Gathering User Interface Design Requirements for Social Computing

Antonio Mendes da Silva Filho
Hans Kurt E. Liesenberg

Relatório Técnico IC–98-38

Agosto de 1998

Gathering User Interface Design Requirements for Social Computing*

Antonio Mendes da Silva Filho†
Hans Kurt E. Liesenberg
Institute of Computing
State University of Campinas
amendes|hans@dcc.unicamp.br

Abstract

Design for cooperation is a challenge. As designers we note that as we are moving towards the final years of this century, several areas have achieved significant breakthroughs. Among them, it is easy to perceive that areas of Computing and Telecommunications have had an impact of paramount importance to society as a whole. These technologies have allowed an increasing integration of research fields, people of various backgrounds and abilities as well as made the interaction of different cultures possible. As a result, we have been living in the Internet era with a very large number of Web sites which can be visited, queried and played with. That constitutes what we call social computing. Application examples are: digital libraries, health care information systems, Physics collaboratories, and Web-based entertainments like interactive Web games. Within this context, we are concerned with the user interface design requirements gathering for such systems. In that sense, we present a protagonist task-based approach for capturing the user interface design requirements.

Keywords: Social computing, user interface design, HCI.

1 INTRODUCTION

The final years of this millennium have led the whole society to a *digital maelstrom* due to significant breakthroughs achieved in the areas of Computing and Telecommunications. As time goes on, we are faced with the convergence of these technologies. Such an integration has made the reshaping of traditional computing possible where most of users are used to work on stand-alone machines. Today the traditional computing is giving way to the *social computing* which allows an ever increasing interaction of research fields, people of various backgrounds and abilities as well as of different cultures.

Social computing is an offspring of the Internet era where a very large number of Web sites can be visited, queried and played with. We have identified a number of applications such as: Digital Libraries [4], [15], Health Care Information Systems [14], Physics Collaboratories [1], and Web-based Entertainments like multiuser Web games [20]. We call it social computing since it allows different players at different sites to play together in a, e.g., networked Web game. We may have as well a Web-based collaborating system, such as the Digital Agora by Watters et al. [21] where the system aims at providing support for active learning in social sciences. Users of that system are typically students, faculty and institution advisers. These examples illustrates scenarios where performers acts cooperatively within a shared working environment.

* This research is partially supported by the Brazilian Council of Research (CAPES).

† Assistant Professor in the Department of Informatics at the State University of Maringá, Maringá, PR, Brazil, 87020-900. Email: amendes@din.uem.br.

The availability of network infrastructure together with the computer resources have made this dream become a reality. Today, it is common to make use of electronic mail, videoconference, networked Web games, digital libraries and so on. Nevertheless, the popularity of the Internet has demanded more and more from interactive systems designers in that they are concerned with the improvement of the usability level of such systems. These networked systems have specific usability problems not found earlier on 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. Within this context, we are concerned with the way designers can gather user interface design requirements. To tackle this kind of problems, we present a protagonist task-based approach in order to capture significant high level aspects of the user interface designs.

Background issues on user interface design and approaches for social computing are provided in Section 2. A protagonist task-based approach for user interface design requirements gathering is given in Section 3. Section 4 illustrates the use of our approach through an example of a multiuser Web game and concluding remarks are presented in Section 5.

2 BACKGROUND ISSUES

Computer systems are intended to aid people to perform their work. Henceforth, such systems must be built to fit the needs of their users. However, there is no way that a system can work well with a user group (e.g. a cooperative group) without a deep understanding of the users in such a group.

Computer technology can and has acted as a direct aid to people at their tasks carried out during work activities. Computers are making large and significative differences in people's life and work styles. At this time, it is important to point out that a computer does not simply replace a manual operation with a computer operation. It changes dramatically the pattern of work and of communication in a workplace. Thus, we aim at showing that computer technology alone cannot provide the answers when we deal with human activities. The tasks, the culture, the social structure, and the individual human being are all essential components of the job, and unless the computational tools fit seamlessly in the structure, the result may be failure.

As designers we need to take all these issues into account. Together with these observations we note that the transformation of telecommunications, brought about mainly by a close relationship with computers, is both driving down the cost of communication and driving up the amount of information that can be exchanged. In this context, we present a way of how designers can gather user interface design requirements for social computing which includes those systems mentioned earlier. We, however, discuss firstly two related approaches that might be used in the design process and have inspired our approach. They are: Participatory Design and Ethnography-based Design.

2.1 Participatory Design

Participatory Design (PD) is an approach where future users of computers systems participate directly with designers in the design process. The approach was pioneered in Scandinavia, is widely accepted throughout Europe and is getting attention in the USA. Clement and Besselaar [3] give a historical review of the PD approach. The ingredients in PD projects have been identified by Clement and Besselaar [3]:

- Access to relevant information.
- Independent voice in decision making.
- User-controlled resources: time, facilities, expertise.
- Appropriate development methods.
- Organizational/technical flexibility.

On top of afore-mentioned issues, Greenbaum [5] gives three reasons for the need of PD: from a *pragmatic* perspective, a *theoretical* perspective, and a *political* perspective.

In the pragmatic perspective, she states that *"... it is generally acknowledged that approximately 60 to 80% of all problems can be traced to poor or inadequate requirement specifications. Obviously, computer systems need to better suit people's working practices. Since those who do work know how it is done, we need to involve the designers of the systems with day-to-day work experience early in the project when the basic design choices are make"*. While from a political perspective, she argues that *"As systems developers we have the obligation to provide people with the opportunity to influence their own lives. We believe it is our professional responsability not only to build systems that are cost-effective but that also improve the quality of work live"*.

2.2 Ethnography-based Design

Ethnography is a technique originally developed by anthropologists where they spend long periods of time in foreign societies aiming to understand the social mechanisms. Within a context of design, the major objective is at getting a thorough understanding of work practices in order to better support the computer system design.

This is an approach that has been receiving an increasing interest. By having its starting point anchored in the social sciences and the humanities, it brings a provoking and relevant perspective into design. The focus is on the detailed analysis of current work practices, as viewed by the people who actually do the work.

According to Blomberg et al[2], the four main principles that guide the ethnographic work are:

- First hand encounters: a commitment to study the activities of people in their everyday settings.
- Holism: a belief that particular behaviors can only be understood in the everyday context in which they occur.
- Descriptive rather than prescriptive: describe how people actually behave, instead of how they ought to behave.
- Members' point-of-view: describe behavior in terms relevant and meaningful to study participants.

The afore-said principles entail that designers should not pre-define any conceptual framework. Instead, designers are expected to capture the social structure in order to better support the cooperative work. Moreover, since work is a socially organized activity, where the actual behavior differs from the way it is described by who does it, it is mandatory not only to rely on interviews but also in observations of everyday activities at the workplace where technology is supposed to be inserted. In that sense, it is similar to the Contextual Design approach of Holtzblatt and Beyer [6, 7] in which CD gathers design data by having designers watching people do their own jobs, interspersing observation, discussion, and reconstruction of past events. In the next Section we present our approach which aims at helping designers in the process of user interface design requirements gathering.

3 PROTAGONISTS TASK-BASED DESIGN

In this Section an approach to gather user interface design (UID) requirements is presented. Within this context, we are concerned with the interface component of an interactive system. More specifically, our interest lies on the way people interact with computers, i.e. the control aspects. In that sense, Myers et al [12] defines Human-Computer Interaction (HCI) as *"the study of how people design, implement, and use interactive computer systems and how computers affect individuals, organizations, and society"*. Furthermore, Myers [11] points out a set of difficulties for designing user interfaces. They are:

- the difficulty in knowing tasks and users;
- the inherent complexity of tasks and applications;
- The variety of different aspects and requirements;
- theories and guidelines are not sufficient;
- difficulty of doing iterative design.

In order to tackle these afore-said difficulties and to better support the process of gathering user interface design requirements, we consider tasks carried out by protagonists. Protagonists are all the components that play roles in an interaction scenario. Such protagonist are both user(s) and system components. These components can be viewed as software components with major functionality roles. Herein, we are concerned with the tasks carried out by protagonists. In this respect, human beings often utilize computers to carry out their tasks. We can have tasks as guiding elements for a UID requirements gathering.

This approach for UID requirements gathering is proposed due to the diversity of users with different knowledge levels as well as a variety of both interaction styles and implementation technology. This need becomes even greater in social computing scenarios discussed earlier on in this paper due to the inherent complexity. Moreover, interface technologies are gradually migrating from command-oriented to noncommand-oriented interfaces as pointed out by Nielsen [13]. A further note is that human beings interact with computers easily if human-computer interaction is based on metaphors of their daily activities. Thus we use protagonist task descriptions so that the requirements can reflect known user metaphors.

We also advocate the need to start task-based requirements gathering at a high abstraction level where only user intentions are captured. This makes the process of requirements gathering easier for handling and upgrading as the refinement process proceeds until detailed requirements are obtained and implementation decisions are made. Besides the use of a protagonist 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 such a process augments the integration between users and designers because they can communicate between each other using a common language derived from such metaphors. 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.

Note that abstraction is a key principle in any engineering discipline and software engineering might not be an exception. Such concept together with modularity [16, 17] are used as design guidelines in our approach. Therefore, a UID requirements gathering based on metaphors of

the domain context aims at facilitating the user involvement in the design process. Below we present the steps needed for our approach.

1. *System description* - Obtain a system description of the system under development (SUD).
2. *System protagonists identification* - These protagonists can be both user and system components depending on decisions of what parts of a working process should or should not be carried out automatically. To do so, we use the Protagonist Action Notation (PAN) [19].
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 Task Coordination Models (TCMs).

So far we have presented the main steps of our protagonist task-based UID requirements gathering approach. To support our approach, we illustrate in the next Section how to carry out the UID requirements gathering through a multiuser Web game example.

4 NETCONNECT4 - A MULTIUSER WEB GAME

This Section presents an example of social computing related to entertainment application. It is so-called NetConnect4 system. NetConnect4 is a simple *multiuser Web game*. We start providing background issues on multiplayer games and game theory to underlie the presentation of this case study and then we illustrate the use of our approach to describe interaction aspects of NetConnect4.

4.1 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 to 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 multiuser 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.

4.2 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 [18] and Luger [10].

4.3 Designing NetConnect4

NetConnect4 is a multiuser 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 protagonist task-based approach to gather UID requirements.

4.3.1 System Description

The game description is as follows.

> NetConnect4 is played with a rectangular board that contains a 7x6 array of 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.

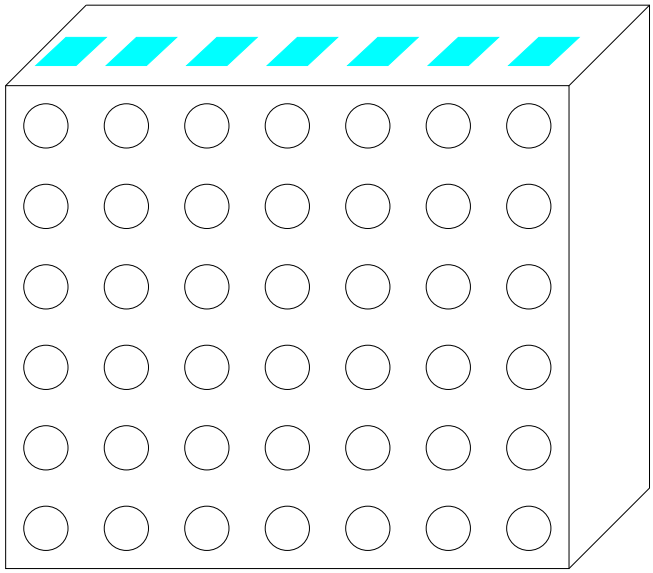

Figure 1: Board of NetConnect4.

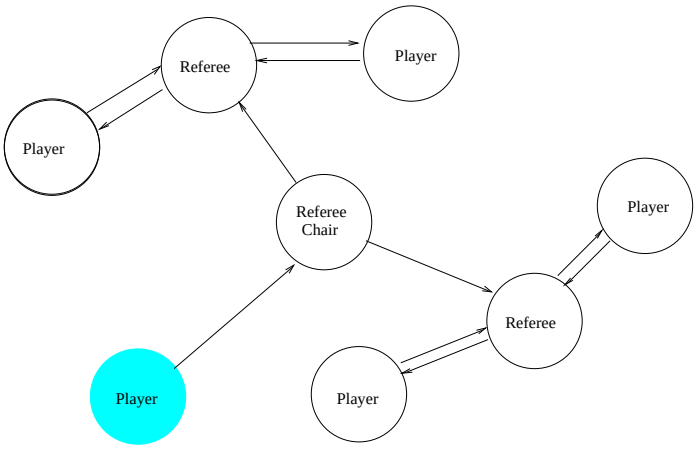

Figure 2: NetConnect4 protagonists.

After obtaining an system 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.

4.3.2 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. In that case, we need to capture not only user actions but also system-generated actions. Thus using a notation that captures only user actions constitutes a hindrance to the user interface design 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 (protagonist-based notation (see [19] for more details) to provide designer with a high abstraction level notation that allows him to document the requirements gathered.

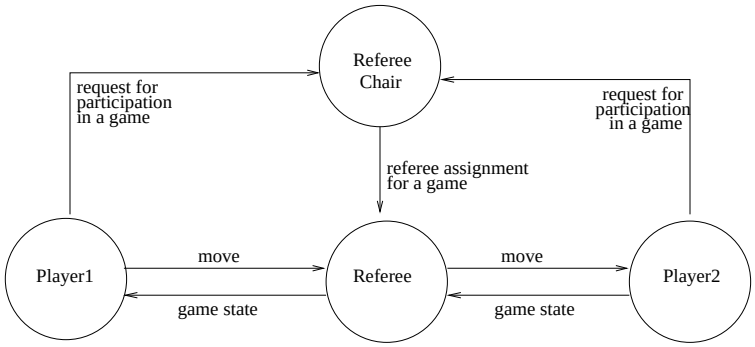

Figure 3: Scenario with NetConnect4 protagonists.

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.

4.3.3 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 we deal more specifically with protagonist actions.

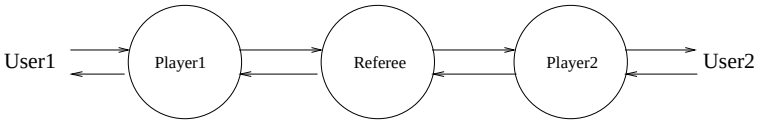

Figure 4: Interaction scenario for NetConnect4.

4.3.4 Identification of the System Architectural Model

At this step, we do a system description in terms of their 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.

4.3.5 Task Description of NetConnect4

From the identification of the protagonists 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 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 illustrates 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.

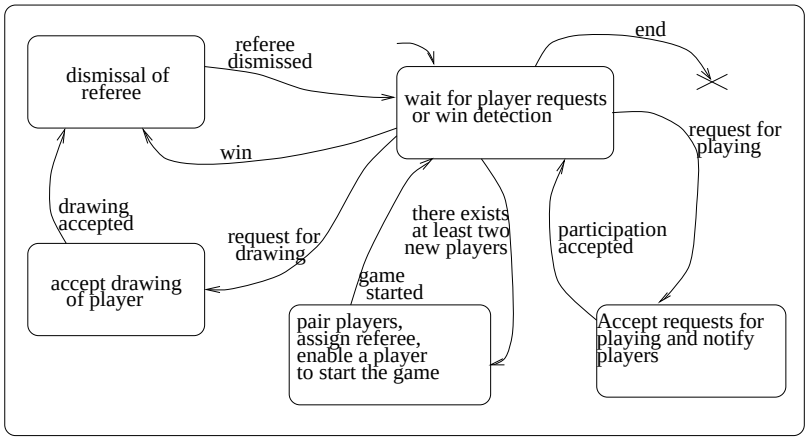

Figure 5: TCM for referee chair.

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 navigation between tasks. In particular, the termination of a task can motivate a navigation to another one or actions of other protagonists may as well cause task navigation in the TCM of the referee chair as when he receives a request from a player for drawing a game.

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 dismiss 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 navigation.

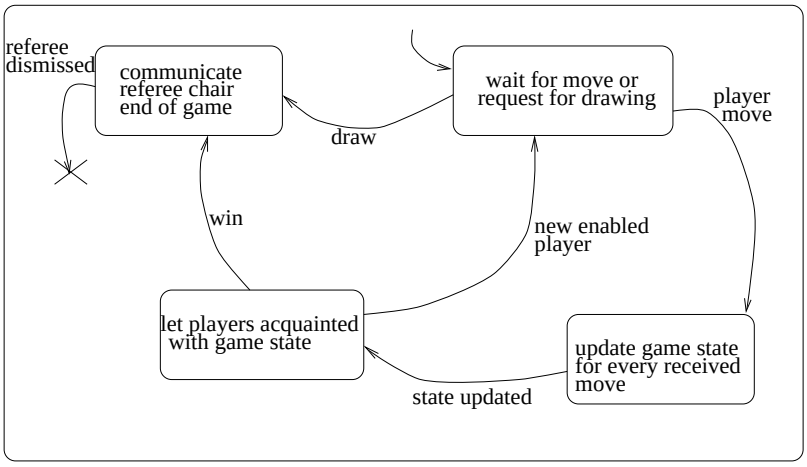

Figure 6: TCM for referee.

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. The tasks of a player is given below.

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.

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.

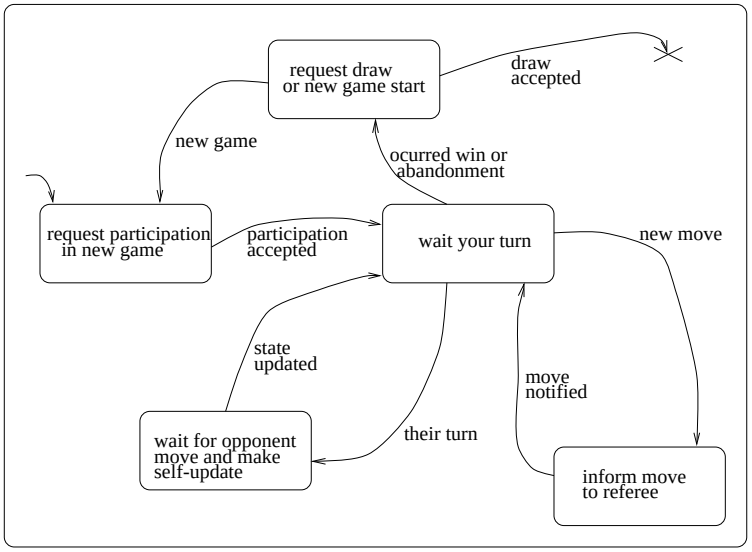

Figure 7: TCM for player.

5 CONCLUDING REMARKS

This paper presents an approach for gathering user interface requirements based on protagonist tasks. Target applications were those in social computing context. This emerging class of application has been a product of the Internet era where a diversity of users and interaction styles has required more and more from designers. One of our major concerns was to find out a way to identify the social scenario where the computer technology can provide support.

Furthermore, using protagonist tasks facilitates and motivates user participation. We also tackle the difficulty of having effective user participation within a design process by using a common language based on protagonist tasks to enable a clean communication between user(s) and designer (team).

Social computing is not out there to reflect the designers' understanding of an activity but to provide computer support to the everyday working needs of our society.

An illustration of our approach was given through a multiuser Web game. This scenario describes the complex activity of managing the entire game. We may have a number of users in great expectation to get paired with someone else in order to play. Besides taking care of that, the system also needs to either assign referees to mediate games or supervise ongoing games. Despite of being a small example, it conveys several difficulties inherent to a social computing application discussed in Section 1.

The work presented in this paper is part of a project which aims at both mapping a HCI design into an interface software design [19] and providing an interface development methodology. Therein, we address the derivation of interface software design from the HCI design. Interface software design is the other major activity we have in an interactive system development. We have been using the Xchart specification language [8, 9] 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/

6 ACKNOWLEDGEMENTS

The authors would like to thank the financial support from CAPES.

References

[1] D. A. Agarwal, S. R. Sachs, and W. E. Johnston. The Reality of Collaboratories. *Computer Physics Communications*, 110(1–3):134–141, May 1998.

[2] J. Blomberg, J. Giacomi, A. Mosher, and P. Swenton-Hall. Ethnographic Field Methods and Their Relation to Design. *D. Schuler and A. Namioka (Eds.): Participatory Design: Principles and Practices, Lawrence Erlbaum Associates, Publishers*, pages 123–155, 1993.

[3] A. Clement and P. Van den Besselaar. A Retrospective Look at the PD Projects. *Communications of the ACM*, 36(4):29–37, June 1993.

[4] IEEE Computer. Special Issue on Digital Libraries Initiatives. *IEEE Computer*, 29(5), May 1996.

[5] J. Greenbaum. PD: A Personal Statement. *Communications of the ACM*, 36(4), June 1993.

[6] K. Holtzblatt and H. Beyer. Contextual Design: Principles and Practice. *D. Wixon and J. Ramey (Eds.): Fields Methods Casebook for Software Design, Wiley Computer Publishing*, pages 301–333, 1996.

[7] K. Holtzblatt and H. Beyer. Contextual Design: Defining Customer Centered Systems. *(Reading), Morgan Kaufman Publishers*, 1998.

[8] F. N. Lucena. XCHART: Um Modelo de Especificação e Implementação de Gerenciadores de Diálogo. *PhD Thesis, Institute of Computing, State University of Campinas, Brasil*, Dec 1996.

[9] F. N. Lucena, M. M. Harada, and H. K. E. Liesenberg. Operational Semantics of Extended Statecharts (XCHART). *Proceedings of the 3rd Workshop on Logic, Language, Information and Computation, Brasil*, May 1996.

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

[11] B. A. Myers. Why are Human-Computer Interfaces Difficult to Design and Implement? *Technical Report CMU-CS-93-183, Computer Science Department, Carnegie Mellon University*, July 1993.

[12] B. A. Myers and et al. Strategic Directions in Human-Computer Interaction. *ACM Computing Surveys*, 28(4):794–809, Dec 1996.

[13] J. Nielsen. Noncommand User Interfaces. *Communications of the ACM*, 36(4):83–99, April 1993.

[14] Comm. of the ACM. Special Issue on Health Care Information Systems. *Communications of the ACM*, 40(8):80–117, Aug 1997.

[15] Comm. of the ACM. Special Issue on Digital Library. *Communications of the ACM*, 41(4), April 1998.

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

[17] D. 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.

[18] S. Russel and P. Norvig. Artificial Intelligence: A Modern Approach. *(Reading), Prentice Hall*, 1995.

[19] A. M. Silva Filho and H. K. E. Liesenberg. Capturing Computer-Human Interaction Design via the Protagonist Action Notation. *Proceedings of the 18th Brazilian Computer Society Annual Conference, Belo Horizonte, MG, Brasil*, 1:276–296, Aug 1998.

[20] Yahoo Web Site. Interactive Web Games/Multi-User Games. *Available at <http://www.yahoo.com/Recreation/Games/Internet_Games/Interactive_Web_Games/Multi_User_Games/>.*, 1998.

[21] 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.