence, the system’s interface can be perceived as a communication generation artifact.

For all this communication process to take place [8] the designer must create an abstracted conceptual model of the application’s model. He/She then codes this information into a message, which is the system’s interface and passes it to the user(s). Each user, if more than one, will create his/her own interpretant, that is ultimately his/her own conceptual usability model of the application. Each user’s assigned meaning for the interface message will differ from the designer’s intended meaning. However, all meanings should be consistent with each other. Figure 2 illustrates this communication between designer and users of a multi-user system.

In order to explain to the users how to interact as a group, the designer must define the group protocols, that is, the group’s ways of interacting. Protocols can be classified as social or technological protocols [12]. The protocols defined by the designer and built into software are the technological protocols. An example of a technological protocol would be a conferencing system that can only process one user’s input request at a time, consequently imposing turn-taking onto group members. Instead of building protocols into the system, the designer can define interacting modes in which the protocol is left to the control of the users, which are called social protocols. Social protocols can be observed when people interact through talk systems. For instance, a person knows the other one has finished the conversation when this other person types -oo-, meaning over. The person agrees that the conversation is over by typing -oo-, meaning over and out. Only then do they leave the system. This protocol is completely defined by users, as far as the system is concerned anyone of the users can end a conversation at any time and without any warning by exiting the system. Mixed protocols can also be designed by defining part of it in the system and leaving another part to be defined by users.

*1 Domain dependent issues will not be treated in this article.*
The choice of which approach is best depends on the domain. The social protocols allow users to identify acting opportunities and to take the initiative. On the other hand, systems take the initiative in technological protocols, ensuring that a determined process will be followed.

A Semiotically-based Framework

The framework presented in this section was created using a semiotic approach. Its main goal is to help users to understand the messages about group interaction being passed by the designer through the interface. Our framework is also aimed at guiding multi-users interface designers to get their message across. To describe a group's interaction, each one of the group's members must be described using this framework. The combination of the members' description depicts the group's interaction.

The first step taken in the construction of this framework was the identification of its dimensions. This identification was based on theory from different areas such as the study of groups [5,22,24,30], groupware [1,12,15,25,27], software engineering [2,14,29], coordination theory [4,19,20,21], user interfaces [6,7,8,17,23,33] and Semiotics and Pragmatics [11,25,28]. We next present each one of these dimensions.

Dimensions

The framework is composed by three basic dimensions: action, communication and observation. The possible values presented for each one of the dimensions are qualitative, that is, they just state whether that value is present or not. These values only become quantitative or ordinal when embedded into a specific domain. Each one of the dimensions can be represented as a true/false axis, as shown in Figure 3. The true side of the axis represent the factually evident presuppositions, whereas the false side represent the presuppositions that are not factually evident.

**Figure 2: Communication process through interface message**

**Figure 3: Framework dimensions**

**Action**

The action dimension represents the work a group member can do. A person's work can be viewed in terms of goals, tasks, plans and resources. Goals refer to the high level objectives each member intends to achieve. Tasks represent the group of activities that must be completed in order to achieve the goals. A plan is the sequence in which tasks must be performed to achieve goals. Finally, resources are the supplies (material, time, money, knowledge, information, etc.) each member has available to work. The issues involved in the group members' relationship with their own actions are the same ones of single-user systems [18], and is not in the scope of this article.

The false values of the work axis (dotted in Figure 3), represent the erroneous presupposition one makes about a member's work. For instance, one member thinks that the other member has to complete task A to achieve his/her goal, however this other member only has to perform tasks B and C. In intelligent systems the knowledge acquisition module (if there is one) allows the users to change their own possible actions to some extent. In that case, some of the resources, plans, tasks or goals that are not part of users' work, and are therefore potentially wrong presuppositions, can become part of their work. This would correspond to passing a resource, plan, task or goal from the false side of the axis to the true side. Notice that the actions that can be altered by the user are predefined by the system's designer and are limited by the group's structure and the users' authority.

**Communication**

This dimension represents the member's ability to communicate with another member of the group. Different aspects of a member's communication have to be specified in order to define his/her ability to communicate. First of all, the communicative activities the member is able to perform: speaking, listening or both.

Members' speaking ability is characterized by the situations when the members can speak and by the kind of discourse
they can produce. Members may be allowed to talk at any-
time they feel necessary or they may be restricted to specific
situations and authorizations. The restriction of speaking in
specific situations can be defined in two different ways: the
user may either be able to speak or may be forced to do so in
those situations. Notice that these possibilities are not mutually
exclusive. The user may be free to talk at anytime and
still be forced to speak in some specific situations.

To define the different types of discourse a group member
may be allowed to produce, we use Searle’s taxonomy of
speech or illocutionary acts [28,33]. He describes five catego-
ries of illocutionary acts:

- assertive: commit the speaker (in varying degrees) to
something’s being the case — to the truth of the ex-
pressed proposition.
- directives: attempt to get the hearer to do something.
  - question: can direct hearer to make an assertive
    speech act in response.
  - command: attempt to get hearer to carry out some act.
- commissives: commit the speaker (again in varying de-
grees) to some future course of action (i.e. promise).
- expressiveness: express a psychological state about a
  state of affairs (includes praising).
- declarations: bring about the correspondence between the
  propositional content of the speech act and reality.

The listening ability does not have to be defined any further.
We consider that if members can listen, they can do so
whenever any other member speaks to them.

The false part of the communication axis represent the wrong
presuppositions members make about their own abilities to
communicate. Some systems enable users to extend their
communication ability through the use of end-user program-
mapping languages [9].

Observation
The true side of the observation axis represents the visibility a
member has. Members’ visibility is defined in terms of what
they are able to see of the other agents (human or auto-
matic), another member’s data or working processes (activi-
ties). Working processes can be decision-making processes,
problem-solving processes or any other kind of processes.
The users’ working processes are determined by the system’s
domain. The observation axis’ false side defines objects that
the member does not see, but knows about. These objects
can be defined in the same way as visible objects.

We would like to point out the difference between seeing
something and knowing about something. A person who
knows something has not necessarily seen that thing. There
are other ways of people learning something; they could have
learned from a previous experience, someone could have told
them about that thing and in the case of users, they could
have learned about it outside the system. Analogously, when
people see something, they will only know it if they under-
stand what it means. In a multi-users system if members

Framework Summary
We next present a summary of the framework described. The
order in which the possible values for each one of the dimen-
sions are listed is meaningless, since the values are just
qualitative

Action:
- resources
- plans
- tasks
- goals

Communication:
- speaker/listener
- when:
  - anytime
  - specific situations (allowed x forced)
- illocutionary acts:
  - assertive
  - directives:
    - question
    - command
  - commissives
  - expressiveness
  - declarations

Observation:
- agents (human or automatic)
- data
- processes

Group’s Interactions
Now that we have described each one of our framework’s di-
ensions, we will show how they can be combined to de-
scribe other members’ properties, such as their position in
the group’s hierarchical structure, visibility degree, work view
and means of participation, and also members’ interactive
behavior that cover conflict and negotiation, coordination and
members’ attitudes. However, a group’s member actual be-

Hierarchical Structure
Hierarchical structure is a group property. Yet, we will de-
scribe it before discussing member’s properties and interac-

2 The framework dimensions: communication, visibility
and action are members’ properties.

3 The only way to guarantee that a user will interact in a
determined way is by defining a technological protocol for
it(see Semiotic Engineering section)
tive behavior. The reason for doing so is that the group's hierarchical structure can be defined by comparing a single property among members: communication. All that has to be done is look at which members can perform a directive command act to which other members, that is, who can give orders to whom. Figure 4 shows an example of the definition of the hierarchy structure of a group by looking at its members' communication. Member C can give orders to members B and D and member B can give orders to member A, therefore members B and D are subordinate to C and member A is subordinate to B. Members B and D can talk to each other, but none of them can give orders to the other, so one is not subordinate to the other. Notice that member A cannot speak to anyone, so he/she is just a listener.

By looking at the hierarchical structure, we can tell if a coordinator's authority is legitimate or not. In the case of the structure depicted in Figure 4, all coordinators are legitimate, since no coordinator can overrule any other coordinator. If member C could also give orders to member A however, the coordinator could overrule coordinator B, delegitimizing him/her (Figure 5).

A group that does not have any hierarchical structure is called anarchic. That happens when no members can use directive command acts with any other members, that is no one can give orders. If everyone can give orders to everyone else, no one really has authority over anyone else, and it is also an anarchy.

Visibility Degree
Comparing the visibility each member has of all the other members of the group and then comparing the result to the group's hierarchical structure we will define each member's visibility degree. In Figure 6(a) we show again the hierarchical structure defined in Figure 3 and in Figure 6(b) we show a graph depicting each members' visibility degree. In Figure 6(c) we see that all members can see their subordinates, coordinators and siblings (members that have same coordinator), but members can only see their direct subordinates' working processes. If coordinator C could see working processes of member A, although he/she could not give direct orders to member A, he/she could give member B orders of what orders to give A. This would also be delegitimation of coordinator B.

Figure 4: Hierarchy structure of a group defined by the communication dimension. Member C gives orders to members B and D, and B gives orders to A.

Figure 5: Coordinator C can delegitimate coordinator B.

Figure 6: (a) Hierarchical structure. (b) Visibility graph. (c) Visibility degree.
Work View
What each member can see of another member’s work defines the work view that member has of the other one. Table 1 shows how a work view of a member can vary.

<table>
<thead>
<tr>
<th>Visibility</th>
<th>Resource</th>
<th>Tasks</th>
<th>Plans</th>
<th>Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent</td>
<td>Which agents own resource</td>
<td>Which agents own task</td>
<td>Which agents own plan</td>
<td>Which agents own goal</td>
</tr>
<tr>
<td>Data</td>
<td>Assigned values to resource</td>
<td>Values used in task or result values</td>
<td>Percentage of plan achieved</td>
<td>Final values achieved</td>
</tr>
<tr>
<td>Process</td>
<td>Process how to get resource</td>
<td>How each task is performed</td>
<td>Steps of plan involved in achieving goal</td>
<td>Process involved in achieving goal</td>
</tr>
</tbody>
</table>

Table 1 : Work View: Work x Visibility
Each cell of the table represents what it means to “see that much” (each visibility level) of “that much work” (work levels).

Means of Participation
The objects of a member’s work another member can talk about are the utterable objects. The amount of participation of a member on another member’s work grows with the number of possible utterable objects and the discourse that can be made about them. If a member is just a listener and therefore cannot say anything about the others work, he/she will not be able to participate on the other’s work.

A member that can speak at anytime has the means to be more participating on another member’s work, than a member that can only speak in specific situations. The more illocutionary acts a member can use, the more ideas is he/she able to express. Consequently, the more he/she can participate in that member’s work. Notice that it is not possible to define which one of the illocutionary acts is more used in participating in other member’s work, but for the directive command act. Therefore, it is not possible to create a participating scale of speech acts.

Conflicts and Negotiation
Group members usually share resources, tasks, plans or goals. Whenever there are interdependencies among group members there are also potential conflicts [20,21,31]. The members’ work view may indicate the conflicts with other members. The more information users can see about a conflict, the bigger are the chances of their identifying the causes of the conflict. To illustrate, suppose two members identify a conflict on the value assigned to a shared resource. If besides seeing the conflicting values of the resources, they can also see each other’s performed working processes to achieve that value, they may be able to identify the conflict’s cause as being a step taken to achieve the value of the resource. This way they would shift the conflict from the value of the resource to a process step taken to achieve that value. In a situation in which conflicting members are not aware of their conflict, the conflict is shifted to the next hierarchical level in which one or more coordinators are aware of it.

Conflicts imply that conflict mitigation is necessary. Mitigation can be achieved either by negotiation or by imposition of a solution by a coordinator [3,4,31]. To be able to negotiate a member must be able to communicate. As the communication power of members increases, so does their negotiation power. Negotiation is also facilitated when users have a wider visibility of the conflicting object, and therefore may be able to identify its cause.

It is not necessary that conflicts be negotiated by conflicting parties. In the case in which conflicts are between subordinates of different coordinators, it may be left to these coordinators to negotiate. The same holds for imposed solutions, that is, a solution can be imposed by a coordinator many levels above (in the hierarchical tree) the level where the conflict actually happened.

Figure 7 shows how conflict and negotiation can be represented in our framework. The shaded area represents the area of possible identification of a conflict between two members, that is, the work view area in which the cause of a conflict may be identified, shifting the conflict. The negotiation space is how much members can communicate about the conflict’s possible identification area. Notice that the negotiation space does not begin in the communication axis’ origin. That is because negotiation requires that all the involved members be able to listen and to perform some kind of speaking act.
It is worth pointing out that conflict identification and negotiation could occur in the observation axis’ false side (reported speech side). In this case, conflict and negotiation are much more complex processes. This complexity is due to the great number of intermediaries that could be involved. Besides, if we were to analyze them, the different communication abilities of each one of these intermediaries would have to be considered.

**Coordination**

As we showed in the definition of group’s hierarchical structure, for a group’s member to be a coordinator he/she must be able to perform directive command acts. Although being able to give orders defines a member as a coordinator, it is not enough to define the coordinating methods he/she can adopt. The coordination methods depend on the work view a coordinator has of his/her subordinates and the other illocutionary acts the coordinator can perform. The coordination methods establish the kind of orders a coordinator can give; his/her opportunities to give them (whenever desired or specific situations) are not relevant to determine them.

We identified four main coordination methods:

- Determination of subordinates’ work: The coordinator determines what are the subordinates’ resources, tasks, plans and goals.
- Determination of input data: The coordinator determines values to be used by subordinates when performing work.
- Determination of process of work object: The coordinator determines processes to be used by subordinates when performing well.
- Suggestion: The coordinator makes suggestions related to subordinates’ work to them.

**Members’ Attitudes**

Durfee et al. [10] describe members of a group as being cooperative, competitive or co-existent. Cooperative members usually share goals [10,21,22], which means that when one member gets closer to his/her goals, he/she is also taking the other members closer to their goals. To cooperate members must recognize its for their mutual benefit [10]. On the other hand, if members have opposed goals they will be competitive [22]. In this case, what gets one member closer to his/her goals, gets the others further from theirs. Even if members do not have different goals, but have differing views on how to achieve them, they may be competitive [10]. If the members of a group are just co-existent members, the goals of one does not interfere with the goals of the others.

Visibility and communication can foster attitudes of cooperative and competitive members. The more a member can see and talk about (participate in) a cooperative’s member work, the more he/she will be cooperative. However, a competitive member will use the information on another member’s work to take advantage over him/her.

**Describing Group’s Interaction**

To describe the group’s interaction it is necessary to describe each group’s member interacting ways. To do so, the member’s work view, means of participation, means of conflict identification and mitigation and his/her attitudes have to be defined in relation to all other members of the group (or subgroup of members that interact the same way). The member’s position in the hierarchical structure and visibility scope only have to be defined once for each member, since they are already defined in relation to other members. If the member is a coordinator, his/her coordination methods must also be defined for each subordinate or subgroup of subordinates. Once each member has been described in terms of his/her interacting ways with different members or subgroups, he/she can be fully described by the combination of these partial descriptions.

**Examples**

In this section we analyze two different implemented multi-users systems according to the framework presented. In each example’s specific domain we instance the quantitative values of our framework’s dimensions. The first example is ADDProc, a multi-users system for the design of oil processing plants [13]. The second example is TeamRooms, a multi-users system that create virtual rooms for groups to work together [26].

**ADDProc**

**Description**

ADDProc is a multi-users systems for the design of oil processing plants. The design process is divided into nineteen different subsystems. Each subsystem is formed by a responsible engineer and an intelligent computational agent counterpart. Engineers and computational agents work together setting parameters’ values and selecting a basic platform design in order to define the subsystem’s equipment list.

Subsystems share parameters among them. Each one has a copy of the shared parameter to which a value can be assigned. Engineers set values to the subsystem’s parameters’ copies and then pass these values on to the system. The system then checks whether the received values of the parameters’ copies are conflicting; if so, the involved users are informed.

Engineers have access to other group members’ work through the explanation component of ADDProc [32]. In it, they can see what are the parameters being considered, their values, the process performed to achieve those values and other users’ actions until that moment. Moreover they can ask questions about the parameters such as “What is its value”, “Why this value” and “Why not that other value”. Users have great visibility of other members’ work, but they cannot talk to each other about what they see, nor anything else.

In addition to the specialist engineers, there is also a more senior engineer, called the coordinator, who is responsible for defining the existent subsystems, the leading engineer for each subsystem and the global variables of the platform design process. This person is also responsible for resolving conflicts and making sure the project will successfully be fin
ished in time. In order to solve conflicts the coordinator, unlike the other members, has access to a conflict identification mode and can speak.

Whenever there is a conflict, the coordinator is allowed to speak to members who have the conflicting parameters. The coordinator’s speech is not predetermined by the system, allowing the user to perform any illocutionary act desired.

In the conflict identification mode, the coordinator has access to all the current conflicting parameters, their values and which subsystem assigned each one of them. The coordinator is also supplied with information on each user’s conflicting value, such as how a change of a parameter’s would affect the subsystem’s project and how much the responsible engineer is committed to that value. Based on this information, the senior engineer can coordinate the project’s conflicts and make sure they are solved.

Framework analysis
First we will describe the dimensions’ values qualitatively for ADDProc[13]. Then, by analyzing ADDProc, we will be able to order them quantitatively.

ADDProc is designed based on predefined group members’ goals and plans. The group’s goal is to design the best oil processing platform plant. To achieve this overall goal, each user’s plan is to set parameters’ values, a basic platform design and the equipment list, in order to try to design the local optimum, that is the best subsystem plant. This way, the actions each member can take in the system involve resources and tasks. Resources are each subsystem’s parameters and their range of possible values. Tasks to be performed by the users include setting values for the parameters in partnership with the intelligent agent, evaluating decisions based on criteria, restriction and possible alternatives and trying to solve conflicts as they are generated.

As for visibility, users can see who are the other group members (agents), their rationale to make decisions about parameters’ values (processes) and the parameters values (data).

As described before, the coordinator is the only member allowed to speak and only in specific situations, whereas the others can just listen. When speaking, the coordinator can use any kind of illocutionary acts.

Next we show a summary of what are the framework dimensions’ values in ADDProc’s domain. Notice that so far we are still considering their qualitative aspect.

ADDProc’s Framework

- **Action**
  - Resources: parameters
  - Tasks: set parameters values solve conflicts
- **Observation**
  - Agents: engineers and intelligent computational agents
  - Processes: rationale to define parameters’ values
  - Data: parameter values
- **Communication**
  - Listen/speak
  - Specific situations
  - All illocutionary acts

In order to define a quantitative order to the framework dimensions we must analyze ADDProc’s interface and find out what are the system’s focus. Whenever there are conflicting parameter values among subsystems, the responsible engineer is informed by the system. The system’s message informs the user which parameter is in conflict, which are the conflicting subsystems and each one’s value. If the engineer wants to understand how each one of the other engineers got to their values, he/she must access the explanation mode. We can notice that parameters and their values are central to ADDProc’s design process. Therefore, we can quantitatively arrange the framework’s dimensions as shown in Figure 8.

![Figure 8: ADDProc’s framework with values quantitatively ordered](image)

ADDProc has two different member kinds: the coordinator and the other engineers, that have already been described. Next we present the framework description of each one of these members.

- **Definition of Specialists**
  - Work:
    - Resources: parameters’ values
    - Tasks:
      - assignment of values to parameters
      - definition of best basic platform design
      - definition of equipment list
  - Plan: sequence of decisions to achieve best local design
  - Goal: best global design
  - Observation:
• Agents: other engineers and intelligent system agents
• Processes: calculation steps of parameters’ values and decision-making
• Data: parameters’ values
• Communication:
  • Speaker/listener: can only listen
  • When: —
  • Speech acts: —

• Definition of Coordinator
  • Work:
    • Resources: parameters’ values
    • Tasks:
      • definition of context:
      • assignment of values to global parameters
      • definition of best global base platform
      • definition of global equipment list
      • definition of existent subsystems
      • definition of responsible engineers
      • resolution of conflicts
  • Plan: execute tasks to achieve best global design by deadline
  • Goal: best global design

• Observation:
  • Agents: engineers and intelligent system agent
  • Processes:
    • calculation steps of parameters’ values and decision-making
    • users’ attempts to solve conflicts
  • Data:
    • conflicting values of shared parameters of all members
    • indicators of engineers work related to each conflicting parameter

• Communication:
  • Speaker/listener: can talk about conflicts
  • When: whenever there is a conflict going on
  • Speech acts: all kinds

Since the coordinator is the only member of the group able to speak and give orders to others, we can identify two levels in ADDProc’s hierarchical structure as shown in Figure 9. Notice that in ADDProc’s case the coordinator’s authority can never be delegitimated, since he/she is the only one in the group who is able to give orders.

The specialists can see each other and each other’s parameters and processes, and so can the coordinator. The subordinate engineers can only see part of the coordinator’s work. They can see global parameters’ values, the coordinator’s base platform and equipment list, but they cannot see the coordinator’s decisions and orders about conflict resolution or the information the coordinator acquired in the conflict identification mode. In Figure 10 we can see ADDProc’s hierarchical structure and visibility degree.

5 Conflicting values of shared parameters are shown to the users by the system, whereas other parameters’ values can be seen when requested in the explanation mode.

The members work view is shown in Table 2. Notice that although the members plan is predefined, it has some variables that may differ from project to project. Hence, there are some information on each member’s plan which are available to the other members.

<table>
<thead>
<tr>
<th>Visibility</th>
<th>Parameters</th>
<th>Tasks</th>
<th>Plan: design of subsystem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent</td>
<td>Which agent owns parameters</td>
<td>Which agent owns task</td>
<td>Which agents owns subsystem</td>
</tr>
<tr>
<td>Data</td>
<td>Assigned values to parameters</td>
<td>Value used in task or result values</td>
<td>Final base platforms and equipment list</td>
</tr>
<tr>
<td>Process</td>
<td>Process to get parameters value</td>
<td>How each task is performed</td>
<td>Design steps</td>
</tr>
</tbody>
</table>

Table 2: Work view of ADDProc members

As we can see in the work view, the user can see other users’ parameters’ values. Whenever a parameter assumes conflicting values, all subsystems to which that parameter is relevant are informed by the system. In a subsystem’s explanation mode all relevant parameters and their values can be seen, as can the steps taken to get to those values. This way engineers may identify conflict cause as being one of these steps that led to a conflicting value.

Figure 9: Hierarchical structure of ADDProc

Figure 10: Visibility scope in ADDProc

Agents, data and processes

Global data and processes
Although the engineers can see the conflicts and identify their cause, they cannot negotiate with other engineers because they are not able to speak. What they can do to try to mitigate conflicts is understand the other engineers’ reasons and see if it is possible to change his/her own value. What leads the users to this cooperative behavior are the interdependencies among them created by the shared parameters, the great visibility they have of each other’s work and a common overall goal.

The engineers may not be able to solve the conflicts, then it is the coordinator’s job to make sure they are solved. To do so, he/she is allowed to speak whenever there are conflicts going on, but he/she does not have to speak at any moment. Moreover, when speaking the coordinator may produce any one of the illocutionary acts. We can see that a coordination method is not predetermined in ADDProc. The coordinator can use the method he/she thinks more appropriate for the moment or the subordinate he/she is dealing with. That is, the coordination method’s choice depends on the coordinator’s personality.

TeamRooms

Description

TeamRooms [26] is an all-purpose groupware system that can be used by any kind of group, independently of their domain. The system creates a space metaphor, providing the groups with a shared environment (team rooms) and tools that support communication among group members and group’s work. These working tools include generic collaboration tools as well as special purpose tools. Rooms can be used by individuals or groups and can be long-lived. Moreover, rooms’ contents are persistent, that is, if a group member goes in the room, removes an object and leaves a note to another member, when the next member goes in, the note will be there and the object will be gone.

Any user can create a new room. Users have access to all rooms and to whom is currently working in which room. There are no differences among members, and they all have access to the communication and working tools.

Framework analysis

As we did with ADDProc we will now analyze TeamRooms according to our framework. In TeamRooms the group’s resources are rooms and users’ objects left in the rooms. As to the groups’ tasks, plans and goals we can only define them independently of the group’s work domain, since TeamRooms is independent of the groups’ domain. TeamRooms users’ goal is to work in group and their plan is to use all facilities provided by the system to enhance their work. Finally, the tasks they can perform are to communicate with other users, work individually or in group using the general and special purpose tools.

Visibility provided by TeamRooms allows users to see who are the users working in each existent room and what are the objects (messages, files, pointers to WWW pages, etc.) left in each room.

Members communication is total, that is all members can talk to any other member and listen to any one of them. They can speak at anytime and produce any kind of illocutionary acts.

The qualitative framework dimensions’ values for TeamRooms follows.

TeamRooms’ Framework

- Action:
  - Resources: rooms and work tools
  - Tasks: communication with other members, access to general and special purpose tools.
  - Plans: use all provided group-working facilities to enhance group work
  - Goals: work in group
- Observation:
  - Agents: other users currently working
  - Data and processes: those left in rooms
- Communication:
  - speaker/listener: can speak and listen to everyone
  - when: anytime
  - speech acts: all kinds

We cannot define a quantitative order to the dimensions’ values in TeamRooms. That is due to the fact that a quantitative order depends on the group’s specific domain and since TeamRooms is a general purpose system, it does not define any specific domain. Dimensions’ values quantitative orders can only be defined for TeamRooms groups’ instances.

TeamRooms does not make any difference among group members, thus its hierarchical structure is an anarchy, as represented in Figure 11. Since there are no predefined coordinators there also are no predefined coordination methods. Groups can define different hierarchical structures in the social protocol. In this case, TeamRooms allows coordinators to use any of the coordination methods, since their speech is free.

Figure 11 : Hierarchical structure of TeamRooms

All members also have the same visibility degree, being able to see other rooms and current users, as well as objects left in the rooms. Figure 12 shows TeamRooms’s hierarchical structure and members’ visibility degree.

Figure 12: Visibility Scope of TeamRooms
Being a general purpose system, TeamRooms has no definition of conflicts among group members nor negotiation. If they happen in a group’s domain it is up to that group to deal with them. Although interdependencies among group members and their goals are domain dependent, members’ great ability to communicate and access other members’ work favors cooperativeness among them.

So, compared to ADDProc, we see that TeamRooms opens many more possibilities of spontaneous group organization. On the other hand, TeamRooms provides less built-in control structures, what means that group organization is more laborious to users. These differences are due to the fact that TeamRooms is a general purpose multi-user system, whereas ADDProc is a specific purpose multi-user system.

**Concluding Remarks**

In this work we presented a semiotic framework for multi-user interfaces. The analysis of such interfaces based on this framework helps to understand the message being passed from the system designer to the users carried by these interfaces. This framework can not only be used as an analytical tool, but also as a guiding tool to multi-user interface designers.

The framework has three dimensions: action, observation and communication. The qualitative values of these dimensions were presented, whereas their quantitative order was only presented for an example, since it depends on the domain. These basic dimensions enabled us to describe other members’ and groups’ properties: work view, means of participation, visibility degree, hierarchical structure, conflict identification and negotiation, methods of coordination and members’ attitudes towards each other.

To illustrate the use of the framework, we analyzed two multi-user systems: ADDProc and TeamRooms. These two systems were intentionally picked to be very different in their objectives. ADDProc is a system to solve very specific domain problems, whereas TeamRooms is an all-purpose system for group work. In both cases the framework analysis enabled us to describe the systems and helped us understand their different interaction characteristics and purposes, what demonstrates its range.

This framework enables us to distinguish forming components of interactive structures of multi-users interfaces and is the first step in the direction of the construction of multi-users interface design environment. Our next step in that direction is to identify rules of formation of these interactive structures, and to propose a full-fledged designing environment for multi-user interfaces.

**References**


About the Authors

Raquel Oliveira Prates is a Ph.D. student in the Informatics Department at PUC-Rio, Brazil. During the year of 1997 she is a visiting researcher in the Computer Science Department at the University of Waterloo, Canada. Her research interests include Human-Computer Interaction, Computer Semiotics and Computer-Supported Cooperative Work.

Clarisse Sieckenius de Souza is an Associate Professor at the Informatics Department of Rio de Janeiro's Catholic University (PUC-Rio). She has a PhD in Linguistics (1987) and has been acting as a researcher and lecturer in the fields of Artificial Intelligence and Human-Computer Interaction for almost ten years. She is the author of over 30 publications among book chapters, journal and conference papers, and others. She has been recently doing research in Computer Semiotics and User Languages Design.

Ana Cristina Bicharra Garcia is an associate professor at the Universidade Federal Fluminense (UFF) in Brazil. She obtained her Ph.D. at Stanford University in 1992. She created the Active Design Documents Approach to assist engineering design processes. She has published at AI and Design journals. Her current interests include Intelligent CAD systems, Multiagent, Knowledge Acquisition and Spatial Reasoning.

Authors’ Addresses

Raquel Oliveira Prates
University of Waterloo
Faculty of Mathematics
Department of Computer Science
N2L 3G1 - Waterloo - ON - Canada
Tel: (519) 885-1211 ext. 5321
email: raquel@inf.puc-rio.br
rprates@se.math.uwaterloo.ca

Clarisse Sieckenius de Souza
Departamento de Informatica, PUC-Rio
Rua Marques de Sao Vicente, 225
22453-900 - Rio de Janeiro, RJ - Brasil
Tel: +55 21 529-9462, Fax: +55 21 511-5645
email: clarisse@inf.puc-rio.br

Ana Cristina Bicharra Garcia
Associate Professor
Departamento da Ciencia de Computacao
Universidade Federal Fluminense, UFF
Praca do Valonguinho s/n
Niteroi, RJ, Brazil
email:bicharra@inf.puc-rio.br