Capturing Computer-Human Interaction Design via the Protagonist Action Notation

Antonio Mendes da Silva Filho
Hans Kurt E. Liesenberg

Relatório Técnico IC–98-23

Junho de 1998

Capturing Computer-Human Interaction Design via the Protagonist Action Notation

Antonio Mendes da Silva Filho
Department of Informatics
State University of Maringá
amendes@din.uem.br

Hans Kurt E. Liesenberg
Institute of Computing
State University of Campinas
hans@dcc.unicamp.br

Abstract

With the recent increase in popularity of the Internet a new user population has emerged with little expertise to use interactive systems. Hence, developing interfaces for such systems has required the involvement of experts of several areas. A further observation is that about 50% of the time and resources are consumed just for the interface component of well-designed interactive systems. In order to contribute to the quality improvement of interface components as well as to make the development process of such components more effective, a specification technique for Computer-Human Interaction (CHI), called Protagonist Action Notation (PAN) is presented. PAN along with the Task Coordination Model (TCM) are used to represent the control aspects of the CHI Design. An example of a networked Web game called NetConnect4 illustrates the use of PAN.

1 Introduction

Computer-Human Interaction (CHI for short) is a new and multidisciplinary field of study. It involves expertise from several areas such as Psychology, Arts and Computer Science. As a new field, CHI is concerned with mediating communication between these two entities: the Human Being and the Computer.

Improving the usability level of interactive systems is one of the main objectives of CHI designers. This requirement becomes even more difficult to attain when we deal with distributed environments. Recent popularity of Web browsers, such as Nestscape, demands more from CHI design specification techniques. These networked systems have specific usability problems not found at stand-alone machines. Internet browsers hide details of the underlying networks from the user. This leads to unpredictability because without that kind of information it is difficult to determine whether retrieval commands will be successful. For example, a remote site failure will prevent requested information from being delivered to a user. Additionally, communication bottlenecks can delay the data transmission between a browser and remote servers.

The development of large interactive systems is as well a complex activity and it becomes even worse because of the need for engaging the user as an active participant during the entire process. Henceforth, we need to have a continuous evaluation during the whole process as suggested by Hartson and Hix [4]. This requirement justifies the need for incorporating an expert in human factors for the system development aiming to achieve quality interfaces.

Besides this requirement, the CHI design has to be transformed into interface software design without violating specification defined at a prior stage. In other words, we need to map the user-centered CHI design smoothly into system-centered design of the interface software. This makes it clear that we need specific representation techniques and tools to support these two major stages of an interface development process.

We consider task-centered modeling as the means of representing CHI design (sometimes called behavioral design). Consequently, a notation to specify CHI design must support the modeling of *user actions* and *interface feedback*. This paper presents the specification technique named Protagonist Action Notation (PAN) that allows the designer to represent CHI design.

Background issues related to specification techniques for CHI design are provided in Section 2. The main features of PAN are illustrated in Section 3 with the help of an example of a game called NetConnect4. Section 4 discusses issues related to the use of PAN to specify CHI design. Concluding remarks are presented in Section 5.

2 Background Issues

Life cycles of interface developments have usually the CHI design as one of the most important stages preceeded by task analysis. Tasks are representative elements for scenarios involving human beings and computers. Whenever a user needs to perform a task, a decomposition of a primary task into secondary tasks as well as a likely execution sequence of such secondary tasks are defined which try to minimize the effort of carrying out the entire task. To do so we need a task-based approach so as to meet requirements of decomposing a system into tasks as well as to identify the relationships among them.

An interface design can be task driven. Initially, we obtain abstract user tasks. The identification of such tasks is based on the application domain. As the development process goes on, we need also to find out the relationships among tasks and we move gradually from abstract tasks to concrete ones. Such an approach of development is known as model-based paradigm as shown by Puerta [18].

2.1 Model-based Interface Development

Model-based design is a technique used to create interface designs represented at a high abstraction level. Design is captured as models described in appropriate notations which capture aspects related to computer-human interactions. This design paradigm provides a high-level abstration to represent the design. The specification of design in terms of high-level abstractions is also advocated by Holtzblatt and Beyer [7, 8].

This approach aims at improving the support for both the development life cycle and the design analysis. Moreover, an improvement in terms of usability and design traceability is achieved. It also facilitates design changes and provides support for reuse [18]. In the CHI context, we can model domain actions and components, interface behavior, user tasks, and presentation aspects. Examples of model-based approaches are given below.

1. UIDE (User Interface Development Environment) - As shown by Foley et al [2], it comprises:

   - *application model* which defines the allowed user actions and their effects on application objects;
   - *interface model* which describes user actions in terms of interface objects;
   - *pre- and pos-conditions* associated with actions;
   - *objects* that are defined in terms of their attributes.

2. ADEPT (Advanced Design Environment for Prototyping with Tasks models) - This approach has an abstract interaction model involving classes and interaction object clusters [23]. It has a concrete interaction model as well which consists of interaction objects, dialog and screen layouts. Furthermore, ADEPT makes use of design guidelines for selecting widgets and interaction styles.

2.2 Task-based Interface Development

Related to the model-based design approach we have the task-based design. This approach integrates human factors with software engineering in order to support interactive system design. Additionally, it offers support for user-centered design where design solutions are based on user task descriptions. There are some examples of environments which support and integrate interface design activities based on a task-based paradigm:

- MUSE (Method for USability Engineering) integrates human factor techniques with Jackson System Development (JSD) method [10]. MUSE uses the JSD design notation to describe task hierarchies.
- TRIDENT [21] is an integrated methodology based on tasks. It is a modeling approach similar to TKS (Task Knowledge Structures) proposed by Johnson [9]. TRIDENT allows automatic generation (by using design guidelines) of user interfaces from abstract models.
- ADEPT [23] is an interactive design process based on tasks which has a development environment to support task modeling.

2.3 Task-based Specification Techniques

Task-based Specification Techniques have concentrated on representing user actions. Related proposals are GOMS (Goals, Operators, Methods and Selection rules) by Card, Moran and Newell [1], TAG (Task Action Grammars) by Payne and Green [17] and the UAN (User Action Notation) by Hartson, Siochi and Hix [3, 5]. However, only the UAN provides support to specify system responses such as the feedback and the state of an interface.

The UAN is a user and task-oriented notation that describes both user actions and interface behavior. The first abstraction of the UAN is a user task represented through a hierarchical structure of assyncronous tasks. The sequence of secondary tasks for each task is independent of other sequences. User actions, interface feedback and interface state are low-level representations.

In order to choose a suitable specification technique for representing CHI design we should consider what likely benefits would be obtained by choosing one rather than other alternatives. Some considerations follow.

1. *CHI is task-based* - Whenever a user interacts with a machine, or more specifically, a computer, he is interested in performing one or more tasks. Therefore, we can consider task as the key element of the human-computer interaction.
2. *Need for sequencing secondary tasks* - Users who want to carry out a particular task usually need to know an order in which to carry out relevant secondary tasks even if they can and normally do work by carrying out several tasks in a concurrent and assyncrounous manner.
3. *Usability* - A task-based approach allows the designer to focus on precise specifications of tasks performed by users to assure a natural human-computer communication.
4. *Concurrency support* - Specification of concurrent responses to user actions requires appropriate constructs within the adopted notation.
5. *Expressiveness to represent temporal properties* - Unpredictable delays can occur when users attempt to access information over a network. Bottlenecks may hamper information on the way back to a browser. This problem can get worse or lessened as the load of the network changes over time. Specification techniques must allow designer to represent the existing temporal relations.

The UAN addresses most of these issues related to CHI design. Nevertheless, we may be faced with a menu-based interface scenario where user can select any of the menu titles and next choose one of the menu options. Users, however, change frequently their intentions and this has to be supported by the interface. Although the UAN describes well low-level user actions, the shift of intentions is characterized as being a high-level one and consequently the tabular form of the UAN is no longer suitable to express this requirement.

Another situation arises from the need to specify unexpected behavior. In this case and the prior one, the Protagonist Action Notation is better suited. PAN provides designers with a specification technique to represent protagonist actions of a system under development (SUD for short). Examples of protagonists are users and software components. By using PAN we are seeking protagonists that play roles in a human-computer interaction. A specification in PAN is described by a directed graph where nodes stand for protagonists and the edges connecting nodes identify the relationships between protagonists. Section 3 illustrates the use of PAN by means of a case study. Note that the whole specification is depicted at a very high abstraction level and this therefore motivates and facilitates user participation quite early in the CHI design process.

As we make progress, we have a further need to describe tasks for each protagonist. PAN provides designer(s) with a Task Coordination Model (TCM for short). This model is used to represent tasks carried out by each protagonist at a high abstraction level. TCM is a directed graph where nodes stand for tasks and edges connecting nodes represent the stimuli that trigger navegation among tasks. Migration from one task to another characterizes both existing relationships among tasks and an intention shift made by a user (or another protagonist). That is exactly what is needed in CHI design, i.e., a global picture of a system described in terms of a task set. The term global is used here to convey the high abstraction level of a task set. Examples of the use of TCM are given in Section 3.

The adoption of a new specification technique depends upon how easy and effective designers can perform their tasks and how benefits can be obtained in real world applications. Below we enumerate evidences which justify the use of PAN in CHI design.

1. Human-computer dialog is more naturally characterized by a task set. Hartson and Hix [6] recommend the use of the UAN to describe behavioral design. Despite of the richness and the applicability of the UAN to CHI design, task representation in UAN is made at a low abstraction level. However, we consider that task specifications must as well be supported at a high abstraction level in order to improve and focus on the task identification and the existing relationships among tasks. Moreover, it becomes easier and more natural to integrate the user into a design process. PAN addresses these issues.

2. Complex systems support task concurrency. PAN allows specification of concurrent tasks at a high abstraction level.

3. Abstraction is a key concept in every design activity. To design a complex system, designers have to use notations and methods that enable them to work out gradually their designs. However, the obtained specification must be consistent and unambigous to allow implementation. The design of an interface is not a different process. An interface is simply a part of an interactive system. Besides consistency and correctness, CHI design carried out at a higher abstraction level allows the designer to concentrate and reason on key components of the system, i.e., protagonists and their respective tasks. Tasks represented in TCM are grouped in order to obtain an interaction scenario (IS). Each IS is described in PAN. A designer has thus as a starting point the identification of protagonists and then their respective tasks.

4. As discussed earlier, the UAN is a rich notation to describe user actions. Nevertheless,

human-computer interaction involves at least two protagonists: a user and a system. We could have more than two if a system is decomposed into components which could be considered as protagonists because they play different roles. Therefore we need not only to describe user actions but also system-initiated actions. PAN allows the designer to represent both types of actions.

5. Along with the abstraction concept, there is a need for modularity. We make use of modularity whenever we need to deal with complex systems. PAN takes into account this requirement by providing designer with a notation at a high abstraction level where the designer first identifies system protagonists (using PAN) and then describes protagonist actions making use of TCM.

6. At last but not least, we need a specification technique for CHI design that conveys a clear idea of what is to be done. When we use an interactive system, we have indeed task support provided by the system. Tasks are the essence of CHI design and means which allow both designer and user to interact in a natural way. PAN addresses this issue providing designer with a specification tool that supports CHI design and its ease of use motivates user to actively participate.

3 NetConnect4 - A Case Study

This Section presents a case study based on the NetConnect4 system. NetConnect4 is a simple *multiplayer game*. We start providing background issues on multiplayer games and game theory to underlie the presentation of the case study and then we illustrate the use of PAN to describe interaction aspects of NetConnect4.

3.1 Multiplayer Games

Multiplayer Games that can be run on multiple machines are also known as *Network Games*, which means that the games are capable of enabling multiple players play interactively on top of a network. In the case of networked Web games, the communication between players is mediated by the Internet. In other words, in a networked Web game which involves two or more players, players are able to play the game together and interact with each other via their Web connection in a concurrent manner.

In multiplayer games, the communication design can be affected by the way the game play progresses, which is determined by the type of the particular game. Most games fall into one of two categories:

- *Turn-based games* are games in which each action in the game is based on a player's turn;
- *Event-based games* are games that are paced by input events that can occur at any time.

NetConnect4 is a turn-based game because players are allowed to make a move only at their turn.

3.2 Game Theory

Game Theory is a research area devoted to the study of decision making in conflict situations. It can be used to shed light on how people interact with each other in a multiplayer computer game scenario. Game theory might help a designer to figure out more creative approaches to the game strategy itself. Such a situation exists when two or more decision makers, or players, with differing objectives act on the same system or share the same resources. Game theory

provides an underlying process for selecting an optimum strategy which depends on the moves of the opponents who have a strategy of their own. In game theory, the following assumptions are usually made:

- each player has two or more well-specified choices or sequences of choices called moves;
- every possible combination of moves available to the players leads to a well-defined end state (win, loss, or draw) that terminates the game;
- a specified payoff for each player is associated with each end state;
- each decision maker has perfect knowledge of the game and of their opponents, i.e., he knows in full detail the rules of the game as well as payoffs for all other players;
- all decision makers are rational, i.e., each player, given two alternatives, will select one that yields the greater payoff.

Although general in scope and not originally directed at computer games, game theory touches on many of the same concerns that are raised when strategies for multiplayer computer games are being designed. Two players in a network multiplayer game often go through much of the same thought pattern as people engaged in a verbal conflict. Game theory applies equally to both scenarios. More details on game theory are given by Russel [19] and Luger [13].

3.3 Designing NetConnect4 with PAN

NetConnect4 is a networked Web game where the major goal is to establish a sequence of four pieces (similar to tic-tac-toe). The following presents the full game description and we use PAN for its CHI design.

3.3.1 System Description

The game description is as follows.

> NetConnect4 is played with a rectangular board that contains a 7x6 array positions (6 lines, 7 columns) as shown in Figure 1. The board of NetConnect4 stands in vertical position where each column corresponds to a set of slots. Thus one move consists of selecting a column and drop into it a round piece. Gravity then takes care of the rest. This is how a player can push pieces into the board. To win a game a player needs to establish a sequence of four of its own pieces horizontally, vertically or diagonally.
>
> Note that NetConnect4 is a turn-based game. Because the opponent is only allowed to make a move when it is his turn. NetConnect4 requires two players for a game. Communication between players is indirect, i.e., a third protagonist (referee) comes into play acting as a communication mediator for the players. In the envisaged system, we may have more than one pair of players playing games. Thus, for every pair of players a referee is needed who is assigned by a referee chair. So after detecting the existence of two interested players not yet paired, the referee chair assigns a referee for the game of those players. Afterwards, the referee chair waits for another couple of players to show up in order to assign another referee for those new interested players. This assignment process proceeds as long as new players show up. Figure 2 shows a scenario where all NetConnect4 protagonists take part. Note that for every pair of players a referee has been assigned by the referee chair. Other referees are assigned to mediate communication between each established pair of players which

have shown up requesting to take part in a game. In other words a player wants to take part in a game, then he must make a request to the referee chair and wait for another player to show up so that the referee chair can pair him with the player and assign a referee to mediate the game. The gray node in Figure 2 illustrates an interested player waiting for an opponent.

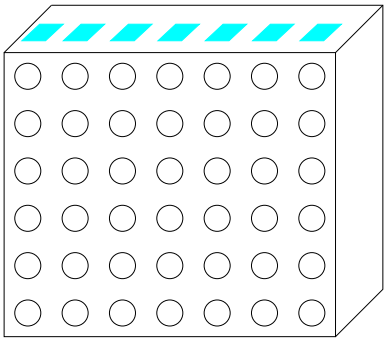

Figure 1: Board of NetConnect4.

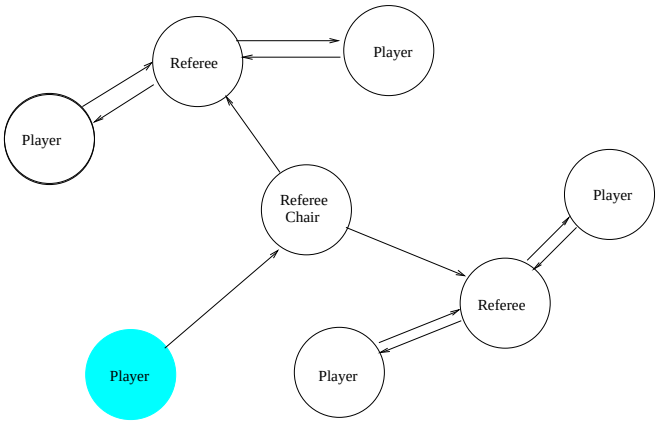

Figure 2: Scenario with NetConnect4 protagonists.

After obtaining an SUD description as above, we need to identify system protagonists as shown in Figure 2. We present as well system components and illustrate the architectural model of NetConnect4.

3.3.2 Identification of System Protagonists

Protagonists are all the components that play roles in an interaction scenario. Such protagonists are both user(s) and system (or system components). These components can be viewed as software components with major functionality roles. In order to carry out tasks, interactions between these protagonists must occur. As a result, specification must capture not only user actions but also system-generated actions. Thus using a notation that captures only user actions constitutes a hindrance to the CHI design specification needs. Consequently, a designer needs a notation that allows him to capture the whole interaction picture. In our example each player interacts with the corresponding player interface component and notifies his move to the assigned referee whenever it is his turn. The remaining protagonists are referees and the referee chair. Figure 3 shows the protagonists identified for NetConnect4. To do that, we use PAN to provide a specification at a high abstraction level.

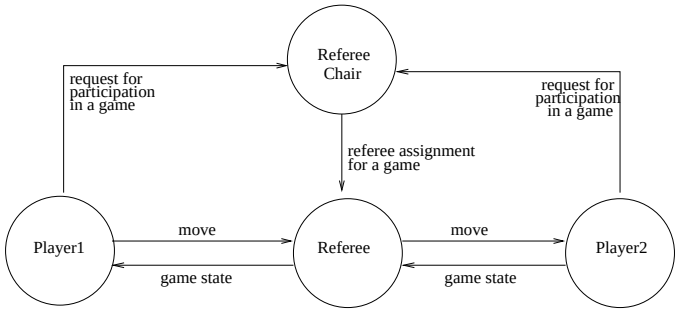

Figure 3: Scenario with NetConnect4 protagonists.

For the purpose of illustration we assume that there are only two players and that a referee has already been assigned to mediate the game. Each protagonist is capable of performing particular tasks as given below. Nevertheless, before we present task descriptions for each protagonist we identify an architectural model for the system. A way of doing that is developing interaction scenarios involving protagonists as follows.

3.3.3 Development of Interaction Scenarios

At a high abstraction level we can characterize an interaction scenario in NetConnect4 as shown in Figure 4. Additionally, we abstract from all interaction details such as *the kind of information exchanged between protagonists*. We focus on interactions between protagonists. Later on (see Section 3.3.5) we deal more specifically with protagonist actions.

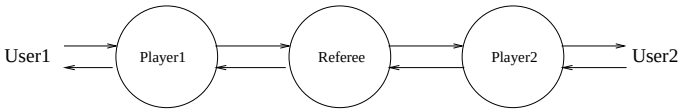

Figure 4: Interaction scenario for NetConnect4 protagonists.

3.3.4 Identication of the system architectural model

At this step, we do a system description in terms of their components or protagonists. It is worth pointing out that both users and software components can play the role of protagonists. Figure 4 gave us an idea of a possible architectural model for the system. That figure suggests that the referee acts as a communication stream manager between the players, i.e., it acts as a server while the two player interface components are its clients. So we identify the *client-server* model for this system.

3.3.5 Task Description of NetConnect4

From the identification of the protogonists and their interactions at a high abstraction level an architectural model for NetConnect4 has been identified. Such identification was derived smoothly based on the adopted interaction scenario where each protagonist (player, referee or referee chair) is capable of performing some particular tasks.

Tasks of the Referee Chair:

1. Wait for:

   (a) player(s) that want(s) to either start or draw a game;

(b) notifications of games that have finished.

2. If there exists a request of a player to play a game, then accept that request.
3. If there exist at least two players that wait to start a game then:
   (a) pair the two players,
   (b) assign a referee for the game,
   (c) enable one of the players to start the game.
4. If there exists a request for a drawing, then accept it and dismiss the referee of that game.
5. If a win has been detected and notified by a referee, inform both players that took part of the game that their game is over and dismiss the referee.

Figure 5 illustrate the TCM for the referee chair. It is worth observing that Figure 5 not only shows the tasks of the referee chair but reveals as well the relationship among them, i.e., how the task coordination takes place. The particular TCM of Figure 5 represents the tasks and their relationships of just one protagonist in PAN. In other words, it describes tasks of and how they are coordinated by the referee chair protagonist. Furthermore, the designer does not conceive this model at once. Initially, he identifies the tasks, then how tasks are related to each other, and finally what stimuli cause navegation between tasks. In particular, the termination of a task can motivate a navegation to another one or actions of other protagonists may as well cause task navegation in the TCM of the referee chair as when he receives a request from a player for drawing a game.

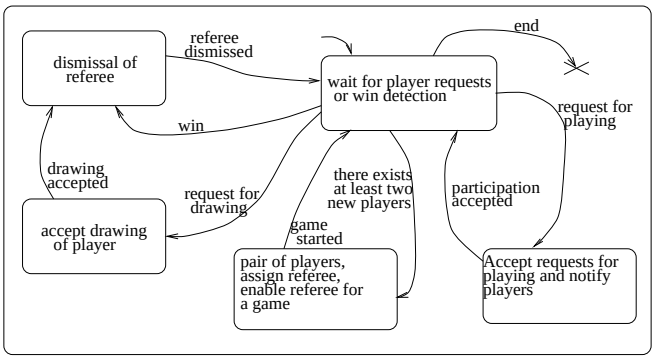

Figure 5: TCM for referee chair.

Directed edges in TCM are labelled with stimuli caused by events generated by protagonists. An event can be passed from one protagonist to another. Additionally, events can carry data between protagonists. The reader should also note that a referee chair does not interfere in a game. It is in charge of only detecting player requests to draw from a game, pair players and dimiss referees when appropriate.

Now consider the tasks of a referee. Note that each referee plays the role of a communication mediator between a couple of players helping them to play the game. In other words, a referee plays the role of intermediating moves of each player against his opponent.

Tasks of a Referee:

1. Wait for a move of the enabled player.
2. If the information about a move is received, then:

(a) update the state of the game;

(b) in the case of a win, then notify both winner and loser as well as notify the referee chair about the win and wait for dismissal;

(c) if no win has ocurred, then enable the opponent of the player who realized the most recent move.

3. If there exist a request for drawing from a player, then let its opponent be acquainted with this fact, notify the referee chair about it and wait for dismissal.

Figure 6 shows the TCM for a referee. We can see the tasks of a referee, relationships among them and the stimuli that motivate task navegation.

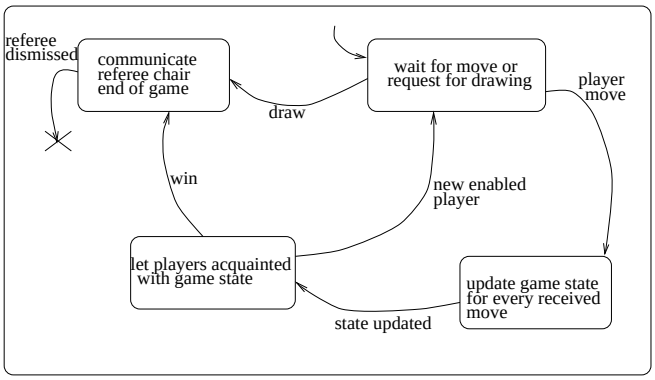

Figure 6: TCM for referee.

The other NetConnect4 protagonist is player. A player component represents a user interface module which receives actions from its associated user and maps them into moves. Note that the basic task of a player is to inform the referee chair the intent of playing a game, and then, after having been enabled, to communicate a move to the assigned referee. A player gets acquainted with the state of the game by referee and updates itself. Moreover, a player needs to reason about the game in order to carry out a new move. Figure 7 gives the TCM for a player.

Tasks of a player:

1. Make a request to the referee chair for playing a game.

2. If you want to draw, then requests it to the referee chair and wait for response.

3. While staying in the game, wait for your turn *and*:

   (a) if a win has taken place, then you:

   - can either play a new game by making an appropriate request to the referee chair; or
   - may want to draw the game by asking the referee chair for it and waiting for a response.

   (b) if it is your turn, then perform a move based on state of the game and let the referee be acquainted with it.

   (c) if it is not your turn, then receive via referee the move of your opponent, update your internal data, notify the user about the new state of the game.

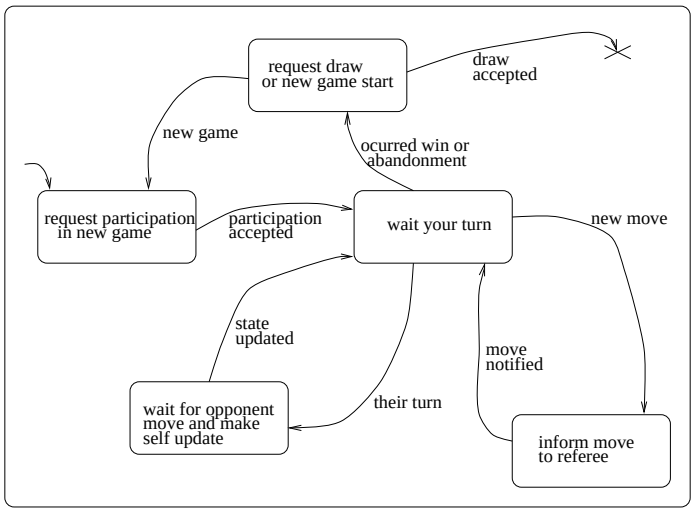

Figure 7: TCM for player.

4 CHI Design with PAN

In this Section a procedure to specify a CHI design of an interactive system is presented. Human beings often utilize computers to carry out their tasks. We can have tasks as guiding elements for a CHI design specification.

Task-based CHI design specification is proposed due to the diversity of users with different knowledge levels as well as a variety of both interaction styles and implementation technology. Moreover, interface technologies are gradually migrating from command-oriented to noncommand-oriented interfaces [14]. Hence, it becomes difficult to use a unique specification approach to meet all these requirement diversities. Furthermore human beings interact with computers easily if human-computer interaction is based on metaphors of their daily activities. Thus we use task descriptions so that our specification can reflect known user metaphors.

We also advocate the need to start task-based specification at a high abstraction level in terms only of user intentions. This makes specification easier for handling and upgrading as the refinement process proceeds until low-level specifications of tasks are produced where implementation decisions are made. From low-level task specification we can map CHI design specifications into an interface software specification as reported in [20]. Besides the use of a task-based approach, we can get users involved early enough in the design process provided that appropriate metaphors related to their daily tasks are used. Using metaphors in task-based specifications augments the integration between users and designers because they can communicate between each other using a common language.

User intentions at the task abstraction level are high-level goals which exist in the user's conceptual model about the system. User intentions to reach a goal are described at a high abstraction level without reference to any system presentation feature. For instance, in a file system *delete a file* represents a user intention in the task abstraction level. However, *drag a file icon to a destination* (e.g. trashcan icon), type a command *rm file* or utter the command *remove file* are respectively descriptions of low-level user intentions with desktop, command-based, and natural language interfaces that perform the associated high-level intention.

A key principle in any engineering discipline is abstraction. Software engineering is not an exception. Such concept together with modularity [15, 16] are used as design guidelines in our approach. Therefore, a high-level specification based on metaphors of the domain context aims to facilitate the user involvement in the design process. Below we present the steps needed for CHI design using PAN.

1. *System description* - Obtain a system description of the SUD.

2. *System protagonists identification* - These protagonists can be both user and system components. To do so, we use PAN.

3. *Development of interaction scenarios involving system protagonists* - For every protagonist, the designer must describe scenarios in order to identify its tasks.

4. *System architectural model identification* - Herein, we seek a system description in terms of its architectural model.

5. *System task representation* - In this step, we define a task set for each protagonist by using TCMs.

So far we have presented our task-based specification procedure for a CHI design. This paper aims at presenting PAN and how it can be used to represent CHI design. Nevertheless it is worth observing that CHI design is not yet complete. Another issue of CHI design is related to presentation aspects. However, it is out of scope of this paper.

5 Concluding Remarks

This paper has presented a task-based specification technique, called Protagonist Action Notation (PAN), for the difficult activity of representing the human-computer interaction design. This difficulty is due to the need for an effective participation of the user within design process as well as for a common language to enable a clean communication between user(s) and designer (team).

In order to illustrate the use of this notation, we worked out a case study so-called NetConnect4 where we can have multiple users using their Web browswers for playing the game. This scenario has the complex activity of managing the entire game. We may have a number of users in great expectation to get paired with another user in order to play. Besides taking care of that, the system also needs to assign a referee to mediate a particular game and supervise ongoing games. Despite of being a small example, it conveys several difficulties inherent to a distributed environment as discussed in Section 1. Another need in the design process is the involvement of the user to effectively participate in this process. However, by using PAN we can tackle with these difficulties as discussed in Sections 3.

PAN could also be used to specify a CHI design of a Web-based collaborating system, such as the *Digital Agora* reported by Watters et al [22]. In that project, users are typically students, faculty and institution advisers. The Digital Agora aims at providing support for active learning in social sciences. This applications and others akin can use PAN to support their CHI design representations. The reader should note that we have not addressed issues related to the interface presentation component since this paper has focused specifically on interface control component.

The work presented in this paper is part of a project which aims at both mapping a CHI design into an interface software design [20] and providing an interface software development methodology. Therein, we address the derivation of interface software design from the CHI design. Interface software design is the other major activity we have in an interactive system development. We have been using the Xchart specification language [11, 12] for the design of the interface software. Further details about Xchart language and environment can be found at: http://www.dcc.unicamp.br/proj-xchart/

References

[1] S. K. Card, T. P. Moran, and A. Newell. *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associates, Inc., 1983.

[2] J. Foley, W. Kim, S. Kovacevic, and K. Murray. UIDE - An Intelligent User Interface Design Environment. In J. Sullivan and S. Tyler, editors, *Architectures for Intelligent User Interfaces: Elements and Prototypes.* Addison-Wesley, 1991.

[3] P. D. Gray and H. R. Hartson. Temporal Aspects of Tasks in the User Action Notation. *Human-Computer Interaction*, 7:1–45, 1992.

[4] H. R. Hartson and D. Hix. Toward Empiracally Derived Methodologies and Tools for Human-Computer Interface Design. *Int. J. Man-Machine Studies*, pages 477–494, 1989.

[5] H. R. Hartson, A. C. Siochi, and D. Hix. The UAN: A User-Oriented Representation for Direct Manipulation Interface Designs. *ACM Transactions on Information Systems*, 8(3):181–203, 1990.

[6] D. Hix and H. R. Hartson. *Developing User Interfaces: Ensuring Usability Through Product and Process.* John Wiley, 1993.

[7] K. Holtzblatt and H. R. Beyer. Making Customer-Centered Design Work for Teams. *Communications of the ACM*, 36(10):92–103, Oct 1993.

[8] K. Holtzblatt and H. R. Beyer. Representing Work for the Purpose of Design. *HICSS Monograph-Havaii International Conference on System Sciences, Lucy Suchman, Editor.*, 1994.

[9] P. Johnson. *Human Computer Interaction: Psychology, Task Analysis and Software Engineering.* McGraw-Hill, 1992.

[10] K. Y. Lim, J. B. Long, and N. Silcock. Integrating Human Factors with the Jackson System Development Method: An illustrated Overview. *Ergonomics*, 35(12):1135–1161, 1992.

[11] F. Lucena. Xchart: Um Modelo de Especificação e Implementação de Diálogo. *PhD Thesis, Institute of Computing, State University of Campinas*, dec 1996.

[12] F. Lucena, M. Harada, and H. Liesenberg. Operational Semantics of Extended Statecharts (XCHART). *3rd Workshop on Logic, Language, Information and Computation*, May 1996.

[13] G. F. Luger and W. A. Stubblefield. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving.* The Benjamin-Cummings Publishing Company, Inc., 1993.

[14] J. Nielsen. Noncommand User Interfaces. *Communications of the ACM*, pages 83–99, April 1993.

[15] D. L. Parnas. On the Criteria To Be Used in Decomposing Systems into Modules. *Communications of the ACM*, 15(12):1053–1058, Dec 1972.

[16] D. L. Parnas, P. C. Clements, and D. M. Weiss. The Modular Structure of Complex Systems. *IEEE Transactions on Software Engineering*, 11(3):259–266, Mar 1985.

[17] S. J. Payne and T. R. G. Green. Task Action Grammars: A Model of the Mental Representation of the Task Languages. *Human-Computer Interaction*, 2:93–103, 1986.

[18] A. R. Puerta. A Model-based Interface Development Environment. *IEEE Software*, pages 40–47, July/August 1997.

[19] S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[20] A. M. Silva Filho and H. K. E. Liesenberg. On Deriving the Design of Interface Software from the Design of Computer-Human Interaction. *Technical Report (upcoming), Institute of Computing, State University of Campinas*, 1998.

[21] J. Vanderdonckt and F. Bodart. Encapsulating Knowledge for Intelligent Automatic Interaction Objetcts Selection. *Proc. Confererence on Human Factors in Computing Systems: InterCHI'93*, pages 424–429, 1993.

[22] C. Watters, M. Conley, and C. Alexander. The Digital Agora: Using Technology for Learning in the Social Sciences. *Communications of the ACM*, 41(1):50–57, Jan 1998.

[23] S. Wilson, P. Johnson, J. Kelly, C. Cunningham, and P. Markopoulos. Beyond Hacking: A Model based Approach to User Interface Design. *in People and Computers VIII, Proceedings of the HCI'93 Conference*, pages 217–231, 1993.