USERS INVOLVEMENT IN REQUIREMENTS ENGINEERING PROCESS

Daniela Elena Herlea
Knowledge Science Institute
University of Calgary
Calgary, Alberta, Canada T2N 1N4
danah@cpsc.ucalgary.ca

Abstract

Software requirements elicitation is the process where the customers' needs in a software project are identified. This process is regarded as one of the most important parts of building a software system because during this stage it is decided precisely what will be built. However, requirements gathering needs close interaction between developers and end-users of the system. If developers and end-users are in different organizations or different cities, meetings can be costly, inconvenient and infrequent. This leads to problems of communication, which can greatly impact the quality of the elicited requirements. Well known requirements engineering methodologies are presented in this paper and the degree of the user involvement in the process of requirements negotiation is discussed. The system called TeamRooms integrates the rich applications and interfaces found in the existing real-time groupware applications, providing a persistent work space suitable for synchronous and asynchronous collaboration. TeamRooms may be used as a method of distance collaboration between the knowledge engineer, the developer who elicits the information, and the domain expert, the knowledgeable end-user. It encapsulates both structured and unstructured work through its applications and also takes into account individual and group work.

REQUIREMENTS ENGINEERING PROCESS

Software requirements engineering is the process of determining what is to be produced in a software system. In developing a complex software system, the requirements engineering process has the widely recognized goal of determining the needs for, and the intended external behavior, of a system design. This process is regarded as one of the most important parts of building a software system:

> " The hardest single part of building a software system is deciding what to build. No other part of the conceptual work is as difficult a establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No part of the work so cripples the resulting systems if done wrong. No other part is more difficult to rectify later" (Brooks, 1987)

Tracing the emergence of significant ideas in software development over the years, one can observe that in the '60s the attention was on coding, in the '70s on design and in the '80s on specification. Jawed Siddiqi agrees that requirements engineering will be a critical issue in the '90s

(Siddiqi, 1994). Siddiqi challenges that "requirements describe the 'what' of a system, not the 'how'". However, in the process of requirements engineering it is often difficult to state the real 'what' level of a system because one person's 'how' may be another person's 'what' and conversely. In this perspective, the requirements engineer faces a complex problem, in meeting the needs of the customer and at the same time meeting the needs of the designer.

Requirements engineering is an important aspect of any software project, and is a general term used to encompass all the activities related to requirements. The four specific steps in software requirements engineering are:

- ***requirements elicitation***
- ***requirements analysis***
- ***requirements specification***
- ***requirements validation***

Although they seem to be separate tasks, these four processes cannot be strictly separated and performed sequentially. Some of the requirements are implicit in the working practices, while others may only arise when design solutions are proposed. All four are performed repeatedly because the needs are often impossible to realize until after a system is built. Even when requirements are stated initially, it is likely they will change at least once during development, and it is very likely they will change immediately after development.

Requirements engineering, a negotiation process

Requirements engineering is one of the most important steps in a software engineering development process, because an effective elicitation process will provide a set of requirements that can be used by the software development team.

We first examine the participants in a requirements elicitation process. There are three main categories of participants: the developer, the user and the customer. In addition, a major requirements elicitation can involve other people such as lawyers, government standards organizations and so on. From this point of view, this process may be regarded as a negotiation process, where potential "users" of the system explore and fully understand the requirements, agreeing on what they *want* and what they *need,* while the developers become confident that they are solving the right problem. A successful requirements process includes effective ways to encourage partnerships between customer and designers and team members. The process gathers information, validates everyone's understanding of the information gathered and communicates open issues.

A successful requirements definition process, as Koltzblatt and Beyer (1995) concluded, is about people talking effectively to each other. The process of requirement-gathering has more to do with people's ability to cooperate and modify how they work than any other factor. The design team's job is to discover and identify the structure of a work domain they need for their design and this is possible only following a process where requirements are regarded as "...emergent, in sense that they do not already exist, but rather emerge from interactions between the analyst and the client organization" (Goguen, 1992). In a requirements elicitation process no single person has the complete picture (Checkland, 1981); each individual user may have only a limited view or perspective of the system to be built and, on the other hand, each user may have different needs or

different priorities which have to be understood by the design team. The process should enable the design team to envision possible solutions to the change within the customer organization, at the same time as specifying the problem. It is the relationship between designers and customers that determines how well the design team understands the customer problem.

Because of the involvement and the necessity of mutual control of the process by all the players, requirements elicitation is, in most cases, a difficult and imprecise process. There are some difficulties which arise in the requirements elicitation process. One important issue is that the lack of appropriate communication within the group greatly affects the final document of requirement gathering. Linda Macaulay (1996) states

> "that there is a failure to realize that appropriate human communication mechanisms need to be established as part of the requirements process. If different interest groups do not communicate effectively with each other, each will seek to exert power and influence over the others... (Gasson, 1995; Markus and Bjorn-Anderson, 1987)"

User participation

Research on the role of user-developer communication has revealed that the transfer of users' understanding and insight of business practice to developers who will translate it into a working computer system is the key to effective communication benefiting both parties. Bostrom (1989) argues that participants "bring different frames of reference to the situation and ... the ability to integrate these diverse views into a shared, accurate and complete model of the desired system is essential to successful projects".

User participation plays an important role in dealing with the development of complex systems and "users and developers will benefit from close interaction by exchanging views, identifying, and resolving conflicts as well as sharing information necessary to effectively accomplish the task" (McKeen1994). The study of the "participation-satisfaction relationship" hypothesis led to the conclusion that system development "requires that the appropriate users participate in the project at the appropriate stage and in a manner that enables a meaningful contribution".

McKeen, Guimaraes and Wetherbe (1994) focus their research on user participation as a determinant of system success. The term "user participation" is used when "referring to the various design related behaviors and activities that the target users or their representatives perform during the system development process". Their work looked at the factors that affected the relationship between user participation and user satisfaction. They revealed the importance of improving the user participation within system development to improve user satisfaction.

Important approaches for encouraging customer participation include socio-technical systems theory (STS) and participatory design (PD) and they support the idea on *how* the customer should participate in the process of requirements identifying. "Many of the best ideas for new products and product improvement come from the customer or end user of the product"(Keil, 1995)

Carmel, Whitaker and George's approach (1993) to customer participation in software development introduces the idea of selecting one or more customer-developer links. These links are defined as "channels that allow customers and developers to exchange information" and the authors argue that "defining and counting links is an important first step toward quantifying the domain of customer participation". Following a multiple-case study on actual software development projects, they introduce a new classification of links: direct versus indirect links.

They support the idea that, from a communication perspective, "direct contact between customer and developers is preferable to indirect contact because it decreases filtering or distortion that may occur". Further more, they observe a strong correlation between the number of links used and the relative success of the projects initiated. The primary difference between various user involvement methodologies and requirements engineering techniques lies in the degree to which the users participate in the emerging system (Carmel *et al.*, 1993).

In fact, the user involvement may fall into three main styles:

- *consultative design*, which leaves decision-making power to information system staff.
- *representative design*, which involves selected user representatives in the actual design formulation and decision making.
- *consensus design*, which fully shares responsibility for the system development process between the users and developers.

Joint Application Design (JAD) and ***Participatory Design (PD)*** methodologies emphasize group processes in developing the requirements representation. These are well-known methodologies in requirements negotiation that promote cooperation, understanding, and teamwork among the various user groups and information system staff. They will be described in detail later in this paper.

Requirements engineering methodologies

The Inquiry-Based Requirements Analysis Model views the analysis process as essentially inquiry-based — "a series of questions and answers designed to pinpoint where information needs come from and when" (Potts, Takahashi and Anton 1994).

The Inquiry Cycle Model, a "formal structure for describing discussion about requirements", addresses the case of mass-market-driven product development, for which there may be no clear customer authority. The term used in this model is "stakeholder", anyone who shares information about the system, its implementation constraints or problem domain. The model consists of an integration of three phases (*requirements documentation*, *requirements discussion* and *requirements evolution*), where the stakeholders write down their proposed requirements, challenge them by attaching typed annotations and then refine the requirements when change requests are approved.

Potts *et al.*(1994), propose using *scenario* as an analysis technique in *requirements documentation*. A scenario is defined as "a description of one or more end-to-end transactions involving the required system and its environment". In this context, they argue that although scenarios describe the system's behavior only in specific situation, "analyzing specific scenarios gives stakeholders insight into general requirements and helps in the refinement process."

During the *requirements discussion* phase, answers to questions about requirements provide stakeholders with a deeper understanding of the requirements and help them notice inconsistencies or missing requirements. The *requirements evolution* is the phase where the decisions about changing or freezing a requirement are taken.

Amore, (the Advanced Multimedia Organizer for Requirements Elicitation), a prototype modeling environment which includes methods and techniques used in the area of requirements elicitation, is introduced by Christel, Wood and Stevens (1993). "Typical software systems contain many thousands of requirements. These requirements are derived from many sources and in many formats, hence a tremendous amount of complex raw data comprise the source material for a given system." (Christel, 1993). AMORE is interested in modeling those vast amounts of raw source material as requirements, and provides access to knowledge about the problem domain, as well as tools for the capture, modeling, analysis and manipulation of raw requirements data.

There are many potential users of the AMORE system, including customers, elicitors, system and domain analysts, designers, testers, maintainers or managers. Each user perceives the raw requirements in a slightly different way. AMORE as a modeling tool provides a model for the capture of information related to requirements. It helps organize the information space in a manner that will support reasoning about the requirements by facilitating browsing, navigating and searching through them. Following an interview or an analysis of video transcripts of the interview, the elicitor is able to identify and store segments of the interview which support certain requirements and associate those video segments with the requirements.

Because of the tremendous amount of requirements in large scale systems, AMORE provides a model of organizing multiple requirements into hierarchical organizational structures, the most common of which are leveled data flow/control flow diagrams and object hierarchy diagrams. The elicitor navigates through AMORE by means of the selected organizational structure. These structures contain requirements located at primitive nodes and, during the eliciting process the hierarchy can be modified, moving, adding or deleting branches as necessary. AMORE uses the System for Access to Information and Learning (SAIL) to provide assistance to elicitors, designers, customers, anyone who need to examine and manipulate requirements.

Soft System Methodology and Ordit. One of the basic modeling techniques in software engineering is Checkland's Soft System Methodology (SSM) which is a methodology for "soft" problems, the ones which deal with 'what' of the problem situation, not with 'how', like "hard" problems. Checkland's methodology recognizes groups who decide on requirements and negotiate contractual issues between customers and suppliers. It suggests managing this kind of process by guided intervention in meetings when people with different objectives and perceptions discuss and provide the environment for understanding of the problem at hand.

"Conventional systems analysis has largely focused on defining information processing requirements rather than looking at informational technology from a wider perspective...This problem is partly addressed by soft systems approach (Checkland, 1981) but it is taken a step further by the ORDIT project, which takes a socio-technical approach (Mumford, 1983) in which the system is viewed as a whole by placing it within the broad operational environment, with the user as an integral part of the system."(Goguen, 1992). Requirements cannot be fully separated from their social context, as "it only become clear what the requirements really are when the system is successfully operating in its social and organizational context" (Dobson, Blyth and Chudge, 1994).

ORDIT, which is an ESPRIT II project, focuses on the need of the organization as opposed to the individual, and agrees on the importance of social context in the requirements definition process. **ORDIT**'s philosophy is that design methods appropriate for technical systems cannot

simply be applied to socio-technical ones, and that consideration must be given equally to both human and technical issues, with success being seen as the construction of a relevant socio-technical system that meets the "real" requirements of the organization (Dobson, 1994). This methodology discusses human requirements of socio-technical systems, and demonstrate how these are linked to the technical features of the system design. "The traditional notion of the software development life cycle, with requirements capture being completed before the design stage, is no longer satisfactory. Requirements capture and design are now seen to be symbiotic. The initial set of requirements needed to start off the design process is gradually refined into a systematic and coherent statement of requirements hand-in-hand with the refinement of the design." (Christel, 1993)

The purpose of the **ORDIT** method is to support the identification and transformation of organizational requirements into precise statements which can be operated upon by system designers. Compared with other methods, **ORDIT** 's modeling may be started at a very early stage to help in exploring the system boundaries and in identifying stakeholders. This process is in contrast to the traditional "waterfall" approach to modeling, in which the output from one stage forms the input to the next stage, and so on, with all the stages following a predetermined order.

The process has four broad interactive component subprocesses:

- *scoping*, *modeling*, *requirements capture* and *solution option.*

The ***Scoping*** process establishes territory and determines the important players; the ***modeling*** process is intended to represent the current understanding of the socio-technical system by producing a set of models; ***requirements capture*** is the process of the interaction with the requirements owners and has to be interactive, and iterative and ***solution option*** is the place where requirements elicited are used to generate a possible design into a socio-technical space.

In any requirements determination activity, all four phases will be visited, often more than once. The **ORDIT** philosophy is its advocation of involving policy makers and problem owners throughout the design of the system. Problem solvers help the problem owners understand the problem, and the problem owners help the problem solvers understand the implications of possible solutions (Christel, 1993).

Nature is a large ESPRIT project initiated in Aachen, Germany. It is regarded as a new trend in requirements engineering because it approaches the requirements definition process with the idea that the environment in which requirements engineering has to operate is continuously changing. "The traditional approach was quite successful as long as the software market was immature, organizations and their software systems were relatively stable over time" (Bostrom, 1977). NATURE defines a framework which builds three specific theories. The requirements domain theory gives advice on what context knowledge is relevant and how to organize it. The requirements process theory offers a unified process meta model in which a small set of building blocks covers a larger spectrum of process guidance strategies with more flexibility than other software process or workflow models. The knowledge representation theory aims at defining what domain and process knowledge to capture, and how to manage this knowledge using an effective mix of informal, semiformal and formal representations (Jarke, Pohl, Domges, Jacobs and Nissen, 1995).

Joint Application Design (JAD) is a workshop approach to system design which has become the most common user involvement methodology in North America. Over years information systems organizations realized that a methodology with a high degree of user involvement would lead to better systems, and they found that system in JAD (Carmel, 1993). It was developed at IBM in 1977 and has been applied successfully on hundreds of projects. The JAD methodology involves a structured and disciplined session led by a session leader. The JAD approach is based on communication through documentation, fixed requirements and rules of work enforced through methods. In theory, JAD supports the entire System Development Life Cycle (SDLC). In practice, the requirements definition process has proven to be the subject of most meetings. This may be because this stage has the greatest impact on future system development. However, JAD meetings are also run throughout the SDLC when using an iterative approach (e.g. prototyping). In theory, JAD is a very structured meeting session which supports user involvement of all parties affected by the information system. The "typical" JAD room contains boards, overhead projector, flip charts. The session facilitator, who has the pivotal role, tightly controls the course of the meeting. The participants are encouraged to go through the room and fill information onto wall charts. Although JAD stresses cooperation in the form of a team, its meetings involve low and middle managers, the people with decision-making authority. Only after the session leader gains enough influence, end users join the collaborative team. Another important point is that information system staff do not contribute in the meeting, just sit as quiet observers. Only recent practices involve the information system and users in a continuous dialog. Ehn argues that the "American" concept of team is a poor compromise that takes from workers without giving them anything in return (Keil and Carmel, 1995). The JAD methodology evolved over the years. Today, meetings are conducted using CASE tools: for depicting data-flow diagrams, Entity-Relationship diagrams, state transitions and other diagramming techniques and screen painters. Another technology in the JAD community is groupware (Mumford, 1979).

As Carmel et al (1993) stated, "JAD represents a movement toward more collaborative practices to enhance the viability of given technical goals. PD represents a movement toward more technical practices to enhance the viability of given goals". Both methodologies are well-known for stressing a high degree of user involvement as imperative to good design of information systems.

Participatory Design (PD), the "Scandinavian approach" to system development, focuses on much stronger involvement of the users than JAD does, facilitating a mutual learning process between users and designers, and joint experiences into a simulated work situation.

As Greenbaum observes, from a philosophical perspective, "involved acting - not detached reflection - is our fundamental way of being" (Checkland, 1981). The developers and people at their workplace do not experience the same things and this means that they cannot easily understand each other's experience. PD focuses on lower-level end users who are introduced to the developers workplace and learn about technical possibilities through "joint applications". The same way, during requirements definition process, the designers try to collaborate with workers at their workplace and both are transformed by learning from one another. The information system technical staff is represented by designers in PD, and they act both as facilitators and technical advisors. The designer has a dual role in PD, and this encourages creativity of the participants. PD doesn't have an invariant structure and this gives it more flexibility; there is no set of practices in Participatory Design approaches. Users and system experts share the responsibility for the quality

of the design proposal. PD involves a great deal of trust in the users and they therefore become responsible for the system they get.

User-development interaction

The most important aspect of user-development interaction is the mutual learning and cooperation among them. Some methodologies assume that the transfer of knowledge between users and designers can be achieved in the environment of a meeting room (e.g. Inquiry based Analysis Model, JAD). At the same time, other methodologies (e.g. PD) foster the full collaboration of stakeholders through a process where users are directly faced with the designers' work situation and conversely, and by the end of the elicitation process everyone learned about real needs of users and technical possibilities.

In this context, success in meeting the real needs of the software system is contingent upon the ability of users to clearly specify what their requirements are. For this reason, requirements definition needs close interaction between developers and end-users of the software. It is critical that requirements engineering tools must support collaborative development of the software requirements negotiation. Requirements definition should be an iterative process where, through reflection and experience, users become familiar with the technology and developers become familiar with the work. For example, scenarios, prototypes or mock-ups which provide the opportunity for the users to "experience" the new technology and for the developers to "experience" the work practice.

In most cases people involved in this process can meet in the same meeting room that implies that: team members are working simultaneously, are generally working on the same thing, are in continuous, real-time communication and aware of each other's activity. A facilitator coordinates the meeting. The analyst plays the key role in the process of elicitation of requirements from the problem "owners". After prototype evaluation each category of participants can easily communicate their suggestions regarding the system's performance. The collaboration is at the highest level in this case.

Cooperation between participants in the process is quite difficult, even if they meet in a meeting room. The process involves a social network of people with different professional backgrounds and different views over the system that must be built. If the participants in the process are in different organizations or different cities, meetings can be costly, inconvenient and infrequent. One technology that may help solve this problem is *groupware*. Groupware systems are computer programs that enable users to work on shared documents over time or distance. They solve the problem of "synchronous distributed interaction". Real time groupware tools are available today that can assist the real time interaction of participants in a facilitated workshop, including the generation, collection, analysis, decision making and documentation processes that are the basis of participant interaction. Participants located in different physical locations can share a computer workspace in which the work of the entire group is presented to each member with continuous real-time update. The way these systems are developed is based on the "What I See Is What You See" concept. Each group member sees the same view of the data on their local workstation. There are some limitations, but the value they provide is substantial.

TEAMROOMS, A GROUPKIT APPLICATION

The groupware system called TeamRooms provides an electronic equivalent of a team room for groups that are either co-located or at a distance. More about TeamRooms as a GroupKit application may be found in Roseman and Greenberg (1995). It is implemented using an extended version of the groupware toolkit GroupKit (Roseman and Greenberg, 1996). Facilities offered by the GroupKit's Application Programming Interface are preserved in TeamRooms. This enabled the developers to move the existing GroupKit applications to TeamRooms and rapidly create new ones. It combines the rich applications and interfaces found in the existing real-time groupware applications, providing a persistent work space suitable for both synchronous and asynchronous collaboration. It encapsulates both structured and unstructured work through its applications and also takes into account individual and group work.

TeamRooms currently runs on Unix based, Windows and Macintosh machines and supports a client-server arrangement. Communication between users is possible even when they do not share the same physical space. Once connected to a central TeamRooms server, users gain access to the electronic rooms. The central server acts like a message bus. The messages between users are routed through the central server, which has an understanding of GroupKit constructs but no knowledge about the specific applications that make up TeamRooms. The central server is important in supporting user authentication – when users log in --, which was more important for TeamRooms than in the more open arrangement found in GroupKit.

TeamRooms as a negotiation *place* for requirements

In this paragraph the features of this system and see how it can assist a requirements negotiation process are discussed. The business case and the roles played by the participants within the organization and the requirements tracking process are shown in Figure 1.

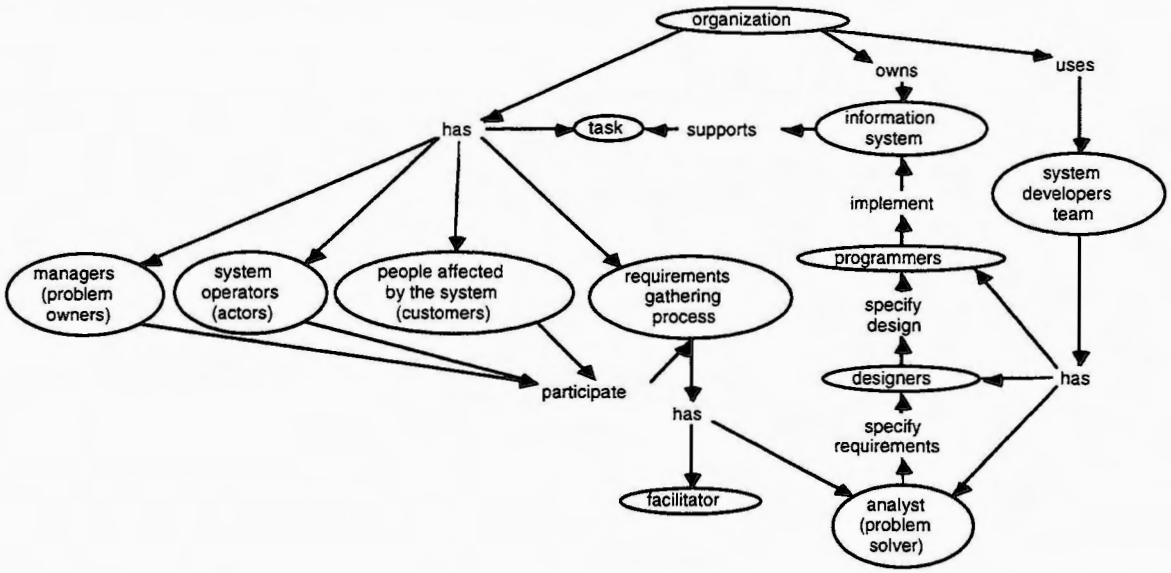


Figure 1 Concept map showing the roles played by the participants within the organization and the requirements tracking process, according to Checkland's methodology

Starting Up the session. Simulating a real physical meeting space, TeamRooms has many rooms. Each room contains both generic communications tools (a chat tool and a backdrop acting as a shared whiteboard) and any number of applets needed to support the group's work. Applets are special-purpose applications, designated for more specific needs of a group. TeamRooms supports any type of application which can be constructed in GroupKit, such as meeting tools, drawing tools, text editors, card games and so on. When a user starts up the system, he or she is prompted for a user name and a password. If he is among the work group permitted to use the system, he or she will be connected to the TeamRooms central server.

TeamRooms may successfully support the requirements elicitation team working at a distance, by helping to reduce barriers to communication. The example that follows assumes that there are four main categories of users of the system: the meeting facilitator, the analyst, the design team and the customer representatives. The requirements negotiation process is about navigating this electronic workspace. The participants may enter any room in the workspace and work with other users in the same room. They can draw objects using the shared whiteboard or outline and organize ideas as they would do in a physical meeting room. In the process of requirements gathering, members of the team can store partial or final documents in databases. Any information can be later retrieved and updated if necessary.

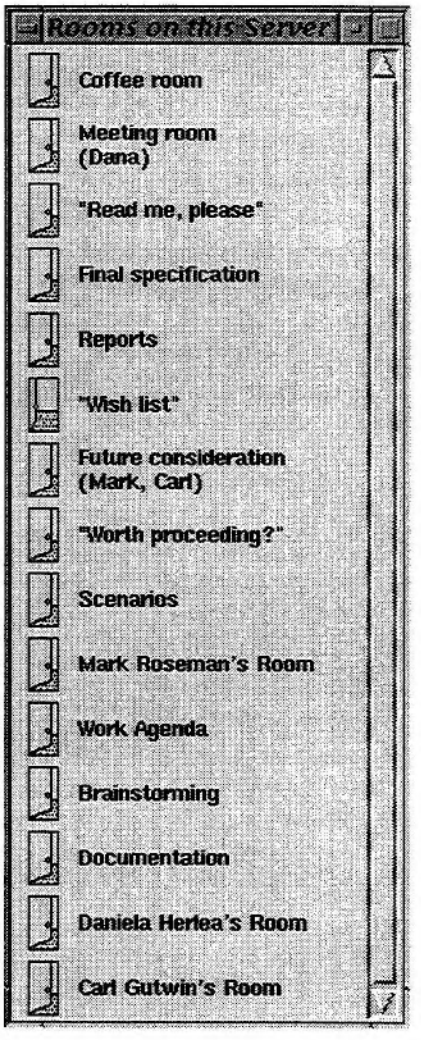


Figure 2. Rooms in the meeting workspace

The rooms in the system. Figure 2 shows the existing rooms in the workspace. Users of the system can enter other rooms by selecting one of the room from this list. Each room supports multiple groupware tools. Along with the generic tools, users can include a set of applets into the working rooms. According to their activities, users of the system may create a variety of rooms to meet their specific purposes. A description of the rooms follows and the roles played by the participants in the process are discussed.

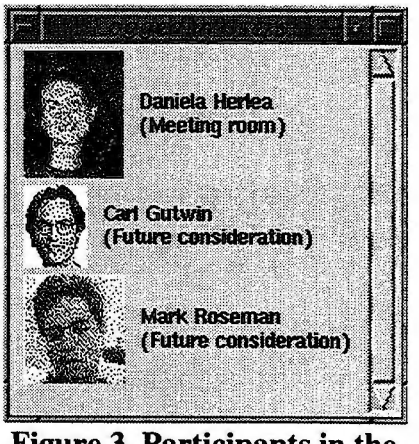


Figure 3. Participants in the process

Awareness of other users in the work space is very important in simulating the real physical space. TeamRooms provides the facilitator with tools to manage effectively the shared space. The facilitator may have the knowledge of the room and action of each user in the working space. Figure 3 shows the list of users currently connected to the server, as well as the room each user is currently working in. This enables the meeting facilitator and other users to locate the other participants and make contact with them. Telepointers showing each participant's position is another way of letting the facilitator know about each user's actions. The working space may be larger than the user's display. A room

radar overview provides a miniature overview of the entire room when the user changes the location of his or her view onto the workspace to suit the needs of the immediate task. The radar shows the position of each user's viewport in the room, telepointers indicating the location of their mouse cursor. Each time when a user navigates the room and manipulates applets, the radar tracks his actions.

COLLABORATION OVER THE ELECTRONIC WORKSPACE

Participants in Requirements Elicitation Process

In the process of requirements elicitation there are four main categories of participants:

- the **facilitator**, who acts as a chairperson of the meeting, has a critical role in organizing the work of the requirements negotiation team.
- the **users**, who are people involved in using the system, are the "owners" of the problem.
- the **analyst**, who is a representative of the design team, has a key role in the transfer of the requirements from the "problem owners" to the design team.
- the **design team**, who are the system implementors, are responsible to meet the elicited requirements.

Rooms in the requirements elicitation electronic space

The rooms in the system were created to fully support the requirements gathering. Each room has a specific purpose in the process. Users may enter the rooms, collaborate with the others in the same room. There is no defined order to navigate the rooms. Requirements elicitation is an iterative process, where the users of the system participate in meetings and discussions held in different rooms, at different times; they enter the same room many times in the process' life cycle. The following rooms are meant to support the collaboration over the electronic meeting space:

- ***Brainstorming Room***, used to record all "quick ideas" generated throughout the meeting.
- ***Meeting Room***, used as a general working space that gathers the participants in an informal environment.
- ***Work Agenda Room***, used to keep an agenda of the meetings, used mainly by the facilitator.
- ***Documentation Room***, used to record the elicited requirements.
- ***Scenarios Room,*** used to build scenarios of future situations.
- ***"Wish List" Room***, used to store the initial list of requirements.
- ***Future Consideration Room***, used to record the intermediary results of the process.
- ***Final Consideration Room***, used to store the final document of requirements, used by the design team in implementing the system.
- ***Reports Room***, used to communicate and inform about the design team progress.
- ***"Worth Proceeding?" Room***, used to "test" the prototypes against the requirements.
- ***Read Me Room***, used to leave notes for other participants in the process.
- ***Personal Rooms***, used to store users' own artifacts.

Collaboration in requirements negotiation place

The **facilitator,** who acts as a chairperson of the meeting, has a critical role in organizing the work of the requirements negotiation team. He or she enables effective communication, works as a project manager for the elicitation team. He can use the ***Work Agenda*** room to keep an agenda of the meetings, lists of activities and organize the work using the Outliner or the Note Organizer applets to create a hierarchical activity structure, or a Concept Map applet to identify priorities.

The organization **analyst** plays a key role in requirements transfer process. He or she is regarded as the "problem solver". The requirements generation process may be divided into to separate transfer processes.

Firstly, there is a transfer process between the analyst and the customer representatives, the "problem owner". The users are encouraged to navigate many rooms in generating requirements.

The ***Brainstorming Room*** may be used to record all "quick ideas" generated throughout the meeting. Users of the system can enter this room and leave notes or ideas on the whiteboard or organize concepts using the Concept Map tool.

The ***Scenarios Room*** may be used to build scenarios of future situations which involve the system to be developed. Scenarios are meant to reflect changes the new system brings about in the organization. The analyst helps the future users of the system to envision future changes in the work environment, to analyze the organizational policies, to gain understanding of the structure of the organization and to determine the scope and boundaries of the system. They can draw on the whiteboard, create activity structures using the Outliner tool or analyze relationships between activities and data, or data and users, using the Concept Map tool. Results of discussions may be stored in files. All participants in the process may retrieve these files later. In this regard, the ***Future Consideration Room*** may be used to keep track of the intermediary results. Figure 4 illustrates a "storage and future retrieval" area. Exploring the users' environment, the capture team collectively investigates the target users' organizational setting and identifies and describes what they do. They produce documents recording the shared view of the users' environment. They create objects reflecting the users' workspace and save them in a database. Records in this database are files containing information that describes these specific objects. Any user of the system may retrieve these files and discuss the list of the users' needs. Later in the process, members of the team may update this "user document", according to their understanding of the user environment.

The ***Documentation Room*** may be used by the analyst to record the elicited requirements. The analyst may decide to use the help of files to store draft plans of the requirements document. Validating understanding of the user environment with the system customers and formulating requirements based on this existing "picture" may be done in the general ***Meeting Room***. Any tool may be used to manipulate the existing information. As a next step, an initial list of requirements may be proposed and stored in a generic "wish list", in the ***"Wish List" Room***. Figure 5 contains a list of contents for a set of issues related to the requirements document.

The final document containing the list of requirements may be stored in the ***Final Specification Room***. It is accessed mainly by the analyst who translates the users' terminology into technical terms.

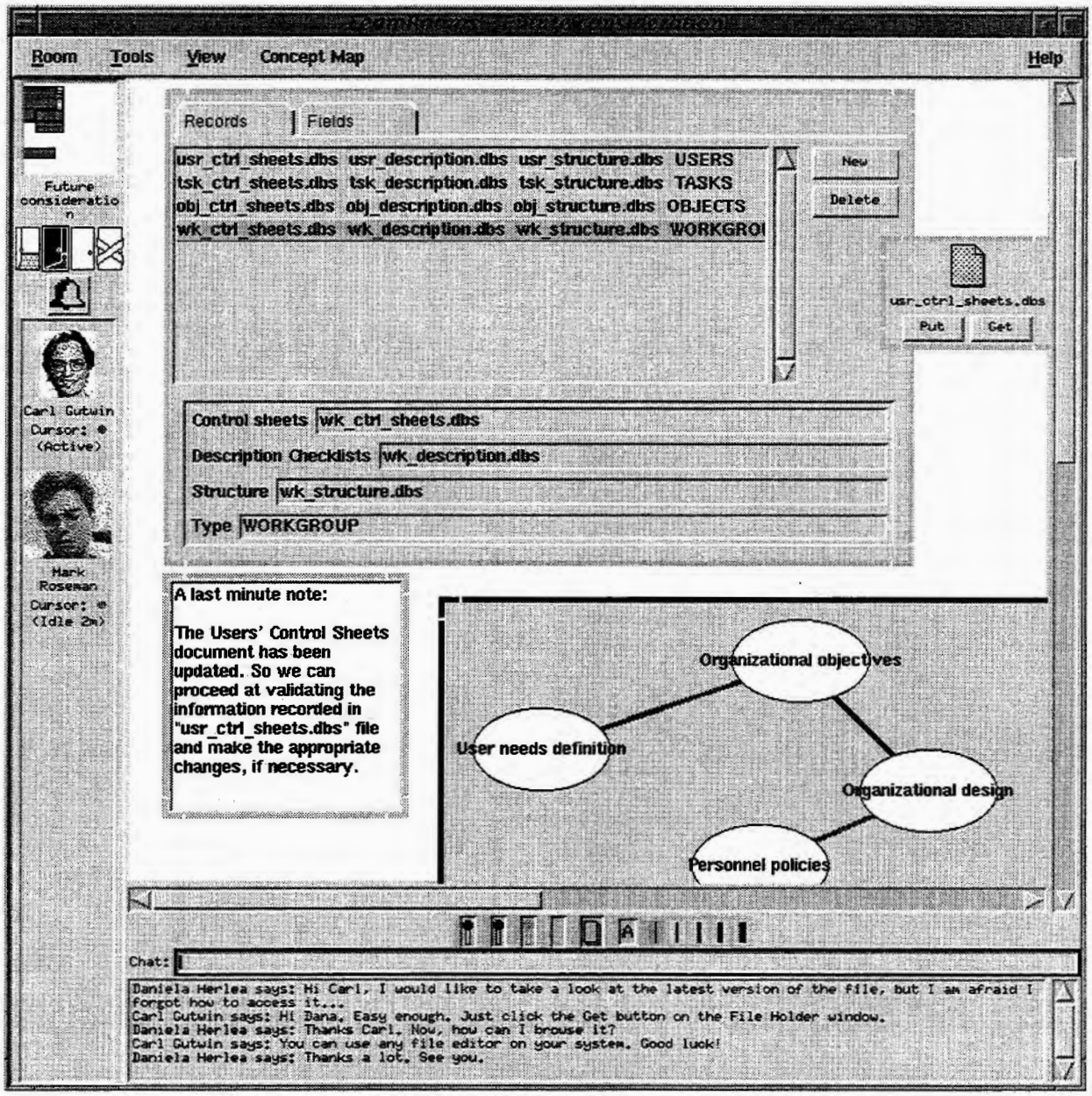


Figure 4. A "storage and retrieval" space for the user environment's objects

At a second stage, there is a transfer process from the organization analyst to the system designers in order to produce an initial set of prototypes. This transfer process takes place mainly in the technical domain. The design team uses the Final Specification document to build prototypes that meet the elicited requirements. Throughout the iterative process of negotiating requirements customers should validate the performance of the prototype against the "wish list" of requirements. Design team may want to release reports on their work -- using the ***Reports Room*** --, communicating and informing about their progress, between releasing the actual prototypes. This enables them to get feedback on partial results.

During the prototype evaluation phase, the system enables participants to "test" the product on their workstations. They can use the ***"Worth Proceeding?" Room***. At the same time they are

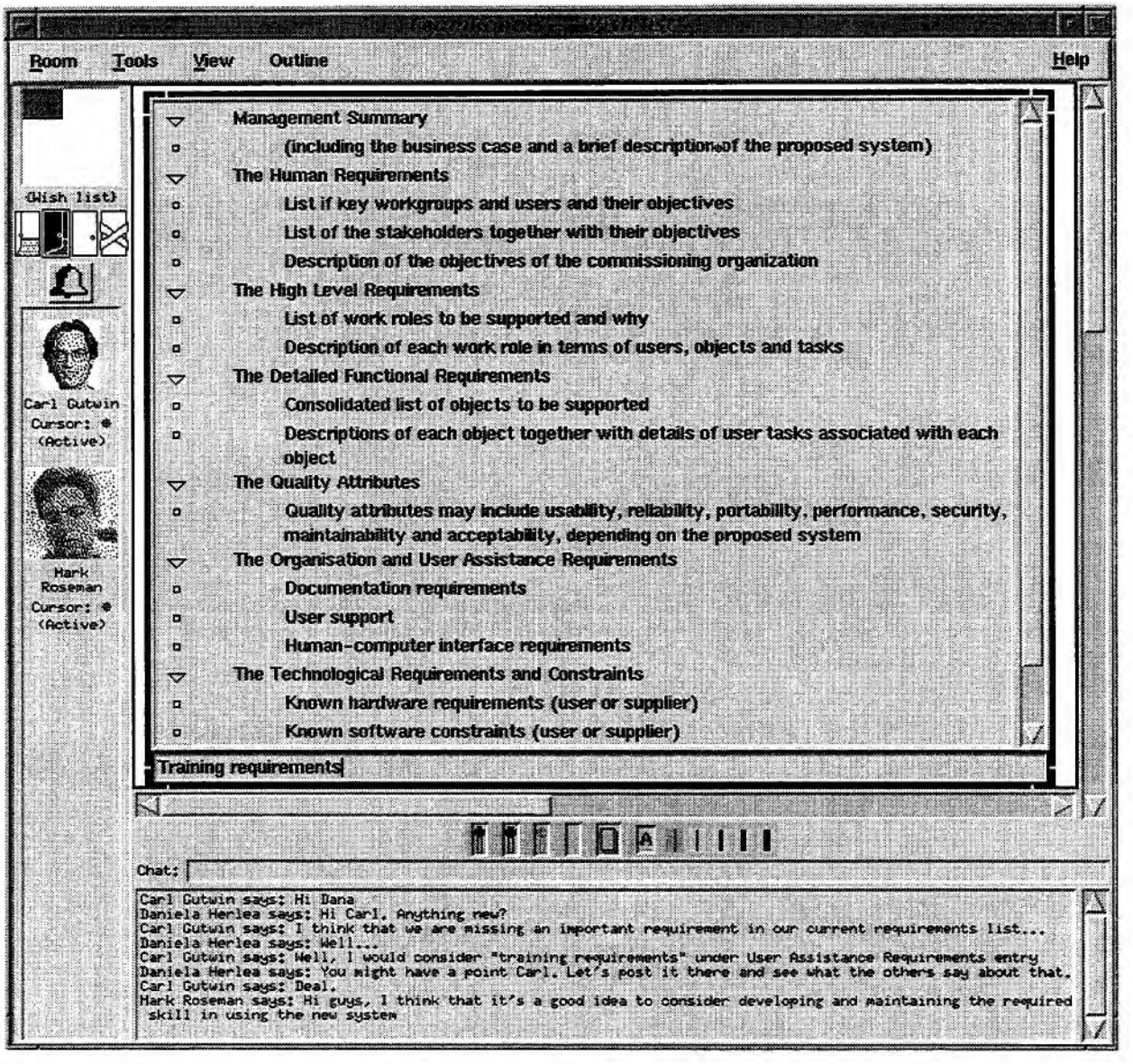


Figure 5. User requirements: list of contents

able to collaborate, to exchange opinions. The image tool may be used to bring graphical images into the meeting space and discuss around them. A design team representative may run the application on his own workstation and send screen captures to all participants, as still images. The customers of the system may see if the system meets the functional or non-functional requirements.

Users of the system may use a Timbuktu Pro window close to their workspace. Timbuktu Pro is a powerful tool that enables collaboration and remote access over the Internet. If the participants in requirements negotiation are connected to a Timbuktu session they can remotely control computers at a distance. Look@Me is a Timbuktu applet which would give the problem "owners" the ability to view others' screens in real time. They would be able to watch the whole activity of testing the prototype from their Timbuktu Pro browser.

The system supports collaboration within the requirements negotiation group and the individual work as well. Any user can create his own room. Users create ***Personal Rooms*** for themselves, as they do with WWW home pages. These rooms may serve as a personal work space, where the user can use different tools and save his artifacts or leave notes for the group members.

In the same manner users may leave notes for the other members in the group, using the ***"Read me" Room***. They may work part-time for this project, working on their functional roles in the organization at the same time. Based on their schedule, they can meet and work in the working space, at different times and leaving notes for the others.

GroupWeb browser is an example of including external information into the workspace. Web browsers became the standard way that people search for and display items of interests. Like normal Web browsers, GroupWeb fetches and displays HTML pages. In TeamRooms, the web browser becomes a shared visual workspace, an excellent presentation tool. It supports groupware features such as telepointers and page synchronization. Users connected to this rooms look at the same web page. Any page change is visible for all of them. *Telepointers* are used as a way to transmit and display gestures. It is useful to discuss information with others in real-time. Usually, HTML pages require scrolling as they rarely fit completely within a window. However, the *multi-user scrollbar* facility -- existing in GroupKit's Web browser -- is not implemented in TeamRooms. Users are not aware of others' viewport unless they communicate through the chat tool.

Other considerations

Exchanging files in real time is a nice feature of TeamRooms. Members of the team may retrieve files in a ftp like manner. A File viewer facility – non existing yet -- would be the best way to browse the content of the retrieved file, allowing participants to work together on the text. This might be a nice feature to add to the system.

However, Timbuktu Pro offers the possibility to exchange files in real-time and edit "live" the same document and discuss around it. This way the participants in the process can work on a "virtual" or "shared" desktop through the power of remote control. Validating the prototypes helps in understanding the requirements. New requirements may be added or existing ones may be changed.

So far, any user of the system may enter any room, changing any applet in the room. This is a weak feature of the system. A well defined set of rights over the rooms in the system would constrain the actions of the team members in the process. For example, the facilitator should have the right to enter any room, but not to modify the work of the users. The customers should have the right to work on the requirements documents, together with the analyst, but not to change the final specification of requirements. This is the analyst's job.

Another limitation of TeamRooms is the lack of an audio link. Microphones attached to computers would provide a real-time audio collaboration. As an example, the analyst would do a better job if interviews were held with participants in the process.

The still images of the users displayed in each room may be replaced with video images. Video conferencing would be very helpful in assisting the analyst in the process of requirements elicitation. Sometimes the gestures and the mood of the participants in the process provide useful information about their involvement in the process of requirements negotiation.

SUMMARY

Software requirements elicitation is the process where the customers' needs in a software project are identified. This process is regarded as one of the most important parts of building a software system because during this stage it is decided precisely what will be built. However, requirements gathering needs close interaction between developers and end-users of the system. If developers and end-users are in different organizations or different cities, meetings can be costly, inconvenient and infrequent. This leads to problems of communication, which can greatly impact the quality of the elicited requirements. Well known requirements engineering methodologies are presented in this paper and the degree of the user involvement in the process of requirements negotiation is discussed.

TeamRooms system provides the electronic equivalent of a team room for groups that are either co-located or distributed. It provides both generic communication facilities as well as very specific tools in each room. TeamRooms is a system whose rooms may be quickly tailored to meet the needs of the group. The system supports a rich persistence mechanism, where any number of versions of one applet's state could be saved and retrieved at a later time. Versions of the applet's contents are automatically saved into a repository when the last user in a room leaves. Selecting the Versions option from the applet's menu, the users may browse the entire history of a groupware document. Shared documents exist beyond the meeting session. This way, team members may work on tasks either individually or in a group, their co-workers being able to find their artifacts for later use or information.

AKNOWLEDGEMENTS

I would like to thank Mildred Shaw for her careful reading of the paper and helpful comments. I am grateful to Brain Gaines for early guidance on this work. Also, thanks go to Rob Kremer for his perceptive critiques that improved the paper and to Mark Roseman who helped me in investigating TeamRooms system.

REFERENCES

Bostrom, R.P.(1989). ***Successful Application of Communication Techniques to Improve the Systems Development Process***, *Information & management (16:5), pp. 279-295*

Brooks, F.P. (1987). ***Essence and Accidents of Software Engineering***, *IEEE Computer, Vol. April 1987, pp. 10-19*

Carmel, E., Whitaker, R.D. and George, F.J.(1993). ***PD and Joint Application Design: a transatlantic comparison***, *Communications of ACM, June/Vol. 34 No. 4, p. 40-46*

Checkland, P. (1981). ***Systems Thinking, Systems Practice***, *Chichester, U.K. Wiley*

Christel, M.G., Wood, D.P. and Stevens, S.M.(1993). ***AMORE: The Advanced Multimedia Organizer Requirements Elicitation***, *Tech. Report, CMU/SEI-93-TR-12, ESC-TR-93-189, June*

Dobson, J.E., Blyth, A.C., Chudge (1994). ***The ORDIT approach to organizational requirements,*** *Requirements Engineering: Social and Technical Issues, Academic Press, San Diego*

Gasson, S. (1995). ***User involvement in decision-making in information systems development.*** *Proceedings of IRIS 18, Gjern, Denmark, Gothenburg Studies in Informatics, Report 7, pp. 201-17.*

Goguen, J. (1992). ***Requirements Engineering: Reconciliation of Technical and Social Issues,*** *tech. Report, centre of requirements Lab., Cambridge, U.K.*

Jarke, M., Pohl, K., Domges, R., Jacobs, S., Nissen, H.W. (1995). ***Requirements Information Management: The Nature Approach,*** *Techniques et Sciences Informatiques*

Keil, M. and Carmel, E. (1995). ***Customer-Developer Links in Software Development,*** *Communications of ACM, May Vol. 38 No. 5, p. 33-44*

Koltzblatt, K and Beyer, H.R.(1995). ***Requirements Gathering: The Human Factor,*** *Communications of ACM, Vol. 38 No. 5, p. 30-32*

Macaulay, L.A. (1996). ***Requirements Engineering,*** *Springer, pp. 83-112*

Markus, M. and Bjorn-Anderson, N. (1987). ***Power over Users: Its Exercise by System Professionals.*** *Communications of the ACM 30*

McKeen, J.D., Guimaraes, T. and Wetherbe, C.J.(1994). ***The Relationship Between User Participation and User Satisfaction: An investigation of Four Contingency Factors,*** *User Participation and Satisfaction, MIS Quarterly/Dec. , pp. 427-447*

Mumford, E., and Hensall, D. (1979). ***A Participative Approach to Computer Systems Design,*** *Associated Business Press, London*

Potts, C., Takahashi, K., Anton, A.I.(1994). ***Inquiry-Based Requirements Analysis,*** *IEEE, March , pp. 21-40*

Roseman, M. and Greenberg, S (1995). ***TeamRooms: Network Places for Collaboration***

Roseman, M. and Greenberg, S. (1996, in press). ***Building Real Time Groupware with GroupKit, a Groupware Toolkit,*** *ACM TOCHI*

Siddiqi, J. (1994). ***Challenging Universal Truth of Requirements Engineering,*** *IEEE, March , pp. 18*