An evolutionary approach to groupware implementation: the context of requirements engineering in the socio-technical frame

Dulce Pumareja, Klaas Sikkel
Department of Computer Science, University of Twente
P.O. Box 217 7500 AE Enschede, The Netherlands
{pumareja, sikkel}@cs.utwente.nl

Abstract

Organizational implementation of new technology brings about design challenges as issues of requirements mismatch and evolving requirements emerge. For groupware systems, these issues are much more problematic as processes surrounding the performance of collaborative work are dynamic, unpredictable, hard to capture and difficult to specify in advance. Requirements for groupware systems are best appreciated by securing a real world reference for its actual use in the context of implementation. Addressing these design issues therefore necessitates an evolutionary approach. To systematically carry out this approach, we draw insights from socio-technical systems design regarding system implementation processes, to contextualize as well as rationalize the role of requirements engineering (RE) in scaffolding the organizational implementation of groupware.

1. Introduction

Groupware is a generic term used to refer to a class of information communication technologies (ICT) that provide electronic support to groups of people engaged in collaborative work. Compared with other business application systems, requirements for groupware applications are difficult to derive and specify. While the economics of software engineering indicate that it is best to properly specify the requirements of a system before construction, when corrections and revisions are less costly, for systems in which needs cannot be specified in advance such as groupware, this may not be applicable. Group processes during the performance of collaborative work are dynamic, unpredictable, and are rather hard to capture, formalize and specify prior to development. Within the CSCW (Computer-Supported Cooperative Work) research community, it is a rather widely accepted notion that the way to properly appreciate a groupware system's requirements is to secure a real-world reference for its actual use in the work context (Bannon & Hughes, 1993).

Mismatch between system functionality and real-world user needs in the context of organizational implementation is an acknowledged problem in software engineering. This necessitates software maintenance efforts. Software systems are built based on a partial knowledge of the real user requirements and their operating environment. As these requirements are also known to change, developers have sometimes the tendency to negotiate a 'freeze' in requirements. With groupware systems, this mismatch may even be worse as the requirements process is poorly informed about the real application domain of groupware which are least understood through formal and structured means. This has been the case in the early development efforts of commercial groupware wherein these systems were built as extensions of personal

systems through the addition cooperative components and neutral information exchanges mechanisms (Sommerville and Rodden, 1994).

Any system implementation within a social context brings about design challenges. One of these design challenges, which exists for groupware systems, is that the use of the system cannot be anticipated or predicted in advance (Norman, 1993). Ciborra (1996) describes groupware as a technology that "drifts" when put into use. This is based on observations of groupware implementation cases revealing a significant departure from the original intended plan of use to the way the system is currently being used by the organization. In these instances of positive adoption and continuing implementation, system use evolves, mostly as a result of a learning process. As users get to know more about the system, they find new ways of using the system to the extent of developing new applications in the case of customizable groupware (Orlikowski, 1996). However, not all groupware implementations have positive outcomes (Grudin, 1988). In instances of rejection or non-use, or sub-optimal use, efforts arise within organizations to stimulate the use of the system or to find other purposes for which the system can be used.

In both implementation scenarios – positive or negative adaptation, a certain design response is needed, either in the form of system adaptation to evolving requirements or by creating and identifying other needs that the system can satisfy. Coming up with an appropriate design response to system-related implementation issues necessitate an evolutionary orientation. Such evolutionary approach towards groupware implementation finds merit in several accounts of case history documenting groupware implementation in organizations (Orlikowski, 1996; Karsten & Jones, 1998; Pipek & Wulf, 1998).

In order to a get a grip on the nature of the design response needed, an understanding of the nature of requirements which serve as parameters for design is crucial. Requirements, in software engineering terms are known to be problematic due to their changing and evolving nature. Problems of requirements elicitation, conflicting demands necessitating a win-win negotiation and making compromises impact the final state of the requirements of the system undergoing development. In the domain of system development, such problem contexts are well known. However, little is known about the properties of requirements in the system's use domain. Most especially in the case of groupware, the nature of its requirements takes on a more subtle and complex nature due to a plethora of factors that come into play. The limitation of most software engineering approaches when it comes to design issues in system implementation is that these issues are lumped together as software maintenance efforts – a reactive measure, which really do not describe how to monitor opportunities for change and improvement.

To this effect, a deeper analysis of requirements from which software maintenance efforts draw their context is necessary. Therefore, the following questions motivate our research: (1) how do requirements change in an evolutionary process? (2) how should a system adapt to evolving requirements?

To address these questions, we draw upon socio-technical systems design framework in investigating the requirements of a system in use and in deriving the appropriate design response in an evolutionary implementation. This framework is adopted in

careful consideration of non-technical issues that are oftentimes the cause of system implementation failure (Grudin, 1988; Ruel, 2001; Laudon & Laudon, 2001).

2. Socio-Technical Systems Design Framework

The socio-technical systems design framework is drawn from the theoretical contributions of socio-technical systems theory and its corresponding framework for systems design.

2.1 Socio-Technical Systems Theory

Socio-technical systems (STS for brevity) theory is a framework for research and action on understanding and supporting work and organizations. STS was conceived out of the post-war action research efforts by the Tavistock Institute on several coal mining and industrial organizations who were in the process of modernization through the implementation of new machines and technology (Mumford, 1987).

The main theoretical input of STS is drawn from systems theory (von Bertalanffy, 1950) together with the notion of open systems. One of the central ideas of STS is the concept of an organization as a socio-technical system. That is, an organization as a whole has both a human component representing the social system and a non-human technical system component. These two relate to each other (Trist, 1981). The theory calls for the joint optimization of the social and technical systems. In STS, the social and the technical systems are regarded as correlative to each other, especially in the context of work as co-producers of outcome. Yet, each system is distinct from one another as each posses a different set of characteristics. The interaction between the two systems has been described as a 'coupling of dissimilars' that can only be jointly optimized. Citing Trist (1981), "The distinctive characteristics of each must be respected or else their contradictions will intrude and their complementarities will remain unrealized."

The basic unit of research in STS is the primary work system, which are sub-systems within an organization that carry out a set of activities in an identifiable boundary such as that of a department or a functional unit. This collection of people, machine and activities are unified by a recognized purpose. STS has been a strong proponent of autonomy and the formation of autonomous work groups has been one of the key interventions carried out in early STS research, which were seen as more efficient work units.

2.1.1 Main provisions and propositions

Socio-technical systems theory seeks to describe the dynamics of new technology implementation within organizations and proposes a dualistic action agenda for managing organizational change. It puts forward the following theoretical propositions:

- An organization is an open socio-technical system comprising of human and technical components that correlate with one another towards the achievement of a common organizational purpose;
- Successful innovation and change process is a product of a joint optimization of these two components by way of mutual adaptation between human users and the system

- Group autonomy is a necessary condition and a success factor in system acceptance, use and adoption.

Further, the theory elaborates the following points (Bikson & Eveland, 1996):

- Changes in the social and technical systems as a result of technology implementation cannot be predicted in advance. As the social and technical systems reciprocally influence each other, the question of what works for each cannot be answered independently;
- Decisions and actions in the course of new technology implementation will have an influence on the outcomes;
- Consequences of technological innovation evolve over time given reciprocal effects and mutual adaptations between social and technical systems.

As an applied theory, the goal of STS is to find out the optimal condition described as the "fit" between the needs of system users and the properties and functionalities offered by the system. In return, the theory prescribes several guidelines and principles in applying STS theory to derive the optimal action response for managing technology implementation. These efforts are usually carried out through action research, which is the accompanying methodology for STS.

2.2 Socio-Technical Systems Design

Socio-technical systems design is an approach to systems design that incorporates the design principles derived from STS studies (Mumford, 1987). The aim is to give equal weight to social and technical issues when new work systems are being designed (Mumford, 2000). It takes into account that different individuals and groups have their own needs, interests and values that must be met, if employees are to accept major change willingly and enthusiastically. Therefore, the setting and specification of human objectives and needs such as job satisfaction and motivation are just as important as the identification business and technical needs in the requirements process of system design.

In the realm of system implementation, socio-technical systems design operationalizes the STS action research agenda of matching social and technical components in terms of a socio-technical systems 'fit' between four interrelated variables: technology, task, people and organization. Mumford (1995) explains:

> "The introduction of a new technical system is likely to disturb each of these variables. A new level of technology will bring with it a new man-machine relationship incorporating both opportunities and constraints. Because tasks are influenced by technology, the task structure of functions or departments using the system will be altered. New tasks mean that new demands are made of people and this will affect job satisfaction positively or negatively depending on whether the new situation meets their values of what is desirable work. Finally, the technology, tasks and people variables interact with an internal environment which provides a structural context for the achievement of the organization's objectives and this interaction may start the looping process again by making demands of technology"

Socio-technical systems design perceives effective system implementation as an evolutionary process of mutual adaptation and integration between these components. Adaptation is described as the movement from one kind of technical and organizational structure and state to another and integration are the necessary actions to ensure that the new situation attains stability and equilibrium (Mumford, 1995).

2.2 STS in Groupware implementation

Bikson & Eveland (1996) sought to explain the dynamics and outcomes of groupware implementation in organizations within a socio-technical systems framework. In their case study conducted at the World Bank on the implementation of a decision-support system called *Group Systems*, the STS theory was validated. The major propositions of STS theory such as mutual adaptation between users and system as well as user-participated design process were observed to be present and 'working out' in practice. Such attention to socio-technical dynamics provided an insightful appreciation of the organizational pay-offs of groupware applications. Bikson and Eveland (1996) identifies the following as key enabling factors to a successful system implementation: (1) attention to process and (2) embracing the idea of continuous implementation through interaction between the tools and the social arrangements that embed them at the point of the task interface.

3. Groupware

Groupware systems are ICT applications that provide electronic support for collaborative work. Originally, groupware systems were built as extensions of personal computing and regarded as variant of corporate computing imbued with the goal of providing support for groups and group work. The context of groupware with regards to other computer applications can be further understood in terms of Grudin (1991; 2001) classification, relating systems development efforts to that of corresponding social context of use (cf: Fig. 1).

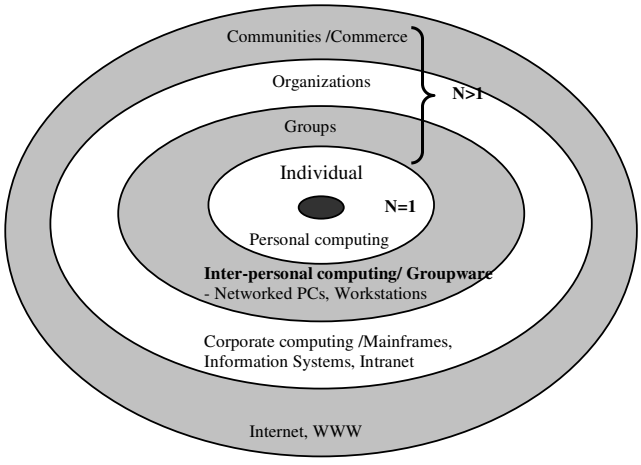

Fig 1. Human structures and systems development efforts classification (Grudin, 1991; Grudin & Poltrock, 2001)

In this classification, there are four levels of societal structures in which systems can be built for: the individual, groups, organizations and communities. Individuals are singular users of systems and are supported by way of stand-alone and personal productivity computer applications. Groups are collection more than one individual that has interaction with one another. Technological support for this kind of users is

provided through network applications, workstations and groupware. Organizations are formal social entities made up of people and groups that relate with one another towards the attainment of a certain goal (Daft, 1998). Organizations were largely and originally the object of most systems development projects and applications consisting of mainframes, corporate information systems, intranets among others (Grudin, 2001). Communities are voluntary aggregation of people and groups, usually geographically distributed, who share a common interest but with no rigid structure imposed. The Internet and its suite of applications such as the WWW are facilitative technologies that provide support for communities.

It can be surmised from this classification that the context of groupware systems begins when the social milieu of use shifts from the individual towards groups and other larger societal structures such as organizations and communities where people and groups come together. On the other hand, moving further through the larger societal structures make it difficult to establish distinctions between interpersonal and organizational computing the extent to which they can be classified as groupware. Group-orientation according to Greenberg (1991) is key factor from which to distinguish groupware systems from other kind of systems. Isolated personal computers and mainframe systems or data warehouse in this sense are excluded from the concept of a groupware. As organizations and communities are also composed of groups and that systems built for these human structures also require groups of people to work together, overlaps exist. In-between are many-kind of group-oriented systems in which there is no rigid dividing line (Bock, 1991; Ellis, Gibbs & Rein, 1991).

Defining what groupware is has been a subject of debate since the 1990s and various definitions of groupware abound in CSCW literature. Each varies accordingly to the bias the author attributes to the technological features of the system or the process in which the system is used (see esp. Grudin, 1994; Greenberg, 1991; Bullen & Bennet, 1990; Johansen, 1988). However, the distinction between groupware and other systems is getting more and more blurred because systems being developed nowadays regardless of the domain have groupware aspects and features.

In this manner, we will define groupware as follows:

> *Groupware is any group-oriented ICT application that provides support for collaborative work.*

By group-orientation, we refer to the scope of the system in terms of its target set of users who can be classified as a group or groups, and its use dimension wherein system use invokes a collaborative work process. A group is a social aggregation of individuals that has the awareness of its existence as a group, circumscribed by its member's mutual awareness of each other and interdependence towards a shared purpose or work.

3.1 Groupware as socio-technical systems

One way of gaining a better understanding of groupware systems is to assume a socio-technical frame of thinking – that is by considering both the technical and the social characteristics of the system.

3.1.1. As a technical system: functional properties

Viewed as a technical system, a groupware application enables interpersonal cooperative processes across time and space. It provides technical functionality for communication, information sharing and coordination. (Grudin, 2001; McGrath & Hollingshead, 1994). In this way, corporate business information systems and other organization-wide applications also contain these features and therefore can be classified as groupware. An integrated customer relationship management (CRM) system for example contains functional components that enable office and field sales representatives to communicate, coordinate and share information (Siebel, 2002). In consideration of these, it is therefore useful to establish distinctions between two main types of groupware: ***special-purpose*** and ***general-purpose*** groupware.

Special-purpose groupware fulfills a specific operational role in the organization. The functional components for supporting cooperation correspond to organizational processes in which the system operates. Another example is a course registration system wherein students, teachers and the registrar can coordinate their activities cooperatively in the context of registration process.

In contrast, ***general-purpose groupware*** is independent of the domain in which the business or organization operates. The functional components enabling collaboration are applicable across different organizations and contexts. Examples of these are email and video-conferencing systems. Commercial groupware such as Lotus Notes, Group Systems, WebEx belong to this kind of groupware systems.(Orlikowski & Hofman, 1997; Compton, 2001). Excluded however are workflow systems from this classification, as general-purpose groupware do not prescribe how work is to be done.

3.1.2. As a social system: use dimension

Viewed from a sociological point of view, the behavioral processes surrounding the use of a system contributes to the 'groupware-ness' of that system. A general-purpose cooperative system such as email is a groupware in the technical sense, however it may be used for mass marketing purposes in which support for collaborative activities is absent. Ellis, Gibbs & Rein (1991) draws on the use aspect of a system as a determining factor in contributing to the 'groupware-ness' of a system. According to them, for a system to be considered as groupware, a common task or goal must be present. However, such criterion is not sufficient. We will cite an example: a collection of secretaries who share similar functions in a faculty may use the same general-purpose cooperative system but are not aware of each other. In this way, it can be said that the ordinary presence of a common task or goal in the user environment does not sufficiently warrant the use dimension of a system as groupware. Workgroup *awareness*, that is, an acknowledged consciousness of being part of a work group that performs shared tasks and responsibilities in the pursuit of a common goal, as an added element in the context of computer-supported work redefines the support system as groupware.

3.1.3. Groupware contexts: socio-technical states

Orientation to the socio-technical frame of thinking in which the functional properties and the use aspect of a system are taken both into consideration, serve as tool for thought upon which systems can be appreciated of their groupware-worthiness.

Groupware systems are dualistic systems in which a holistic approach is necessary for reflection, as one aspect does not completely provide a meaningful representation.

In this manner, rather than treating groupware aspects in separation, it is more meaningful to refer to groupware contexts, which are characterizations of different socio-technical states in which a system given its properties is used. These states, following the notions put forward in section 3.1.1 and 3.1.2 are classified along the socio-technical dimensions of (a) degree of work group awareness with respect to system-use and (b) functionality specialization. The degree of work group awareness can be described either as low or high. Low work group awareness implies that the user group has little or no consciousness of being in work group, but are at the same time tasked with the performance and execution of shared work and goal. On the other hand, a high degree of work group awareness is present when there is task interdependence among system users and that they internalize being part of a work group which is externally recognized as a functional unit. In terms of the extent of generality or specialization of the functional components, groupware systems can be classified either as general purpose or special purpose groupware (see section 3.1.1).

These notions put altogether to form a framework for describing the context of groupware in terms of different socio-technical states is shown in Figure 2.

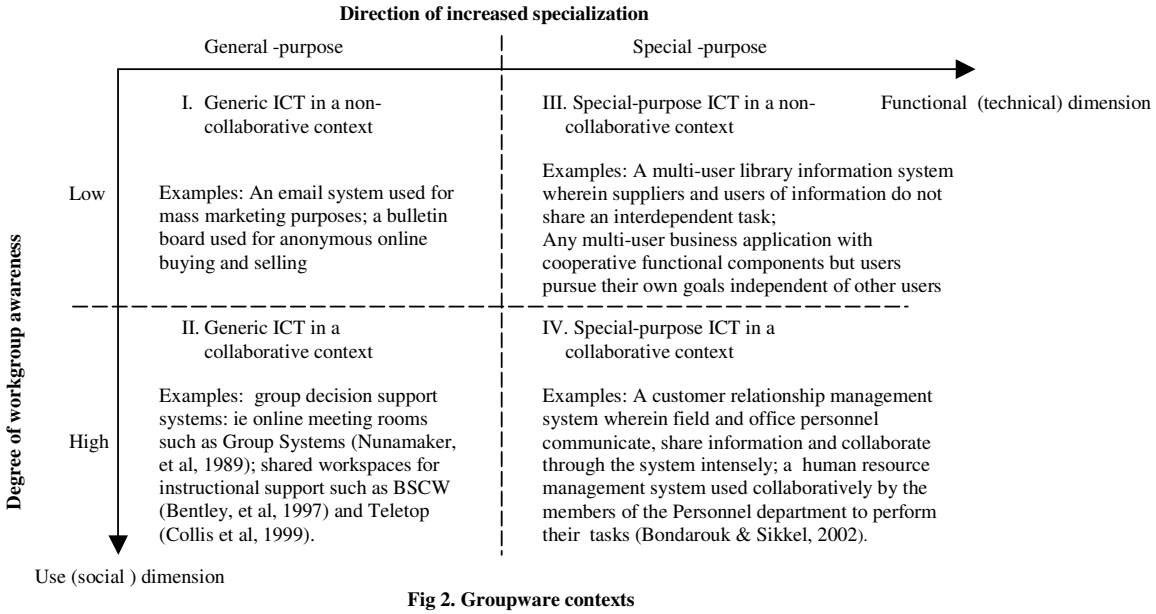

Fig 2. Groupware contexts

According to this classification, groupware contexts can be described in terms of four socio-technical states:

- Generic ICT in a non-collaborative context (Quadrant I)

  This socio-technical state is characterized by a general-purpose groupware application, such as email or a video conferencing system but the use context is marked by a low-level of workgroup awareness. Users do not strictly form a group or collaboration amongst them is little or non-existent.

- Generic ICT in a collaborative context (Quadrant II)

  This is a state wherein a general-purpose groupware such as the examples mentioned in the chart, are used intensively in workgroup-aware user group who share tasks, responsibilities and are oriented to a common goal.

- Special-purpose ICT in a non-collaborative context (Quadrant III)

  This state is populated by a special-purpose groupware, which could be any distributed or multi-user business application that provide functionality for communication, information sharing and coordination but the user group does not share responsibilities and tasks; users pursue their own goals independent of other users.

- Special-purpose ICT in a collaborative context (Quadrant IV)

  In this socio-technical context, the groupware system is a special-purpose groupware with an intense use dimension: users collaborate closely with one another through the system.

Groupware implementation in organizations can be described and analyzed in terms of these socio-technical states. By referring instead to contexts, we say that groupware systems are not defined only on the basis of their functional properties but also by the properties of the social environment in which they operate and are used. Groupware is both people and tools that people use. A groupware system is best appreciated *in situ*.

The broad definition of groupware as any group-oriented ICT that provides support for collaborative work finds support in terms of Greif (1988) who mentioned, that in the long run, the differentiation of a segment of the software product market in terms of groupware might not make much sense at all; all software will have the required features to support group use when appropriate.

3.2. Groupware Applications

There exists a plethora of software applications and products classifiable as groupware on the basis of their functional properties. To cite examples of groupware applications and products, it would be useful to refer to a certain classification taxonomy of which most instances of groupware in literature are ascribed to. These taxonomies are derived from a time-space dimension classification, application domain or work content and workflow, among others (Ellis, Gibbs & Rein, 1991; Put, 1996; Coleman, 1997; Johansen, 1988).

As per definition, instances of groupware applications and products do not exclude large-scale enterprise-wide information systems that have functionalities supportive of a cooperative group work process. To make the distinctions clearer among examples, there are two levels of classifications used in the listing below. The first level deals with the **specialization type** of the functional properties of an application as described in section 3.1.3. The second level is a taxonomical listing of the applications and products according to their application class or domain.

In terms of specialization type, there are two core categories of groupware: *general-purpose* and *special-purpose*. General-purpose groupware systems are those classes of

groupware applications and products that support the social interactions that facilitate the performance of a business process regardless of its content or domain. Examples of these groupware applications such as these are shown in Table 1.

Table 1
General-purpose groupware applications and products

Application Class	**Electronic communication and messaging systems**
Applications & Products	Applications for exchanging messages, regardless of time differences. *Sample applications:* Email | Bulletin Boards | Discussion Boards *Sample products:* Microsoft Outlook/ Outlook Express; Lotus Mail; Eudora Mail; Pegasus Mail O'Reilly Web-Board; Lotus Notes Discussion Database; Netscape Collabra
Application Class	**Group calendaring, scheduling and coordination systems**
Applications & Products	Applications for calendaring, agenda planning, meeting and resource allocation. *Sample products:* Microsoft Outlook; Lotus Notes Organizer; The Coordinator (Winograd, 199x; Robinson, 199x)
Application Class	**Decision support and electronic conferencing and meeting systems**
Applications & Products	Applications for supporting group decision making and for facilitating meetings despite difference in time and locations. *Sample products:* Group Systems (Nunamaker, et al, 1989);Video Conferencing Systems; Microsoft Net Meeting; WebEx; Lotus Sametime
Application Class	**Collaborative authoring and design systems**
Applications & Products	Applications for group editing, shared screen work, group document or image management and document database. *Sample applications:* Multi-user editing systems | content management systems | Hyper-text editing systems Sample Products: Group Writer (University of Calgary); GROVE; Xerox PARC Note Cards; Interwoven TeamSite 4.5
Application Class	**Collaboration and information sharing systems | Shared workspaces**
Applications & Products	Applications for collaboration, usually composed of an application suite, with shared work spaces and accessible or are ported through the web *Sample applications:* Knowledge management systems | Online instructional and classroom l support systems *Sample products:* Basic Support for Cooperative Work (Bentley et al, 1997); Teletop (Collis et al, 1999);
Application Class	**Extensible composite collaboration systems**
Applications & Products	These are Commercial-Off-The-Shelf (COTS) groupware applications, which feature an all-in-one concept by incorporating several groupware applications in one, i.e. email, discussion, video conferencing combined in one composite system. Further, some of these kinds of groupware are customizable, extendable and even serve as an development environment for constructing new groupware applications. *Sample products:* Lotus Notes; Microsoft Exchange / Exchange Server ; Novell Netware

Adapted from Ellis, Gibbs & Rein (1991); Put (1996); Coleman (1997)

It must be noted that this classification does not imply mutual exclusivity among application classes, as overlaps exist. A particular product can support multiple functionalities, especially in the case of composite systems wherein different groupware services such as email, video conferencing, and shared workspaces among others are bundled in one application. In the examples given, a popular product of this kind is Lotus Notes.

On the other hand, special-purpose groupware systems are those business information systems whose content and processes correspond to a particular business domain and business process. In other words, these kinds of systems facilitate the performance and execution of a mainstream business process within the organization. They can be contrasted with groupware in terms of the scale of its targeted use. Usually these systems are intended for an organization-wide use, but since organizations are made up of groups, these special purpose applications in return have a module or a functionality that supports a group process. Hence, these kinds of systems are also groupware systems to such extent. Examples of these kinds of applications are shown in Table 2.

Table 2
Special-Purpose groupware applications and products

Application Class	**Workflow systems**
Applications & Products	*Sample products:* IBM Flowmark; Action Tech Action Workflow;
Application Class	**Enterprise applications / Business information systems module**
Applications & Products	These are specialized, composite systems that focus on automating and facilitating mainstream business processes or functions of an enterprise or organization with modules or sub-systems that are for use by a group of people. *Sample applications:* Lead Management Module of a Customer Relationship Management System for Financial Institutions; Human Resources Information System of a Hospital Information System *Sample products:* People Soft CRM Financials, Beaufort Systems (Bondarouk & Sikkel, 2002); Siebel Systems Employee Management Relationship System

The listing we have provided here is just a sample. It is highly likely that there are more examples of groupware applications and prototypes that are in existence or are currently under development aside from than the ones provided in this list. Most especially in the case of enterprise applications or business information systems modules, the list of special-purpose groupware applications can be endless.

For more examples and other groupware taxonomies, we refer the reader to Ellis, Gibbs & Rein (1991) and Coleman (1997).

3.3 Groupware: a conceptual map

To visualize groupware in terms of interrelated applications and concept, we provide herewith a conceptual map as framework for understanding groupware systems. This is shown in Figure 3.

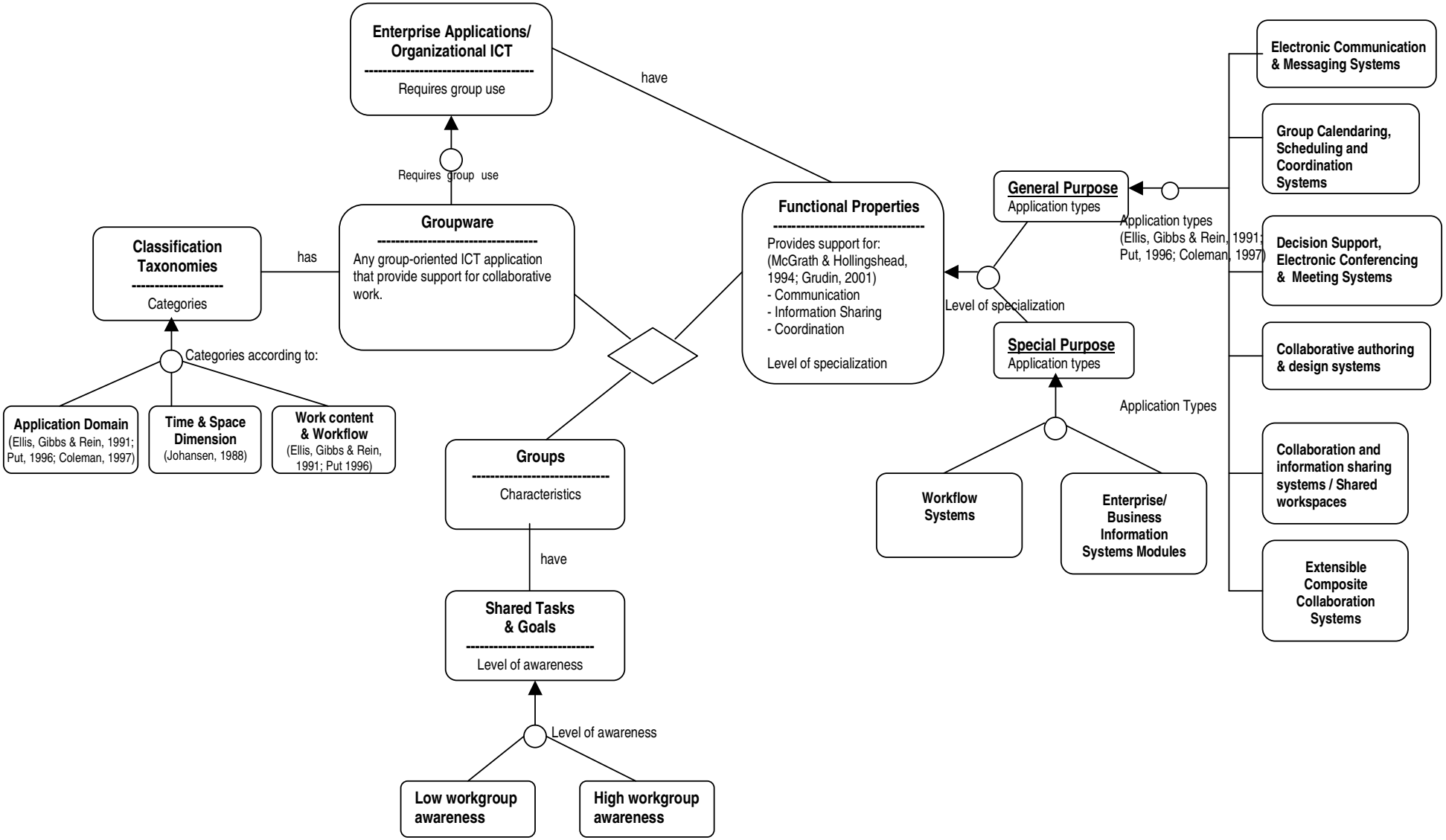

Fig. 3: Groupware concept map

4. Requirements

A requirement is a multi-layered concept in the context of software engineering. It can refer to a fuzzy idea that manifests itself in certain ambiguous ways as a need and becomes articulated as a problem amenable to a software solution. It can be further refined such that it already provides a sketch to a solution and written in paper to comprise what is called a requirements specification document.

As a starting point, requirements are desirable properties of business and system (Wieringa, 2001). These properties are expressed as business needs and goals as well as software specifications (Alexander, 2002; Wieringa, 2001) that motivate software design and development. Business needs correspond to business requirements, of which user requirements derive their context. Software requirements are specifications that satisfy the business solution that addresses the need or goal. They are descriptions of how the system should behave or of a system property or attribute as well as constraints on the development process (Sommerville & Sawyer, 1997). To distinguish between software requirements and design, requirements should only specify what a system is supposed to do without specifying how this should be done.

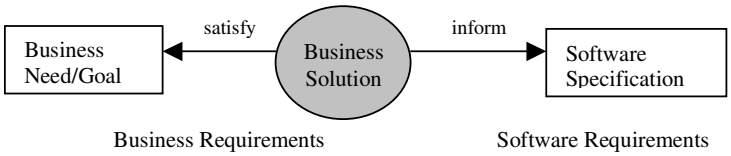

Fig. 4: Business requirements and software requirements

Software requirements are generally classified into types: functional requirements and non-functional requirements. Functional requirements describe what a system is supposed to do in terms of behavior and services that it has to deliver. For example, a course registration system should enable a student to register for a course. Non-functional requirements, also called quality requirements specify how well a system performs its intended functions (Lauesen, 2000). Other types of requirements are constraints, which are also desirable properties that are enforced by the environment that restricts design freedom. For example, the course registration system has to be implemented only on Windows platform or that the organization decided to use Lotus Notes as the platform for developing a knowledge management system.

4.1 Requirements Engineering

The process of discovering, analyzing, documenting and maintaining requirements for the purposes of building a system is called requirements engineering (Sommerville & Sawyer, 1997). However, it is with some reservations that this process is referred to as engineering, as it falls short of the actual enactment of a design process. Wieringa (2002) refers to it as the problem analysis of a design process and it requires description, application of theories, analysis and verifications of results.

Requirements engineering is an important phase in the systems development lifecycle. Every software lifecycle model incorporates requirements engineering or a counterpart process thereof, as a front-end activity with several cycles of iteration in

the design and development of software systems (Pressman & Ince, 2000; Pfleeger, 1999, see ch. 2 ; Rational, 2001).

As a process, RE is composed of a set of tasks that include (Wieringa, 2002):

- *Requirements elicitation*: the task of gathering requirements from several sources and stakeholders;
- *Requirements analysis*: modeling and analyzing requirements such that a consistent set of requirements is achieved;
- *Requirements specification*: describing the requirements and writing them down to a certain formal extent that is still understandable for the stakeholders;
- *Requirements validation*: communicating the requirements with the customer so as to verify for correctness and acceptance;
- *Requirements maintenance*: ensuring that the requirements retain their consistency and monitoring

From among these RE tasks, requirements elicitation is considered to be a problematic and difficult task. Elicitation is also a misnomer, in the sense it is assumed that the requirements are in the minds of people that can be easily extracted. Requirements, on the other hand are emergent, in the sense that are formulated as process of reflection and interaction between users and requirements analysts (Goguen, 1993). On the other hand, communication problems hinder elicitation processes, as users find it hard to articulate their needs. It is not that they do not know what their work is about, it is just that some of these experiences are *tacit* and are hard to verbalize.

Another issue in requirements engineering is the problem of requirements change and evolution. Requirements continually change, as the software project unfolds. That is, clients keep on changing their mind even before and while the system is being developed. In the instance when for instance a system has been built, they sometimes realize that it does not fulfill their expectations, even though the system satisfies the written requirements (Lauesen, 2000).

4.2 Human, Social and Organizational Factors

There has been a growing support for the incorporation of contextual aspects such human, social and organizational factors in the RE process (Sommerville et al, 1994; Norman, 1991; 1993; Land, 1987). Much of the structured and conventional approaches to RE has been criticized for its adherence to formality which is believed to be one of the reasons why systems are not accepted or are implementation failures (Goguen, 1994; Hughes et al, 1994). Large development projects fail because of inadequate requirements or that the processes of identifying requirements are not satisfactory (Gause & Weinberg, 1989; Hughes et al, 1994).

The need to take these factors in consideration in the RE process becomes more pronounced when it comes to the design of cooperative systems such as groupware. As a result of considerable attention paid to these issues, research in RE had progressed towards the integration of a multidisciplinary perspective in the process. Concepts, theories and principles especially from psychology, sociology and anthropology are finding their way in the RE research agenda (Nusebeih & Easterbrook, 2000).

4.3 Socially-oriented Requirements Elicitation Techniques

Goguen (1994) argues that requirements engineering is a highly social and contextual process of which non-technical approaches are needed to gain a deeper understanding and insight of what the real world needs are.

Alternative requirements elicitation techniques derived from the social sciences are now being used to inform and augment requirements. Some of these techniques are:

- **Ethnography**
  Ethnography, the method of unobtrusive observation of the work setting and context, has been proposed and applied as an alternative requirements elicitation technique. One of the advantages of this method is that it reveals the true work practices instead of the 'prescribed practices' as set out in company handbooks and manuals (Sommerville & Rodden, 1994). Active research and applications of this technique in the computing domain is being pursued at the Lancaster University and in the CSCW research community.

- **Contextual Inquiry** (Holtzblatt & Beyer, 1996)
  Contextual inquiry is a requirements elicitation technique comprising the *contextual design* method. It is a bottom-up approach to requirements in which data about how people work drives the design process. This technique is also based on ethnography with the addition of contextual interview. With contextual inquiry, the future users of a system are observed and interviewed while they are currently performing their work. The researcher/observer can interrupt to ask questions as an outsider while the user is currently doing his/her task. The main idea is to derive design implications from a deeper understanding of how people work in which both formal and informal processes are taken into consideration.

- **Participatory Design**
  The participatory design technique, also known, as the 'Scandinavian school', in eliciting requirements is by allowing the future users of the system to take part in the actual design process (Kuhn & Winograd, 1996). In this approach, users then reflect on what their requirements are and they are also responsible for its analysis and implementation.

- **ETHICS (**Mumford, 1995)
  ETHICS (Effective Technical and Human Implementation of Computer-based Systems) is a method derived from socio-technical systems theory. It also relies on end-user participation in requirements elicitation. ETHICS expands the concept of requirements to include human requirements such as job satisfaction and quality of life in the requirements process. A variant of the ETHICS methods is QUICKEthics, which is a front-end process that requires a mix of activities directed at the eliciting accurate information. These activities include questionnaire, group discussions and the visual build-up of knowledge needs through prioritizing information needs and assembling these into an essential model (Mumford, 1995).

5. A Framework for the Requirements Process in an Evolutionary Implementation

In the beginning of this paper, we raised several issues and design challenges concerning the organizational implementation of new technology, in particular groupware applications. These issues relate to the mismatch between real world requirements and system functionalities brought about by a poorly informed requirements process, and to that of evolving requirements which prompts the need to come up with a design-oriented response to support and facilitate the change processes in system implementation. The theoretical basis from which to tackle and address these issues is drawn from socio-technical systems theory out of which a conceptual framework and an agenda for the requirements process is derived.

5.1 Socio-technical systems view on groupware implementation

Socio-technical systems theory can be used to describe the process surrounding groupware implementation. With this theory, groupware implementation in organizations can be explained as a process of mutual adaptation between two techno-social components: the groupware application and the group of people who make use of the system. The introduction of a groupware application in organizations brings about changes in the social structures affected directly or indirectly by its implementation. Likewise, any adoption of new technology leads to a new socio-technical state from which new forms of support is needed, making new demands to the system. In return, the interplay between these two components develops into a cycle of socio-technical dynamics imbued with the goal of attaining an optimal match between user needs and system functionalities. Implementation is thus, an evolutionary process characterized by changes in the state of the socio-technical system as a result of adaptation and integration.

The optimal match between users and system is substantiated by the concept of fit in the interaction between groupware and user group at the point of the task interface (Bikson & Eveland, 1996; Mumford, 1995; Trist; 1987). The factors of best match lie along the dimensions of groupware features, its functional components, vis-à-vis work group functioning, work processes, communication and relationship structures. The basic unit of analysis is the workgroup that immediately makes use of groupware technology. As an open system, the workgroup also relates to other social structures such as the individual and the organization as a whole.

This socio-technical dynamics cycle in the context of groupware implementation is represented in the following diagram:

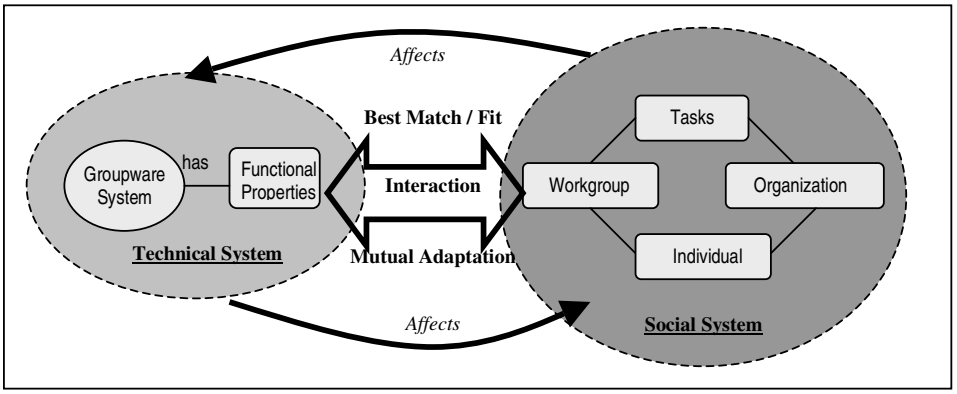

Fig. 5: Socio-technical dynamics for joint optimization

In this model, the socio-technical dynamics can be explained in terms of 3 variable groups: technical variables, social variables and interaction outcome variables. Constructs for these variables are shown in Table 3.

Table 3.
Socio-Technical Variables

Technical System Variables
Groupware application properties Product Identity – what the system is about Product type – in-house, COTS, contracted Level of specialization – whether general purpose or special purpose (ref: section 3.1) Functional properties – functions for enabling communication, information sharing and coordination Quality properties – quality application attributes such as speed, system requirements Application architecture – internal organization of the application in terms of its conceptual, module as well as physical organization
Social System Variables
Group characteristics Group attributes such as size, composition, location Group structure – leadership, hierarchy Group formation – how the group was formed, formal vs. informal membership Group awareness – low or high, cohesiveness Group behavior – cooperative vs. competitive (Karsten & Jones, 1998) Group autonomy – centralized vs. autonomous decision making, extent of freedom with regards to system design, experimentation in use (Trist, 1981) *Group Task Factors* Group Goals Group task characteristics – task interdependence, goal interdependence, information exchange needs Group task bottlenecks – problem areas affecting group task execution *Individual Characteristics* Personal background – educational, training, work experience Attitude towards technology Previous experience with groupware Training needs Skills required to perform tasks Job satisfaction criteria (Mumford, 1995) *Organizational and Contextual Factors* Organizational business domain Organizational support – budget, training, rewards Cultural contexts – perceptions on: Trust – perceived ability to rely on colleagues and managers Equity – feeling that efforts are fairly rewarded
Interaction-Outcome Variables
Group Outcomes Quality of group performance Quality of group cooperative/collaborative processes Group development – reinforced group awareness *Individual Outcomes* Individual expectations on the system Individual satisfaction on system use Appreciation of group membership Individual breakdowns in system use *System Outcomes* Enhancements – ways in which the system can be improved Affordances – new forms of system use other than expected

On the other hand, the best match or the socio-technical fit between the two systems is ascertained through a cross analysis of between groupware application properties e variables and constructs. These dimensions of fit are shown in Table 4.

Table 4.
Dimensions of Fit

Groupware properties vs. Group characteristics
Groupware properties vs. Group tasks
Groupware properties vs. Individual characteristics
Groupware properties vs. Organizational and contextual factors

5.2 The context of RE in an evolutionary implementation

Socio-technical systems theory provides for an agenda and context for requirements engineering in an evolutionary implementation. In its purview, STS suggests that new technology implementation should be accompanied by action research efforts in order to guide the achievement of the best match between user needs and system functionality. In software engineering terms, these efforts draw parallels with the requirements process wherein system functionalities are derived from an understanding of user needs.

In the search for the optimal match between social and technical systems, the discipline of requirements engineering can fulfill an important role of a method in facilitating the change processes resulting from new technology implementation. These, together with the design issues affecting the organizational implementation of new technology define a differentiated role and context for RE efforts. Thus, in the case of an evolutionary implementation, new concepts of requirements, as well as the requirements process arise:

- ***Requirements engineering as a continuing process***
  The case for continuing and extended RE has been argued for by Jarke and Pohl (1994) in consideration of continuous change and the need for a more responsive IT sector. This assertion even finds more relevance in the context of system implementation, wherein any technical work done on the system is referred to as system maintenance. In most software engineering models (see Pfleeger, 1999, ch 2; or Pressman & Ince, 2000), very little attention is paid to the implementation[1] aspect of a software system, except for maintenance. However, the bulk of software maintenance efforts (Munro, 1992; Frazer; 1992; Rajlich & Bennett, 2000) in which roughly 80% is spent on major design work[2] instead of fixing bugs (corrective maintenance) (Pressman & Ince, 2000). This information alone provides substantial evidence that RE ought to be a continuing process.

  Moreover, software design and development has already been articulated as an iterative and continuing process that overcomes the notion of software

[1] Implementation is a temporal concept to refer to the acquisition, installation and use of ICT in user organizations

[2] These software maintenance tasks would include: adapting existing systems to changes in their external environment, making enhancements requested by users and reengineering an application for future use.

maintenance. This implies that pre-cursor design activities such as requirements analysis persist. Hughes et al (1996) argues that software design issues do not cease with the initial specification of requirements, but persists when systems have been introduced into their settings. System use itself provides a significant source of design requirements that are often overlooked in the elicitation and maintenance processes.

And lastly, requirements evolve. The interplay between system and users bring about changes in the implementation environment. As both human and systems move from one structural state to another, new needs arise requiring new means of support. However, before systems are made to adapt to evolving requirements, it is necessary to first recover the initial requirements made of the system (Caroll et al, 1991)

The case for a continuing requirements process is likewise consistent to the claim that requirements for systems meant to support cooperative work such as groupware should be founded on actual instance of use. A continuing requirements process in the groupware implementation process serves to bridge the gap between the real work situation needs and system capabilities.

Continuing requirements process implies that the tasks of elicitation, analysis and verification persist and iterate in the implementation phase of new technology, supplemented by real world requirements identification in the sense of a socio-technical fit between human and social variables with that of technical properties (Mumford, 1995). In addition, the output of the process is not limited to design specifications pertaining to the technical system but also to social system, which may entail an organizational management strategy. This process is visualized in Figure 5 below:

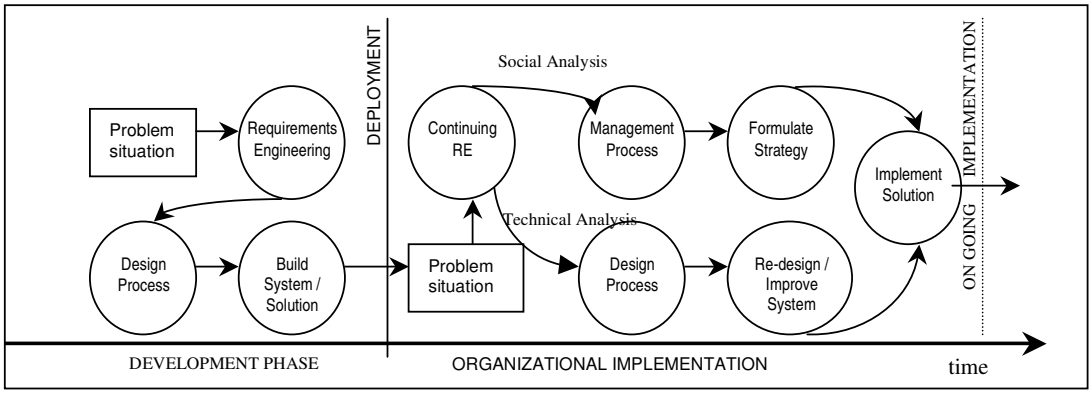

Fig. 4: The context of continuing RE

- ***Broader view of requirements***

  According STS theory the concept of requirements relevant to system design is not limited to business and system requirements. There are also human requirements and social requirements that need to be taken into consideration in system design such as job satisfaction and autonomy (Mumford, 1995; Land, 1987). Human users of systems have psychological and emotional needs that influence the way they perceive and interact with a computer system when they perform their tasks. Groups of people on the other hand have their own dynamics

that need to be properly understood in order to effectively be able to support them. These aspects are actually beyond the normal scope of a design task and designers are not experts in these matters. But these are nonetheless very important. Additional knowledge and exposure in these activities need to be acquired.

- ***Better Informed RE Process***
  Human and socially oriented elicitation techniques that afford rich descriptions and exposure to the context of work and system use can help augmenting the RE process in obtaining holistic view of the broader scope of requirements. Techniques such as contextual inquiry and observational studies are suitable techniques from which to derive a better understanding of group processes and groupware use.

While conceptualized as a continuing process in the organizational implementation phase of a software product, RE in the context of system implementation does not imply a mirror image of the RE tasks normally undertaken in the development phase of a system. Special attention to evolution and requirements change necessitates that a regulatory cycle of evaluation is put in place. More importantly, within the socio-technical design framework, RE fulfills the role of a method from which to derive the best fit between socio-technical variables. This implies that the scope of RE activities is not limited to determining system specifications only.

6. Towards a research model for an evolutionary approach to groupware implementation

In this paper, we have laid the agenda for an evolutionary approach to groupware implementation, founded on the design issues and challenges confronting the organizational adoption of collaborative technology. The evolutionary approach to groupware implementation proposed herewith is a method of analysis and applied research, drawing theoretical inputs from socio-technical systems design framework that:

- Recognizes the continuing role of the RE process in supporting the implementation of new technology as a means to seek the optimal fit between user needs and system properties;
- Sanctions a more human and socially informed RE process in the use context of groupware applications
- Expands the extent of RE efforts to include system evaluation with regards to initial requirements, groupware context assessment, fitness analysis, system-push requirements elicitation as well as socio-technical system requirements specification.

This approach encapsulated as a continuing RE process within the socio-technical framework consisting of differentiated activities is provided for in the following research model given in Figure 5:

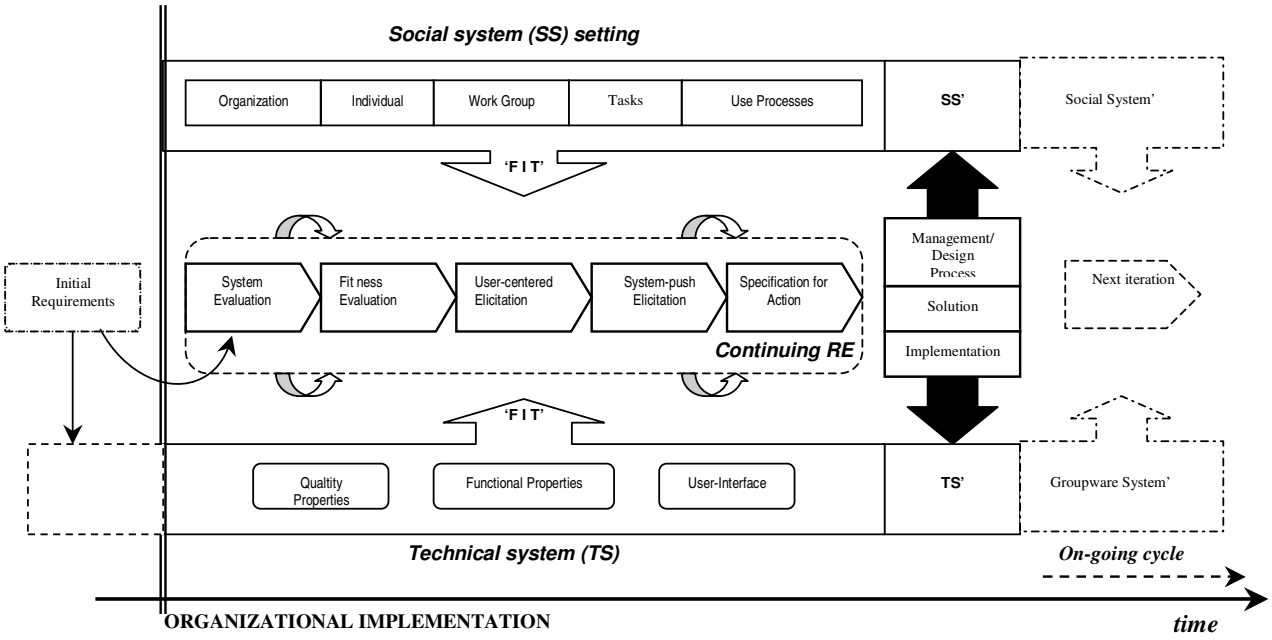

Fig. 5: A model for an evolutionary approach to groupware implementation

In this model, requirements engineering serves as feedback mechanism and a diagnostic process for anticipating requirements evolution in the course of implementing and using a new technology. Table 5 provides a framework plan on how this model can be carried out.

Table 5
Framework plan for an evolutionary method

RE Effort	Description and Activities
System Evaluation	Evaluation to which extent the implemented system fulfill the original requirements for the system. Activities include: - Recovering the initial and agreed upon requirements for the system: *What are the needs that gave rise to system implementation? Who were the ones who wanted the system?* - Goal vs. implemented system assessment: *Were the goals set for the system achieved?*
Fitness Evaluation	Analysis of the problem situation using the socio-technical systems concept of fit. This effort should attempt to describe how well the properties of system matches the characteristics of the social environment involved with the system. Activities include: - Analysis of groupware implementation context (refer to section 3.1) - Comparative analysis of groupware functions, non-functional properties and user interface with regards to its process of use. Relevant questions are: *Does the user like the system? Does the system help the individual perform his/her daily task? Does the use of the system lead to an individual's job satisfaction? Does the system help the group perform collaborative tasks? How does the system influence an individual's attitude towards group processes? How crucial is the system in the day –to- day operations of the individual? group?organization?*

User-centered Elicitation	Deriving real world needs from the actual use of the system and from studies of how work is actually done. Alternative requirements elicitation techniques such as ethnography and contextual inquiry are advocated. Activities can consist of: - Apprenticing with users - Performing observational studies - Task analysis - Needs assessment taking into consideration personal requirements for job satisfaction
System-push Elicitation	Analysis of how the technical system is actually being used by the users, i.e. for what specific purposes. Deriving design and management implications from the requirements elicitation process with the focus on determining what kind of support is needed to help users do their tasks? Relevant questions are: *What other purposes can the system be used for? What are the problems and difficulties encountered when using the system? How can be the system improved? What kind of support do the users need in helping them do their tasks?*
Specification for Action	Defining design changes or enhancements to be made to the technical system and making recommendations for providing organizational support to the users of the system.

The central idea behind this model is that when a developed system such as groupware is implemented, continuing support for the process is needed in order to get closer to the purposes in which the system was built or acquired for. The requirements process when building a system can provide this kind of support. However, the extended process needs to take into consideration the social aspects of implementation and a more holistic perspective is needed. Adapting a system towards evolution does not entail making changes to the technical system only but also to the social world of users who interact with the system.

7. Future Work

It is our goal to conduct systematic investigation of groupware implementation in organizations by way of case studies. It is our goal to understand patterns of adaptation in the implementation process. Likewise, it is also our goal to apply the method to in order to abstract guidelines for managing requirements evolution as well as the implementation process in general. We believe that action research is the most feasible research strategy to adopt carry out long-term investigations on real life groupware implementations.

At the moment, we have one on-going case study:

- Knowledge Management System at a large Dutch holding company in the insurance business, a merger of several independent insurance companies.

Other case studies being initiated include:

- Voice-operated alarm system at a health care institution
- Lead management system at a Dutch bank.

References

Alexander, I. (2002). Being Clear: Requirements are EITHER Needs or Specifications. Requirenautics Quarterly: The Newsletter of the Requirements Engineering Specialist Group of the British Computer Society, 25, pp. 10-12.

Bannon, L. and Hughes, J. (1993). The Context of CSCW. In K.Schmidt (ed.) *Developing CSCW Systems: Design Concepts*. Report of COST14 "CoTech" Working Group4 (1991-1992), Feb 1993, pp. 9-36.

Bentley, R., Appelt, W., Busbach, U., Hinrichs, E., Kerr, D., Sikkel, K., Trevor, J. and Woetzel, G., (1997). Basic support for cooperative work on the World Wide Web.*International Journal of Human-Computer Studies* Vol. 46, pp. 827-846.

Beyer, H., & Holtzblatt, K. (1998). *Contextual Design: Designing Customer-Centered Systems*. San Francisco, California: Morgan Kaufman.

Bikson, T., and Eveland, J.D. (1996). Groupware implementation: reinvention in the sociotechnical frame. *Proceedings Conference on Computer Supported Cooperative Work 1996* pp. 428-437.

Bullen, C. & Bennett, J. (1990) Learning from user experience with groupware. In *Proceedings Conference on Computer Supported Cooperative Work '90*, October, Los Angeles, CA, pp291-302.

Bock, G. (1992). Groupware: software for computer-supported collaborative work. In. D. Marca & G. Bock (Eds.) *Groupware: software for computer-supported collaborative work*. Los Alamitos, CA.: IEEE Computer Society Press.

Bondarouk, T. and Sikkel, K. (2002). Explaining Groupware Implementation through Group Learning. Paper submitted for publication.

Carroll, J. M. , Kellogg, W. A. and Rosson, M. B. (1991). The task-artifact cycle. In J. M. Carroll (Ed.) *Designing Interaction: Psychology at the Human-Computer Interface*. Cambridge: Cambridge University Press.

Ciborra, C. U. (1996). Introduction: What does groupware mean for the organizations hosting it?. In Ciborra, C.U., *Groupware and Teamwork: Invisible Aid or Technical Hindrance?* (Wiley Series in Information Systems, pp. 1-19.). Chichester: John Wiley & Sons.

Coleman. D. (1997). *Groupware: Collaborative Strategies for Corporate LANS and Intranets*. San Francisco, CA: Prentice-Hall.

Collis, B.A., & Boer, W.F. de.(1999). The TeLeTOP Method at the University of Twente. *International journal of educational telecommunications*, Vol. 5 No.4, pp. 331-359.

Compton, J. (2001), (Article found on the internet). Groupware grows up. In <http://www.zdnet.com/products/stories/reviews/0,4161,2698269-7,00.html>

Daft, R. (1998). *Organization theory and design*. (Sixth Edition). Cincinnati, Ohio: South-Western College Publishing.

Ellis, C. A., Gibbs, S.J., and Rein, G.L. (1991). Groupware: some issues and experiences. *Communications of the ACM, 34 (January)*(1).

Frazer, A. (1992). Reverse Engineering - hype, hope or here?. In P.A.V. Hall, *Software Reuse and Reverse Engineering in Practice* (pp. 208-242). Boundary Road, London: Chapman & Hall.

Gause, D., and Weinberg, G. (1989). *Exploring Requirements Quality Before Design*. Dorset House Publishing, New York.: Dorset House.

Goguen, J. A. (1993). *Social Issues in Requirements Engineering*, Proceedings of the IEEE International Symposium on Requirements Engineering, San Diego, California, January 4- 6, pp. 194-195.

Goguen, J. (1994). Requirements Engineering as the reconciliation of the technical and social issues. In Goguen, J.A. and Jirotka, M (Eds.), *Requirements Engineering: Social and Technical Issues*, Academic Press.

Greenberg, S. (1991) Computer-supported Cooperative Work and Groupware. In Greenberg, S. (Ed.), *Computer-supported Cooperative Work and Groupware (pp. 1-8).* Academic Press.

Greif, I. (Ed.) (1988) *Computer-Supported Cooperative Work: A Book of Readings*. San Mateo, CA: Morgan Kaufmann.

Grudin, J. (1988). Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces. *Proceedings of the conference on Computer-supported cooperative work* Portland, Oregon, USA. (September 26-28, 1988)

Grudin, J. (1991). CSCW: The convergence of two development contexts. In *ACM SIGCHI Conference on Human Factors in Computing Systems* (April). New Orleans, ACM Press.

Grudin, J. (1994). Groupware and social dynamics: Eight challenges for developers. *Communications of the ACM, 37*(1), pp 92-105.

Grudin, J. (2001). Collaboration Technology in Teams, Organizations and Communities. Tutorial Notes. *Seventh European Conference on Computer Supported Cooperative Work 2001*. 16-20 September. Bonn, Germany.

Hughes, J., O'Brien, J., Rouncefield, M., & Rodden, T., Sommerville, I. (1994). Presenting Ethnography in the requirements process. *Proceedings 2nd IEEE International Symposium on Requirements Engineering*, York, England, IEEE Computer Society Press, pages 27-34.

Hughes, J., O'Brien, J., Rouncefield, M., & Rodden, T. (1996). ″They're Supposed to Be Fixing it": Requirements and System Redesign. In Peter Thomas (Ed.) *CSCW*

Requirements and Evaluation. London: Springer-Verslag.

Jarke, M., and Pohl, K. (1994): Requirements engineering in 2001: (virtually) managing a changing reality. *Joint Special Issue on Software Engineering beyond 2001, IEEE Software Engineering Journal*, 9(5).

Karsten, H. & Jones, M. (1998) The Long and Winding Road: Collaborative IT and Organisational Change. In *Proceedings of Computer Supported Cooperative Work '98*. Seattle, Washington: ACM

Kuhn S. & Winograd, T. (1996). Participatory Design. In Winograd, T., Bennett, J., De Young, Y., & Hartfield, B.(Eds.), *Bringing Design to Software*. New York. Addison-Wesley.

Johansen, P. (1988). *Groupware: Computer Support for Business Teams*. New York: The Free Press.

Land, F. (1987). Adapting to Changing User Requirements. In Galliers, P. (Ed). *Information Analysis: selected readings*. Addison Wesley Publishing, Sydney.

Laudon, K., and Laudon, J. (2000). *Essentials of Management Information Systems Organization and Technology in the Networked Enterprise* (Fourth Edition). New Jersey: Prentice-Hall.

Lauesen, S. (2000). *Software Requirements - Styles and Techniques.* Denmark: Samfundslitteratur.Frederiksberg.

McGrath, J. & Hollingshead, A.B. (1994). *Groups interacting with technology: ideas, evidence, issues and an agenda*. Thousand Oaks, CA: Sage.

Mumford, E. (1987). Sociotechnical Systems Design; Evolving theory and practice. Bjerknes G. & Ehn P. & Kyng M. (eds.) *Computers and Democracy*: *A Scandinavian Challenge*. Avebury, pp. 59-76.

Mumford, E. (1995). *Effective Systems Design and Requirements Analysis - The ETHICS Approach*. MacMillan Press Ltd.

Mumford, E. (2000). A Socio-Technical Approach to Systems Design. *Requirements Engineering,* 5, pp. 125-133.

Munro, M. (1992). Software maintenance, reuse and reverse engineering. In P.A.V. Hall, *Software Reuse and Reverse Engineering in Practice*. Boundary Road, London: Chapman & Hall.

Norman, D. (1991).Collaborative Computing: Collaboration First, Computing Second. *Communications of the ACM* 34(12) pp. 88-90

Norman, D. (1993). *Things that make us smart: defending human attributes in the age of the machine*. Massachussetts: Addison-Wesley.

Nunamaker, J., Dennis, A., Valacich, J., Vogel, D., George, J. (1989). Electronic Meeting Systems To Support Group Work. *Communications of the ACM*, 34 (7): 40-61

Nusebeih, B., and Easterbrook, S. (2000). Requirements Engineering: A Road Map. *Proceedings of International Conference on Software Engineering (ICSE-2000)* Limerick, Ireland: ACM Press. (June 4-11, 2000)

Orlikowski, W. (1996). Evolving with Notes: Organizational change around groupware technology. In Ciborra, C.U., *Groupware and Teamwork: Invisible Aid or Technical Hindrance?* (Wiley Series in Information Systems, pp. 23-60). Chichester: John Wiley & Sons.

Orlikowski, W., and Hofman, J. (1997). An improvisational model of organizational change: the case of groupware technologies. *Sloan Management Review, Winter 1997,* pp. 11-21.

Pipek, V., and Wulf, V. (1999). A Groupware's Life. In S. Bødker, M. Kling, and K. Schmidt, *Proceedings of the Sixth European Conference on Computer Suppored Cooperative Work, 12-16 September 1999, Copenhagen, Denmark* pp. 199-218. The Netherlands: Kluwer Academic Publishers.

Pressman, R., & Ince, D. (2000). *Software Engineering: a practitioner's approach.* London: McGraw Hill.

Pfleeger, S.L. (1999). *Software Engineering: Theory and practice*. Prentice-Hall International, Singapore.

Put, F. (1996). Computerondersteunend samenwerken: classificatie en fundamenten van groupware. *Informatie,* March, pp. 7-12.

Rational Software. Rational Rose Software. http://www.rational.com.

Rajlich, V., and Bennett, K. (2000). A Staged Model for the Software Lifecycle. *IEEE Computer, 33*(7), pp. 66-72.

Ruël, H. (2001). *The non-technical side of office technologies!* PhD Thesis. University of Twente, Enschede, The Netherlands.

Siebel Systems. http://www.siebel.com

Sommerville, I., Bentley, R., Rodden, T., and Sawyer, P.(1994). Cooperative System Design. *The Computer Journal*, 37(5), pp. 357-366.

Sommerville, I., and Rodden, T. (1994). *Requirements Engineering for Cooperative Systems*. Centre for Research in CSCW, Lancaster University.

Sommerville, I. & Sawyer, P. (1997). *Requirements Engineering: A good practice guide*. John Wiley & Sons, Chichester.

Trist, E.L. (1981). *The Socio-Technical Perspective*. The evolution of sociotechnical systems as a conceptual framework and as an action research program. In van de Ven A. & Joyce W.F. (eds.) *Perspectives on Organization Design and Behaviour*.Wiley, pp. 49-75.

Von Bertalanffy, L. (1950). The Theory of Open Systems in Physics and Biology, *Science*, Vol.111 (1950), pp.23-29.

Wieringa, R. (2001). A Framework for Aligning Software, Architecture & Requirements. Unpublished paper.

Wieringa, R. (2002). Requirements Engineering. Course material for the course on Requirements Engineering. University of Twente, Enschede, The Netherlands.