Chapter 2
Groupware development in the human context

In the past few decades, since the intervention of personal computers, computational artifacts have been proliferating dramatically. Computer technology has entered our daily lives in many aspects, from organized to unstructured ad-hoc activities, from office to home, from formal work to entertainment, and it is playing an essential role in human communication and collaboration. Yet, the diverse, pervasive and ubiquitous use of computer technology poses new challenges to research into the development of computer technology when seamless integration to human activities and context are the primary concerns. Research communities such as Human Computer Interaction (HCI), Computer Supported Cooperative Work (CSCW), Participatory Design (PD), and Information Communication Technology (ICT) focus on different facets in the development of computer technology and yet share a broader common concern: how can computer technology support human activities efficaciously. In CSCW, the challenge is referred to as the *social-technical gap*: the gap between what we can support technically and what we must support socially (Ackerman, 2000). It is this socio-technical challenge that frames the broad context for this chapter. Instead of studying the broader context, this chapter focuses on the development of groupware, a kind of computer system which is more sensitive to the socio-technical challenge than other species.

This chapter develops an argument that there is no silver bullet for groupware development. Successful groupware development requires insightful understanding of how work is carried out in practice, and contemporary research does not offer a unified coherent theoretical framework for this purpose. Beginning with a taxonomy of group work, section 2.1 discusses the five important aspects of group work. That is, group awareness and coordination of

work, articulation work, creativity, experimentation, knowledge sharing and knowledge construction. These are the basic aspects that successful groupware has to support.

Groupware is a socio-technological system and its development is unusually challenging due to the dynamic nature of groups and the evolving environment for group work. Section 2.2 discusses the socio-technical problem and the issues that are at the centre of concerns in groupware development. Contemporary paradigms for groupware development are reviewed in section 2.3. Due to the differences among groups and the dynamic nature of group work, a generic approach to groupware development is unlikely to suit all types of group work or all groups. Section 2.4 discusses the evolving nature of human activity, current theoretical influences on groupware development, and paradigms to evolve groupware. Indeed, in supporting human activities, the groupware must be able to *evolve*, allowing re-configuration throughout its lifecycle. Section 2.5 discusses a novel perspective on groupware development that is *efficacious* in the sense discussed by Jullien (1995) with reference to the notion of disposition.

2.1 Supporting group work

In order to provide effective computer support for group work, it is vital that groupware developers and groupware methods developers recognize the different types of group work and their nature. It is also essential to know that all successful groupware has some common features. In this section, a taxonomy of group work will be derived from the classification of groupware and five common features of successful groupware will be discussed.

2.1.1 Taxonomy of group work

Classifications of groupware such as (Johansen, 1988), (Teufel et al., 1995) and (Ellis et al., 1991) provide useful information to help users of off-the-shelf software to understand the kind of support that a specific groupware is offering and therefore help them to select the appropriate tool according to their needs. These classifications also provide a shared reference point, which eases the communication among groupware researchers, developers and other participants who are involved in the development process. Indeed, each

classification scheme reflects a facet of the whole spectrum of group work.

Johansen (1988) proposed a two dimensional taxonomy for groupware. He suggests that groupware can be conceived to support collocated or distributed groups in the space dimension, and asynchronous or synchronous work in the time dimension. Most of the groupware classifications are derived from Johansen's space/time classification. For example, Grudin (1994b) has extended Johansen's taxonomy to take account of the predictability of space and time constraints, which results in a 3x3 matrix, and the impact this has on the way in which group work is organised. Andriessen (2003 p.12) combines Johansen's classification with five functions of ICT in the group process. Recently, Penichet et al. (2007) produced a more complicated 7-dimensional matrix, which combine CSCW characteristics (i.e. information sharing, communication, and coordination) with a modified space/time dimension matrix. Johansen's and Grudin's classification were popular in 1990s and remain highly cited amongst CSCW researchers.

There are other classifications of groupware which do not follow Johansen's dimensional perspective. For example, the 3C model (Teufel et al., 1995) reflects a triangular relationship among communication, coordination and cooperation (cf. Borghoff and Schlichter, 2000), but such reflection may contradict the four-level of communication perspective (Borghoff and Schlichter, 2000, p.110). Ellis et al. (1991) differentiates groupware according to which application features they support. In Ellis et al's classification, different groupware may share common features. Because of this, compared to Johansen's classification and its derivatives, these classifications reflect less useful perspectives for understanding the characteristics of group work.

Apart from space, time, predictability, and CSCW characteristics, group work can also be classified as structured or unstructured, planned or ad-hoc, data centric or process centric, and according to other dimensions such as group size (cf. Johansen, 1988; van der Aalst, 2007). Moreover, CSCW characteristics can be extended into a full spectrum of degrees of engagement (which will be discussed in chapter 5). There is endless possibility in classifying group work. Certainly, we can enrich the taxonomy in an open-ended way with additional dimensions through further analysis of group work, but more significantly, we shall recognise

that group work in practice is usually dynamic. It rarely settles into one particular mode of interaction. Actual human collaborations are fluid and do not usually follow a single pattern of work.

2.1.2 Awareness and Coordination

Awareness of other people's work is crucial in a group work setting. As Dourish and Bellotti (1992) describe:

> *"awareness is an understanding of the activities of others, which provides a context for your own activity. This context is used to ensure that individual contributions are relevant to the group's activity as a whole, and to evaluate individual actions with respect to group goals and progress. The information, then, allows groups to manage the process of collaborative working." (ibid)*

This understanding is an ongoing interpretation of both one's own and others' activities and artifacts involved in the process of group work (Chalmers, 2002), and it enables an individual to make conscious decisions (Borghoff and Schlichter, 2000) that promote a harmonious interaction, which eventually improves the productivity of the group (Kirsch-Pinheiro et al., 2003).

Dourish and Bellotti's (1992) notion of awareness is generally referred to as "group awareness" (Gutwin, 2004). In sociology, awareness is referred as shared cognition or team knowledge (Andriessen, 2003; Cannon-Bowers and Salas, 2001). In terms of computer support, awareness can be further categorized according to the nature of information it discloses and supports, e.g., presence awareness, emotion awareness, workspace awareness (Gutwin, 1997), change awareness (Tam and Greenberg, 2004), activity awareness (Hayashi et al., 1999; Carroll et al., 2006), gaze awareness (Ishii and Kobayashi, 1992).

According to Dourish and Bellotti (1992), awareness has dual roles in group work: a high-level of awareness of other people's work helps to avoid redundancy of work, while a low-level awareness allows "fine-grained shared working and synergistic group behaviour". (Dourish and Bellotti, 1992) The support to awareness is important for coordinating group

work (Dourish and Bellotti, 1992) and this inevitably involves division-of-labour. Lack of support to awareness in some situations may have unpleasant consequences, e.g. presence disparity in mixed-presence groupware (Tang et al., 2004).

Although the impact of awareness on group work is clear, implementing awareness support in groupware remains challenging. As Chalmers (2002) pointed out "representation and interpretation affect the degree and character of awareness afforded by computer systems: awareness of people and of information artifacts". This highlights the complexity and the socio-technical nature (cf. §2.2.1) of the issue. Further, awareness information can overload the recipient if not carefully designed (Kirsch-Pinheiro et al., 2003). In some situations, it may become a disturbance and significantly degrade the performance of an individual and the system (Dourish and Bellotti, 1992). It is difficult to maintain a balance between the amount of information to be disclosed and the degree of awareness of other people's work. This is because different activities and different group, with few exceptions, require different kind of awareness support (Kirsch-Pinheiro et al., 2003). This makes it hard to generalize approaches to providing support for awareness and groupware developers often have to build this support from scratch (Kirsch-Pinheiro et al., 2003).

2.1.3 Articulation work

Research into articulation work has gained much attention in CSCW since Schmidt and Bannon (1992) observed that the lack of focus of the field "may hinder its further development and lead to its dissipation", and suggested that "CSCW should be conceived of as *an endeavor to understand the nature and requirements of cooperative work with the objective of designing computer-based technologies for cooperative work arrangements*" (ibid, p.5, italics in original).

The notion of articulation work was first introduced by a sociologist, Anselm Strauss, in 1985 (Strauss, 1985; Fjuk et al., 1997). It is used as a framework to study the interwoven nature of the division-of-labour and the interdependence among individuals in group work (Fjuk et al., 1997). Articulation work refers to the interaction between individuals that is necessary to, e.g. coordinate, align, schedule, and integrate interdependent activities which occur during the course of cooperative work (Schmidt and Simone, 1996; Lee, 2005). Strauss (1993)

defines:

> *"Articulation stands for the coordination of lines of work. This is accomplished by means of the interactional process of working out and carrying through of work-related arrangements. Articulation varies in degree and duration depending upon the degree to which arrangements are in place and operative." (Strauss, 1993, p.87)*

Articulation work is different from cooperative work. While the former is concerned with coordinating individual tasks taking into account the distributed nature of group work, the latter merely concerns the distribution of tasks to individuals who shared a common workspace, as Schmidt and Simone (1996) pointed out:

> *"Cooperative work is constituted by the interdependence of multiple actors who interact through changing the state of a common field of work, whereas articulation work is constituted by the need to restrain the distributed nature of complexly interdependent activities." (Schmidt and Simone, 1996, p.158)*

The relationship between cooperative work and articulation work is recursive since the management of an articulation work can invoke another articulation work or cooperative work, which again may yield another articulation work (Star and Strauss, 1999, Schmidt and Simone, 1996). This recursion, in theory, can be infinite, but, as Schmidt and Simone (1996) have pointed out, real world work practices usually terminate such recursions before they degenerate into chaos.

Supporting the ongoing articulation work is one of the key issues in CSCW (Schmidt and Bannon, 1992). Central to articulation work is the concern for informal interactions, that are not explicit in the formal description of work, to bring the work back on to track when there are unexpected contingencies and breakdowns (Fjuk et al., 1997; Star and Strauss, 1999; Evans et al., 2001). Schmidt and Simone (1996) observe that articulation work may become intuitive and remain invisible in collocated group work where actors can interpret others' activities through their bodily actions. Unfortunately, articulation work becomes "extremely complex and demanding" in a real world complex group work setting (Schmidt and Bannon, 1992; Schmidt and Simone, 1996). Providing computer support for articulation work remains

difficult due to the situated characteristics of the work activity (cf. Suchman, 1987; Grinter, 2000). Indeed, it may entail a thorough analysis of the culture and politics of the work, if the work is to be made visible (Star and Strauss, 1999; Schmidt and Bannon, 1992).

2.1.4 Creativity

Creativity has been a topic for research in sociology and psychology, but only recently has it caught the attention in HCI and CSCW, e.g. collaboratories (Sonnenwald, 2003; Farooq et al., 2005). Despite the fact that human creativity has the characteristics of being "inventive, wily, sly, cunning, and crafty" (Spinuzzi, 2003, quoted by Kaptelinin and Nardi, 2006), individual creative contributions are essential in group work because they involve "an instant restructuring of the whole representation of a problem" (Kaptelinin and Nardi, 2006; also cf. Wertheimer, 1966). Although this process is entirely internal to an individual, Kaptelinin and Nardi (2006, p.210) also acknowledge that "a subject can reframe a representation to successfully solve a problem", and the importance of this framing of a problem in problem solving has already been stressed by Jackson (2001).

Although creativity is still by-and-large viewed as a quality of an individual and is sometimes viewed as a gift endowed at birth, there is no doubt that many creative ideas are the result of collaborative work (Nijstad et al., 2006). Fischer (2005) echoes this view by saying:

> *"Much human creativity is social, arising from activities that take place in a social context in which interaction with other people and the artifacts that embody collective knowledge are essential contributors." (ibid)*

A similar perspective can also be found in (Kaptelinin and Nardi, 2006), in which the authors argue that "[t]here is no lone genius" (ibid, p.214).

Group creativity is not merely a sum of individuals' creative work as if they are working alone, but it is an emerging property of the group (Stahl, 2006; Kaptelinin and Nardi, 2006). However, there is much evidence in sociology suggesting that groups are far less productive and creative than the sum of their individual talents might indicate, e.g. (McGrath, 1984; Mullen et al., 1991; Nemeth and Nemeth-Brown, 2003; Paulus et al., 2006). It is not unusual to find that groups are making poorer decision than individuals (Nemeth and Nemeth-Brown,

2003). Nijstad et al. (2006) observed that this reduced productivity is due to 3 reasons: i) group members must express their ideas in turn; ii) group members are unwilling to share their ideas[6], iii) high-performing individuals tend to match their effort to that of their low-performing colleagues. While there is seemingly little computer technology can do in case (ii) and (iii) (Nijstad et al., 2006), there is certainly a role that computer technology can play in supporting creativity in group work.

In tackling (i), Nemeth and Nemeth-Brown (2003) stress the benefits of allowing different perspectives in the group. This proposal is strengthened by Nijstad et al's (2006) observation that if individuals could express their ideas simultaneously through e.g. typing and writing, though not verbal communication, there would be less productivity loss (2006). Similar thinking along these lines can be found in (Farooq et al., 2005), in which they propose to give support for i) divergent and convergent thinking, ii) the development of shared objectives, and iii) reflexivity.

It is worth noting that the development of computer support for creativity is still at an early stage. As Kaptelinin and Nardi (2006) point out, there is still no framework for analyzing how effects at an individual level can cause effects at the group level.

2.1.5 Experimentation

Following the creativity thread that has been discussed in the previous section, there are situations in which one may wish to explore an innovative idea further, e.g. to explore a configuration to see if it fits the situation at hand. This can be due to uncertainties about the problem at hand, the desire to gain further understanding about the current situation, or imperfect knowledge that necessitates learning by doing. I refer to this kind of problem solving related activity as *everyday experiment*.

Everyday experiment is different from a controlled experiment that is usually carried out in a

[6] In some situations, individuals may be reluctant to share their perspectives because it may conflict with the group's perspective. In order to maintain a consensus, they usually drop their perspectives silently (Baron et al. 1992; Paulus et al. 2006). Janis (1982) refers to this phenomenon as 'Groupthink'. This usually results in reduced creativity for the group (Nemeth and Nemeth-Brown 2003).

laboratory[7]. In contrast to laboratory experimentation, everyday experiment is informal and casual in nature, e.g. it is subject to fewer safety regulations. In the broad sense, experiments can be classified as *exploratory* experiments or *post-theory* experiments, depending on whether "there is a reliable basis for prediction, or criterion for successful outcome, drawn from theory or previous experience" (Beynon and Russ, 2008).

Experimentation has been seen as significant in innovative work[8], e.g. product design (Thomke, 1998), because "all experiments yield information that comes from understanding what does work and what does not work" (Thomke, 2003, p.20). Although Thomke (2003) is more concerned with the systematic experimentation involved in the use of computer simulations, models, and prototypes in controlled environments, I believe that the less-rigorous everyday experiment can also stimulate new ideas in group work.

2.1.6 Knowledge sharing and knowledge construction

As Divitini et al. (1993) observe, knowledge is crucial for group learning and its continuous development:

> *"The knowledge generated and used by the members of an organization in their cooperative work is a fundamental context for the effective accomplishment of their on-going activities as well as a basic background for the future ones." (ibid)*

Knowledge sharing and knowledge construction are two closely related concerns that are sometimes regarded as one concern under the umbrella term of knowledge management. Knowledge sharing can be conceived as part of the collaborative knowledge construction process (cf. Stahl, 2006). It concerns the exchange of information, i.e. knowledge, among individuals in groups, and it is one of the basic group processes (Andriessen, 2003). Sharing knowledge, like sharing awareness information (cf. §2.1.2), also suffers from similar social

[7] In the narrow sense, an experiment can only be carried out in a laboratory environment, in a controlled fashion.

[8] There is a subtle distinction between creativity and innovation. Amabile et al. (1996) define "… creativity as the production of novel and useful ideas in any domain [and] innovation as the successful implementation of creative ideas within an organization. In this view, creativity by individuals and teams is a starting point for innovation; the first is a necessary but not sufficient condition for the second" (ibid).

concerns – there may be no direct benefit for sharing knowledge with the group, and no guarantee that the other group members may benefit from the shared knowledge.

Knowledge construction concerns the emergence of knowledge and is usually part of the collaborative learning process in a group. Because of that, it is more often studied in the context of, e.g. computer supported collaborative learning (CSCL), knowledge management, and organisational studies.

Andriessen (2003) described two strategies, originally proposed in (Hansen et al., 1999), to provide technological support for knowledge management in groups. The first approach, the *codification strategy* entails codifying knowledge. This can be successful in relation to storing a large amount of information, but is less effective in relation to practical useful *knowledge*[9] (Andriessen, 2003). Andriessen (2003) argues that the limitations of codification stem from the loss of the personal experience and the contextual information that is originally embedded in the *knowledge*. The second approach, the *personalisation strategy*, focuses on inter-personal communication and knowledge sharing, mentoring, and *communities of practice* (CoP) (Andriessen, 2003). In this approach, the technological support is limited to providing pointers to the expertise (e.g. yellow pages) and promoting the communication between people (Andriessen, 2003). Andriessen (2003) stresses that each strategy has its own merits – the former is better for routine work, and the latter is more suitable for non-routine work.

2.2 Issues in groupware development

In this section, I first define what I mean by groupware, then I move on to the discussion of the socio-technical problem in groupware development. I then revisit Brooks's (1987) "No Silver Bullet" argument. Finally I argue that there is no standard way to approach groupware development and, due to its socio-technical nature, groupware development is a situated activity.

[9] Andriessen (2003) pointed out that, in the strict sense, once knowledge is stored in an 'impersonal system', e.g. a computer system, it is "no longer knowledge, it has become information".

2.2.1 Groupware and CSCW

Many people believe that the study of groupware is equivalent to the study of CSCW. However, this is not quite the case (cf. Schmidt and Bannon, 1992). The term 'groupware' was used by Johnson-Lenz and Johnson-Lenz (1982) before the term CSCW was coined, and it was originally used to refer to the software that supports group processes (Penichet et al., 2007). CSCW is a multidisciplinary research field with a much broader scope, which includes, e.g., computer science, psychology, and sociology.

Groupware, literally, is a combination of the word 'group' and the word 'software' which suggests a meaning of *groups' software* or *software for groups*. However, the term does not have a standard definition in the literature. For example, some people consider workflow systems as groupware, but others do not (cf. Bannon, 1992a). Johansen (1988) defines

> *"Groupware is a generic term for specialized computer aids that are designed for the use of collaborative work groups. Typically, these groups are small project-oriented teams that have important tasks and tight deadlines. Groupware can involve software, hardware, services and/or group process support." (Johansen, 1988; cited by Borghoff and Schlichter, 2000, p.92)*

Ellis et. al. (1991) define groupware as:

> *"computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment." (ibid)*

More recently, Orlikowski and Hofman (1997) describe groupware technology in the following term:

> *"Groupware technologies provide electronic networks that support communication, co-ordination and collaboration through facilities such as information exchange, shared repositories, discussion forums and messaging." (Orlikowski and Hofman, 1997)*

Johansen's (1988) definition includes all kinds of computer technologies for supporting

collaborative work. It indicates that groupware development should include the development of hardware, software, and other kinds of system support. In Ellis et. al. (1991), the focus is on providing a shared workspace for a common goal. Orlikowski and Hofman (1997) believe that groupware should focus on supporting group interaction through technologies, with no particular emphasis on supporting surrounding human activities carried out without directly interacting with computers. This divergence in perspectives is highlighted by Grudin (1991a), who points out that researchers even disagree on the classification of email. Whereas Kraut believes that email is the only successful groupware, Bannon and Schmidt argue that email is merely an enabling technology (Grudin, 1991a).

The term "computer-supported cooperative work" (CSCW) was coined by Paul Cashman and Irene Greif in a workshop held in 1984 (Grudin, 1994b). Greif (1988) described CSCW as "an identifiable research field focused on the role of the computer in group work" (quoted in Andriessen, 2003, p.10). However, the scope of CSCW was broadened to not only finding a role for the computer but to understanding how people actually work when Wilson (1991) said:

> *"CSCW is a generic term which combines the understanding of the way people work in groups with the enabling technologies of computer networking, and associated hardware, software, services and techniques" (quoted by Borghoff and Schlichter, 2000, p.92).*

This broadening of scope becomes yet clearer when Bannon and Schmidt (1989) write:

> *"CSCW should be conceived as an endeavor to understand the nature and characteristics of cooperative work with the objective of designing adequate computer-based technologies". (ibid, p.360)*

In the context of this thesis, the term CSCW is used to refer to the research field and its theoretical foundations for building computer systems, and the term groupware is used to refer to software in the computer system (in the narrow sense).

2.2.2 The socio-technical challenge

When compared with single user software development, groupware development is more

challenging. On the one hand, groupware developers are facing all the problems that are inherited from typical software development (cf. Sommerville, 2004). On the other hand, they are facing both technical and social challenges that are unique to groupware development (cf. Borghoff and Schlichter, 2000). These challenges, however, are intertwined in nature. A technically sound solution can fail due to the social issues of the group (Grudin, 1994a). As Grudin (1994a) writes:

> *"Groupware may be resisted if it interferes with the subtle and complex social dynamics that are common to groups" (Grudin, 1994a).*

Similarly, a socially sound solution may not be realisable due to the technical difficulties. Success and failure of groupware are equally hard to predict (Grudin, 1994a). A groupware that works for one group may not work for another. This suggests that the process of developing a groupware is a socio-technical challenge that has to be addressed *in situ*.

Five aspects of the socio-technical challenge

The CSCW research community had already recognized the socio-technical nature of groupware development in the 1990s (cf. Grudin, 1988; 1991a; 1994a; 1994b). Many experiences and lessons had been reported in journals[10], conferences[11], and workshops[12]. I shall not repeat their findings here or attempt to give a comprehensive survey of the literature. Instead, to put this thesis into context, I derive five aspects of the socio-technical challenge, mainly drawing on the work by Grudin (1988; 1991a; 1994a; 1994b), Ellis et al. (1991), and Ackerman (2000). These aspects are discussed below.

The organisational aspect

Grudin (1994a) observed that the benefit that one might receive from using a groupware depends on a number of factors. These include the user's preferences, backgrounds, roles,

[10] E.g. Computer-Supported Cooperative Work, Human-Computer Interaction, Interacting with Computers.

[11] E.g. International Conference on Computer-Supported Cooperative Work (CSCW), European Conference on Computer-Supported Cooperative Work (ECSCW), International Conference on Supporting Group Work (GROUP), International Conference on Computer-Human Interaction (CHI), Nordic Conference on Computer–Human Interaction (NordiCHI).

[12] E.g. Collaboration Researchers International Workshop on Groupware (CRIWG).

and tasks (Grudin, 1994a). There is not necessarily a direct connection between the work, e.g. using the groupware to share information, and the benefit of the work (Grudin, 1994a). For example, in a groupware that provides a shared calendar feature for project management purposes, the benefit from using such feature is greater for a manager of the team, rather than her team members (Grudin, 1994a).

A related issue is the conflict of interest between individuals and groups or among individuals. Conflict of interest is common in organizations (cf. Brooks, 1999). Conflict of interest can become a barrier to achieving critical mass in groupware adoption (as will be topical in discussing the Adoption aspect below) or reduce the efficacy of the groupware. Markus and Connolly (1990) discuss the conflict of interest issue thoroughly, where they call it the *interdependence problem*. Grudin (1994a) refers to it as the *Prisoner's dilemma*. According to Markus and Connolly (1990), there are two types of interdependence in the context of groupware. The first one is *usage interdependence*, which is defined as "a person's ability to use an application in one way depends on another individual's different use of the same application" (Markus and Connolly, 1990). The second one is *payoff interdependence*, which occurs when "one person's use of an application creates costs or benefits (negative or positive externalities) for others who use the application in essentially similar ways" (Markus and Connolly, 1990). While usage interdependence always results in different or asymmetric payoffs to the users, payoff interdependence may result in asymmetrical or symmetrical payoffs (Markus and Connolly, 1990).

The third issue which falls into the organizational aspect is the decision making associated with groupware development. For instance, Grudin (1994a) observed that managers who have the power to make decisions can be falsely informed by their intuition. Grudin (1994a) argues that this can be due to underestimating the down side of the system, or to failure to foresee the difficulties that may be entailed in developing, evaluating, and using the groupware. In addition, "[i]ntuition fails when the intricate dynamics of such situations are not appreciated" (Grudin, 1994a, p.101).

For the first issue in the organisational aspect, incentives to compensate for individual effort (i.e. extra work) in using the groupware system are crucial (Grudin 1988; 1994a; Ackerman,

2000). These incentives can be provided either in the groupware or within the organizational context (Ackerman, 2000). The key idea is to ensure that individuals perceived a potential benefit in using the groupware. For the second issue, negotiation may be needed to resolve such conflicts of interest (Ackerman, 2000). For the third issue, Grudin (1994a) emphasizes the importance of user involvement and democratic decision-making in the development of groupware. For example, paradigms such as Participatory Design may help to fulfill this need.

The social aspect

Ackerman (2000) observes that human social activities are "fluid and nuanced". Ackerman (2000), citing Goffman's work (1971), argues:

> *"people have very nuanced behaviour concerning how and with whom they wish to share information. People are concerned about whether to release this piece of information to that person at this time, and they have very complex understandings of people's views of themselves, the current situation, and the effects of disclosure" (Ackerman, 2000)*

For example, people may resist using a groupware for project management that automatically sends notification message to supervisors and directors when the deadline of a piece of work is reached (Borghoff and Schlichter, 2000). As Grudin (1994a) pointed out, groupware might fail if it disturbs the social processes of the group. It is often hard to develop and to support the level of flexibility and detail of human social activities in computer systems (Ackerman, 2000), because "[the] computer is happiest in a world of explicit, concrete information" (Grudin, 1994a, p.97). Due to this difficulty, it may be easier to allow the social mechanisms to regulate the work which the groupware is supporting (Ackerman, 2000).

Work practices

Although there is usually a standard procedure for work, Grudin (1994a) observed that people quite often do not follow the standard procedure for work. Grudin (1994a) argues that "working to rule" may bring the work to halt, and "A wide range of error handling, exception

handling, and improvisation are characteristic of human activity" (Grudin, 1994a). Sachs (Sachs, 1995) distinguishes two types of work. The first type of work is organizational work, which follows formal organizational procedures and is characterized as explicit and structured (Kaptelinin and Nardi, 2006). The second type is informal (i.e. ad-hoc) and unstructured work, which follows work practices and focuses on conceptual understanding (Kaptelinin and Nardi, 2006). According to Kaptelinin and Nardi (2006), citing Sachs's work (Sachs, 1995), groupware developers need to take both types of work into account if the system is to be used to support work. However, most designs are merely informed by the formal work view (Kaptelinin and Nardi, 2006). Grudin (1994a, p.98) argues that if we follow this path of design, we may require "AI techniques beyond the state of the art … to make the system useful". Grudin (1994a) suggests tailoring as a way to address this problem.

A number of ethnomethodology analyses studied the failure of workflow systems in the 1990s (Kaptelinin and Nardi, 2006). Kaptelinin and Nardi (2006) argue that these failures were due to placing the design focus on optimising the effectiveness and efficiency of the organizational tasks rather than on how people actually work. (Kaptelinin and Nardi, 2006). To design better groupware, researchers have been employing field studies, such as ethnography, to inform their design decisions (cf. Ackerman, 2000). However, even after two decades, CSCW researchers still know little about how people actually work[13].

Evaluation

According to Grudin (Grudin, 1994a), groupware is difficult to evaluate for several reasons:

1. It is not easy to replicate all factors, e.g. social, political, economical, and organisational, that affect the use of groupware in laboratory environment.
2. It is hard to schedule an evaluation due to the number of people involved.
3. It is hard to expose interaction issues which can usually only be discovered over a long period of time.

[13] This observation was made by Christian Heath who gave the concluding remarks in the Colloquium on "Challenging Groupware: Emerging configurations for distributed interaction" on 12 February 2008. The author attended this colloquium (cf. also the blog by Sigfridsson at http://socgsd.blogspot.com/2008/02/colloquium-on-challenging-groupware.html).

4. It is hard to identify the factors that lead to success or failure. For instance, a talented group may be able to adapt a seriously flawed system (Grudin, 1994a). This also makes generalizing from previous experience difficult, if not impossible.

Adoption

For a single-user application, persuading one out of five users to use the product is a big success; but for groupware, only one out of five members of the group using the groupware is a huge disaster (Grudin, 1994a). Unfortunately, the adoption process of groupware varies from group to group. This is because of individual differences (e.g. skills, knowledge, beliefs) and the context in which the group is situated (e.g. organization, social practice, and politics). Strictly speaking, there are no two identical groups.

The efficacy, i.e. usefulness, of the groupware requires a critical mass of members of the group to adopt it (Grudin, 1994a). This critical mass is usually high, i.e. the groupware may fail even when only a few individuals refuse to use it (Grudin, 1994a). A groupware is less likely to gain a critical mass of adoption if it presents social or organizational issues as I discussed above. Other issues such as usability and their physical location of the hardware may also affect the possibility of achieving a critical mass (cf. Grudin, 1994a).

The socio-technical gap

Ackerman (2000) argues that the socio-technical challenge in groupware development is due to a mismatch between the social requirements and the technical feasibility. Ackerman (2000) refers to this mismatch as the *socio-technical gap*, as he writes:

> *"The socio-technical gap is the divide between what we know we must support socially and what we can support technically. Exploring, understanding, and hopefully ameliorating this social-technical gap is the central challenge for CSCW as a field and one of the central problems for HCI. Other areas of computer science dealing with users also face the socio-technical gap, but CSCW, with its emphasis on augmenting social activity, cannot avoid it." (ibid)*

In arguing for the significance of the socio-technical gap, Ackerman (2000) has considered three arguments:

The socio-technical gap was due to groupware developers' habit of ignoring social factors. However, Ackerman (2000) argues that researchers have gradually accepted the existence of the gap and recognized its nature. As Ackerman (2000) observed, "[s]ocial nuance and flexibility were slowly added to all CSCW systems, as the undesirability of being explicit became an assumption with CSCW" (Ackerman, 2000). For example, according to Ackerman (2000), lessons were learnt from The Coordinator (Flores et al., 1988) and Answer Garden (Ackerman and Malone, 1990), and the inflexible access control was improved when Answer Garden 2 was developed (Ackerman and McDonald, 1996).

It is tempting to think that the gap will disappear when new technology becomes available, but Ackerman (2000) argues that this gap is unlikely to go away because there has been no success in the past twenty years or so despite the tremendous effort in attempting to bridge the gap in other areas of research, such as artificial intelligence, information technology, and information science (Ackerman, 2000). Furthermore, Ackerman argues the efficacy of other approaches, such as neural networks, has not been proved. He also argues that the gap may be due to a fundamental problem in the von Neumann architecture on which modern computing is based:

> *"As Hutchins and others have noted (Hutchins, 1995, Clark, 1997) the standard model of the computer over the last thirty years was disembodied, separated from the physical world by ill-defined (if defined) input and output devices. In this view, the von Neumann machine came to be socially inflexible, decontextualized, and explicit. … the existing von Neumann architecture led to programming practices that in turn led to explicit and inflexible systems using data that were far too simplified over the natural world" (Ackerman, 2000).*

The gap could be mended by a co-evolutionary perspective. The rationale behinds the co-evolutionary perspective is that people are more flexible than machines and therefore should able to adapt themselves to the machine efficiently and effectively. Ackerman (2000)

explains that the co-evolutionary perspective can hardly be dismissed because it is hard to see that human culture would not adapt to any technology. However, there is evidence that poorly designed technology can result disasters and inevitably have to be *re-designed* (e.g. the Clayton Tunnel railway disaster in 1861 (cf. Harfield et al., 2005). As the subtitle on the CPSR[14] website: "Technology is driving the future… It is up to us to do the steering".

Nevertheless, there is an implicit assumption in Ackerman's (2000) claim concerning the provenance of the socio-technical gap – that we can know what the social requirements are. Unfortunately, after twenty years of empirical studies, the community has not come up with a well-defined set[15] of social requirements that a groupware must support in order to ensure its success. And, despite Ackerman's (2000) hopes for technological solutions to meet social requirements, he does acknowledge that there will always be a compromise between the ideal "technical working" system and an "organizationally workable" system.

2.2.3 Brooks's "No Silver Bullet" argument

In his seminal paper "No Silver Bullet – Essence and Accidents of Software Engineering", Fred P. Brooks Jr. argues that the difficulties for software development can be divided into the essential and accidental[16] (Brooks, 1987; 1995; Fraser et al., 2007). Brooks (1995) argues that, unless the incidental difficulties are 90% of all the difficulties, spending time in tackling incidental difficulties will not result in significant improvement for the software development process as a whole. According to Brooks (1987), the essential difficulties are *born* with software development, while the accidental difficulties stem from its production process. Brooks (1987; 1995) argues that the hardest part in software development is the *mental crafting* of the conceptual construct, not its implementation process. For Brooks (1987), developing software is always challenging due to its essential complexity, conformity, changeability, and invisibility. Brooks (1987) explains:

---

[14] Computer Professionals for Social Responsibility, http://cpsr.org/

[15] There are new findings about how people work together continuously, but it is also the case that generalisation is difficult, if not impossible.

[16] Brooks explained that the word "accidental" is somehow misleading. He explains: "By *accidental*, I did not mean *occurring by chance*, nor *misfortunate*, but more nearly *incidental*, or *appurtenant*" (italics in original, (Brooks 1995, p.209)). Brooks used the word "incidental" instead of "accidental" in a recent workshop in OOPSLA '07.

Complexity – there are no two parts that are exactly the same beyond the statement level, and complexity scales up nonlinearly in size (Brooks, 1987). However, Brooks (1995) explains that most of the complexity in software is due to the representation involved in its implementation, i.e. what is accidental in Brooks's terms, such as algorithms, connectivity, and data structures.

Conformity – this concern relates to the artificial nature of software and the human agency imposed on it. Whereas physical laws can guide the modelling of natural phenomena, there are no fundamental principles that one can follow when making complex objects in software development (Brooks, 1987). Software developers have to conform to different arbitrary standards from time to time, from systems to systems, not due to physical necessity but due to the fact that these standards are designed by different people (Brooks, 1987). In the other words, *software does not conform to any principles that can be found or can be established in nature*. This makes software engineering different from other engineering disciplines as well.

Changeability – software inevitably changes from time to time because "the software product is a cultural matrix of applications, users, laws, and machine vehicles. These all change continually, and their changes inexorably force change upon the software product" (Brooks, 1987, p.12).

Invisibility – Unlike a floor plan, which can be represented in a geometric reality, software is hard to represent in two or more dimensions (Brooks, 1987; 1995). The reality of software is not in a physical space; one cannot interact with software abstractions as when interacting with other physical objects. Software is inherently invisible and unvisualisable (Brooks, 1987).

Brooks (1987) has considered four strategies for attacking the essential difficulty of software development, i) buying instead of building in-house software, ii) rapid prototyping in supporting requirements elicitation and refinement, iii) growing instead of building software, iv) investing in great designers. Much has changed in the software industry since the publication of Brooks's (1987) seminal paper, e.g. the propagation of object-oriented methods and tools, the advocacy of agile development paradigms. Even so, despite the fact

that more than 20 years has passed and the responses of Brooks's critics[17], e.g. (Harel, 1992) and (Cox, 1990), the author argues that Brooks's (1987) argument concerning the nature of software development still holds. Indeed, these essential difficulties constitute part[18] of, in Brooks' terms, the *essential* difficulty in designing a groupware for supporting a group because groupware is one of the species of software. Of particular relevance amongst Brooks's (1987) recommendations for tackling the essential difficulties of software development is the notion of growing software. I shall argue that groupware should be *grown organically* within its production environment. That is, it should be developed by those who use the groupware in the actual environment in which the actual group work is carried out. I will elaborate how this can be done through an Empirical Modelling approach in chapter 7.

2.3 Contemporary groupware development paradigm

This section describes the contemporary approach to groupware development. This includes Tang's (1991) iterative process for studying and supporting cooperative work (cf. §2.3.1), the ethnography-oriented design tradition since 1990s (cf. §2.3.2), and the recent call for increasing the level of user participation, e.g. through Participatory Design (cf. §2.3.3).

2.3.1 The iterative process

Tang (1991) described an iterative approach for studying cooperative work (see figure 2.1)[19]. This approach consists of three stages, i) observe, ii) understand, iii) support. Based on the description in Tang (1991), Borghoff and Schlichter (2000), and other reports in the literature,

[17] In his response to Harel's (1992) questioning of the need to make a distinction between the essential and the accidental, Brooks (1995) argues that it is necessary and "[it] is absolutely central to understanding why software [construction] is hard" (Brooks 1995, p.214).

[18] The other part of the essential difficulty is the socio-technical challenge, which was discussed in section 2.2.1

[19] Initially, it was a research approach for studying collaborative work. John Tang uses his iterative approach to study and develop collaborative drawing tool support for a small group of designers in connection with his doctoral research (cf. Greenburg 2008). While this approach may look over-simplified in the light of other more sophisticated software engineering methodologies, it became popular in the 1990s and most of the groupware development projects nowadays more or less follow Tang's approach. i.e. observe, understand, and support. Greenburg (2008) credits Tang's contribution retrospectively, and refers to Tang's approach as "group-centered" design.

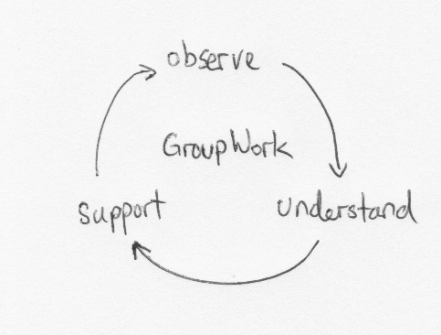

Figure 2.1 – Tang's iterative process for groupware development[20]

these three stages can be characterised as:

i. Observing how people carry out the work – this may involve field studies.

ii. Understanding how people carry out the work – if groupware has been deployed, analyze and evaluate how the group adopts the groupware in its context.

iii. Developing computer tool support – i.e. groupware, for the group, based on the analysis and evaluation.

However, there is no predefined entry point for this approach (Borghoff and Schlichter, 2000). For example, one development may begin with an observation on how the group carries out the work, while another may start from building computer tool support for the group and observe its adoption later. This development process may go through various cycles before it can achieve a satisfactory groupware to support the group (Dourish, 1995). Similar iterative approaches have also been discussed elsewhere, e.g. (Boehm, 1986), in the context of systems development.

2.3.2 Ethnomethodology and Ethnography

Despite the merits of an iterative approach, such as is depicted in figure 2.1, more informative techniques for studying the workspace for groupware are required. Ethnomethodology (Garfinkel, 1967) and ethnography have been gradually adapted in groupware development and other systems designs since 1990s (cf. Anderson, 1994;

[20] This figure is adapted from (Tang, 1991)

Dourish and Button, 1998; Dourish 2001, 2006; Harper 2000). As Dourish and Button (1998) have observed, the use of ethnography became popular for studying workspaces in HCI and CSCW after Lucy Suchman's (1987) work – *Plans and Situated Actions*. It is worth noting that ethnomethodology is not the same as ethnography. While ethnography is a technique that anthropologists use to inquire into human societies (Crabtree 2003; Dourish 2001), ethnomethodology is "a particular analytic orientation to the practical issue of the problem of social order" (Dourish and Button, 1998). Ethnomethodology is often perceived as the same as ethnography because ethnomethodologists often use ethnographical data in their studies, and their differences can only be detected in their "analytic mentality" (Dourish and Button, 1998). Indeed, other social scientists, e.g. activity theorists, may use ethnography as a data collection technique (Crabtree, 2003).

The data generated by an ethnographical investigation can be characterized as situated, qualitative, and open-ended (Dourish, 2006). This is because ethnographical investigation pays particular attention to people as the subject of study, and produces an account – from the investigator's perspective and arguably on a subjective basis – of how people work. The virtue of ethnography is that it not only guides the observation, but it also provides a framework for evaluating and understanding the group work in its context. In relation to, the iterative approach described in section 2.3.1 (also cf. figure 2.1), ethnography naturally fits into the observation stage. For instance, according to Hughes et al. (Hughes et al., 1995), ethnography can play the role of:

i. A 'quick and dirty' technique to get groupware developers to achieve preliminary understanding of the workspace in a relatively short time frame.

ii. A concurrent investigation which is carried out in parallel with the technical development and informs design decisions periodically.

iii. An evaluation technique to study the groupware prototype.

In reality, ethnographical investigations are usually carried out in parallel with the development process (Borghoff and Schlichter, 2000) and it becomes a division of labour (Dourish and Button, 1998). Since ethnographers are typically social scientists and are usually not involved in the groupware implementation at all, the only point of contact

between ethnographers and groupware developers is in team meetings at which ethnographers report their findings to the groupware developers, or when ethnographers formulate their findings into some form of requirements. What is more, the ethnographers are usually seen, not as people directly involved in the work process, but as the *spokesman* for the workers (Hughes et al., 1994). In this case, ethnography is arguably not part of the groupware development process at all (Dourish and Button, 1998). The conflict between the ethnographers and the systems developers reflects the divide between two disciplines, social science and computer science, and leads to what Grudin and Grinter (1995) have called "the ethnographer's dilemma"[21]. Dourish (Dourish, 2006) argues that this conflict is due to the lack of a fundamental connection between them[22].

Apart from the practical difficulties, the use of ethnography in groupware development has invited other criticism. Harper (2000) makes the criticism that some researchers are using the term as a buzzword to denote close observation research projects and that how these inquiries were structured was usually unspecified. The descriptive nature of ethnography has also invited criticism. For instance, Kaptelinin and Nardi (2006) criticize the heavy use of text as the main medium for reporting ethnographic studies. They argue that this makes it hard to generalize, and therefore diminishes its value when compared to other theoretical approaches to interactive systems design[23]. There are complaints that ethnography is not generating the "right materials for design" (Harper, 2000). Moreover, ethnography cannot capture the users' crafting skills and their reflection on the work. In discussing an ethnographic study of air-traffic control, Shapiro (1994) reports that the ethnographers failed to reveal the organisational arrangements for air-traffic controllers and the personal attributes that might have affected their work. In the end, there is simply no firm definition of

[21] Dourish and Button (1998) further break down the *ethnographer's dilemma* into the *paradox of system design* and the *paradox of technomethodology*.

[22] Dourish and Button (1998) have a proposal – technomethodology – of how ethnography and systems design may be connected together at the analytic level, instead of at the empirical level (i.e. 'implications for design' (Dourish 2006). Discussion of Dourish and Button's proposal is beyond the scope of this thesis. For the interested reader, the author recommends reading (Dourish and Button 1998), (Dourish 2001, ch. 3), and (Dourish 2006).

[23] Kaptelinin and Nardi (2006) also criticized Garfinkel's position on ethnomethodology as "radical antitheory", which "avoids generalization and abstraction" (Kaptelinin and Nardi 2006, p.17).

what ethnography *is*, particularly in CSCW (Harper, 2000).

2.3.3 Participatory design

Participatory Design (PD)[24] is a collection of principles and practices that is primarily aimed at making computer technologies and social institutions more responsive to human needs (PDC, 2008). In fact, PD has been applied outside computer systems development, e.g. in engineering projects (Grønbaek et al., 1993). Traditional PD practices only cover the design and planning activities before procurement (e.g. Bødker et al., 2004), but modern PD practices have broadened to cover most activities in the entire development process, from early project planning to systems evaluation (cf. Muller, 2001)[25]. PD has become popular in systems development since 1990s. This is due to its promise of active and direct user participation (cf. Bødker et al., 2004). That is, the workers who will eventually use the computer system should be involved in its design process and the decision-making that may affect its workspace (Greenbaum, 1993). This strong commitment is sometimes deemed to be bringing democracy and respect to the workspace and to systems development (cf. Kensing and Blomberg, 1998). The strong commitment in PD has a root in 1970s, where the workers and their unions feared that the introduction of computer technologies would eventually lead to job losses (Kensing and Blomberg, 1998). It was conceived as a political strategy of systems design – a strategy to rebalance the power between the workers and the management (Kensing and Blomberg, 1998).

In the context of groupware development, user participation is seen as a remedy to overcome part of the socio-technical challenge (cf. Mambrey and Pipek, 1999). Karasti (2001) reports a propitious experience of integrating ethnography into PD through, what he called, *appreciative intervention* in a workshop[26] in relation to the development of a teleradiology system. In the workshop, ethnographers, developers, and the workers carried

[24] The detailed history of Participatory Design, particularly the politics involved in introducing computer technologies into the workspace, is beyond the scope of this thesis. See (Kensing and Blomberg 1998), and (Schuler and Namioka 1993).

[25] In fact, there is no standard answer to the question "What is PD?" (cf. Törpel2005).

[26] A Participatory Design technique that is used to co-construct the future work practices with the workers.

out sustained analysis of the workers' everyday practice through pre-recorded video clips. According to Karasti (2001), the main benefits of such approach are:

i. The overall systems development is oriented towards the actual work practices.

ii. The user participation has informed and is intertwined with the systems development, evaluation, and envisionment.

iii. The tension between the developers and the workers is lessened.

iv. The developers were made aware of what is important and must be preserved in the workspace.

Recently, Pekkola et al. (2006) proposed a participatory iterative approach for information systems development[27] based on their experience of groupware development projects. Their participatory iterative approach begins with project definition and has eight stages. After the initial requirements elicitation, they follow an iterative process for prototyping (i.e. analysis and design, implementation, deployment), gather user feedback, and re-analyse the requirements. Although Pekkola et al. (2006) have emphasized user participation in their development process, their approach does not make use of some of the important features of PD, e.g. joint decision-making in the design, future workshops. It also omits the ethnographical investigation that would be included in most groupware development. In this approach, the authors hope to make a blend between PD, an iterative approach, and an ethnographical approach. However, I argue that this blend is not smooth. If PD is intended to be the potential key to tackling the socio-technical challenge, it requires integration of a kind closer to what Karasti (2001) has used in the teleradiology project. Nevertheless, I am going to study this *participation problem* in groupware development in depth in chapter 3, as it is central to the theme of this thesis.

2.4 Accommodating evolving human activity

In this section, I discuss the evolving nature of human activity in groupware development

[27] Compared with CSCW and PD, research into Information Systems Development is more organisation-oriented and its development process is not flexible enough to cope with changing context (cf. Pekkola et al. 2006).

and review approaches and strategies that have been used to make the groupware "softer" to cope with this evolution.

2.4.1 Evolving nature of human activity

There is no doubt that humans are living in an ever-changing world. To adapt ourselves to this ever-changing world, our activities evolve from time to time. In fact, we often change the *activity* to adapt to the situation in which the activity is carried out (Suchman, 1987). As Béguin and Clot (2004) pointed out "activity cannot be reduced to execution procedures applied more or less passively … even the most repetitive movement of a production line worker is always unique" (ibid, p.57). Think of the activity of enjoying a cup of coffee in a café. There may be a limited repertoire of actions that can be carried out in this simple activity, e.g. buying a cup of coffee from the till, sweetening the coffee, finding a seat, reading a book. However, the occurrence of and the sequence of the actions and operations that constitute the activity can be quite different[28]. For example, one may find a seat before buying a cup of coffee from the till, add more milk because there is no more sugar left, read a the same book but start a different page, and so forth. In this example, although the overall organisation of activity is more or less the same, the *detail* in the activity is always different and its actions are always *unique*. This is because actions are situated in the context of the activity (cf. Suchman, 1987). Béguin and Clot (2004) also remind us that, in other circumstances, the organisation of an activity *can* be changed:

> *"Obstacles, disagreements [and] conflicts encountered in activity generate tension ... and invite the subject to mobilize and develop [new forms of organization]" (quoted in Kaptelinin and Nardi, 2006, p.220)*

Situated actions (Suchman, 1987), in fact, are not only limited to everyday life activities – they can also be found in the use of computer technologies. For example, there is some flexibility about the sequence of actions in the activity of sending an email to a co-worker, as long as the send button is clicked in a graphical interface of an email application. In fact, we

[28] From an Activity Theory perspective, an activity is composed of actions, and actions are composed of operations (cf. Kaptelinin and Nardi 2006).

use computer technologies in whatever way is appropriate to our needs, rather than necessarily respecting their *designed* usage. From the systems developer perspective, it is an unanticipated use. Indeed, the level of unanticipated use of computer systems has surprised many systems developers and this has resulted in an emerging research topic in HCI[29].

2.4.2 Evolution of groupware

Groupware cannot be developed and remain *as is* if its aim is to support the evolving nature of human activity. Many concepts and design principles have been developed in the literature to promote software flexibility. Certainly, groupware developers may apply these concepts and design principles to make groupware cope with the evolving nature of our activities. Although the aims of these concepts are similar, I argue that there are subtle differences among these concepts and techniques in the literature in relation to their scope of concern and intervention. I discuss these concepts briefly.

Extension

The idea of designing extensible software is not new. This idea can be traced back to the 1970s when the software industry faced the challenge of changing requirements after the software was deployed. Parnas (1978) reports four common issues[30] in the software systems at that time which led to difficulties in extending or tailoring the software for different needs. Parnas (1978) asked us to consider software as a set of components ("software as a family of programs" in Parnas's terms) and to identify a minimal set of components ("minimal subset" in Parnas's terms) for basic usage, such that additional functionality can be derived from this minimal set.

In view of subsequent development, the idea of making software extensible is closely related to the idea of making software reusable. That is, the software extension which is built on the base system, which in turn can be seen as being "reused". The notion of extension is the

[29] There is a tension between 'curriculum for use' and 'unanticipated use' (Bertelsen, 2006).

[30] Three out of four were related to modularity in software architecture. Unsurprisingly, a lot of novel ideas in this paper have been taken up in structured programming, and in object-oriented programming.

simplest concept to cope with changing requirements. Nowadays, quite a number of applications, e.g. Firefox and Adobe Photoshop, support additional features through plug-ins, i.e. an extension component that can be plugged into the application for additional features. In the context of groupware, this concept of extensibility was exemplified by available architectures, e.g. (Gibbs, 1989; Hummes, 2000). However, this concept can only be used to add new features, but not to rectify major faults in the existing software[31].

Tailoring and Customization

Tailoring can be viewed as the "further development of an application during use to adapt it to needs that were not accounted for in the original design" (Mørch et al., 1998). As Stiemerling et al. (1997) put it, "[t]ailorability is a property of software which allows to change certain aspects of the software in order to meet different user requirements". Another similar concept to tailoring is *customization* (Mackay, 1990). Some researchers view customization as the same as tailoring, e.g. (Wulf, 1999). Others distinguish customization from tailoring. For instance, Bentley and Dourish (1995) see the customization process as consisting of tailoring activities. The concept of tailoring goes beyond the concept of extension. Tailorable software must have an extensible architecture underneath[32], and the flexibility to fulfill the dynamically evolving requirements by adding new features or changing the behaviour of the software is provided to the end-users without rewriting or recompiling the software (Stiemerling et al., 1997). That is, the *tweaking* of the software is shifted to the end-user from the developer. According to Stiemerling et al. (1997), this shift is necessary because "the continuous involvement of systems developers may retard or even impede a necessary adaptation of the software."

In the context of groupware development, tailorability is often seen as a mandatory feature of groupware due to the rapidly changing context in organisational work (Mørch et al., 1998).

---

[31] This is because the base system is assumed to remain unchanged. While the extension components can be added anytime in the use context, the extensible architecture is designed and developed in the main development process. In a highly modularized operating system, e.g. Linux, it is possible to replace a faulty component without redeveloping the entire system. By definition, this form of component replacement is *patching*, not *extending*, the existing software.

[32] Technically, this is often component-based, e.g. (Stiemerling 1997, Stiemerling and Cremers 1998)

In a workshop on tailorable groupware, Mørch et al. (1998) identify four levels of tailoring activities:

i. Changing parameters of the application.

ii. Changing the composition of the application.

iii. Extending the application with high-level computational mechanisms, e.g. APIs.

iv. Modifying the source code of the application.

Tailoring at the third and the fourth level in the above classification is sometimes viewed as end-user programming (Nardi and Miller, 1990; Nardi, 1993). Arguably, the fourth level of tailoring activities, i.e. modifying the source code of the application, has gone beyond the boundary of "tailoring". I argue that such activity is more appropriately referred to as re-development or maintenance of the application, depending on the nature of such modification. A similar perspective is adopted in (Teege et al., 1999).

Tailoring may also be a recipe for achieving a critical mass of adoption of the groupware, as it allows different members of a group to customize the groupware to fulfill their different personal needs (Bentley and Dourish, 1995). However, as Teege et al. (1999) pointed out, tailoring may redistribute the work and the benefit in a group – an issue that I have discussed in the organisational aspect of the socio-technical challenge (section 2.3.2). In a study of tailorability in a visualization system, Tim Mansfield reported that users might not fancy using advanced tailoring features of an application due to psychological barriers (Mørch et al., 1998).

Appropriation, Adoption, and Adaptation

From a theoretical perspective, the concept of *appropriation* in interactive systems design was borrowed from Adaptive Structuration Theory (Andriessen et al., 2003; Dourish, 2003). From the AST perspective, appropriation concerns the mode or fashion in which the group uses, adapts, and adopts technology in relation to its *structural features* and the *spirit* of

those features[33] (Andriessen et al., 2003; Poole and DeSanctis, 1990). However, Andriessen et al. (2003) argue that framing the concept of appropriation from a social-scientific perspective "does not cover the design of customizable or tailorable artifacts which may let use evolve into co-construction."

From a more practical perspective, the concept of appropriation is similar to the concept of customization and tailoring, but it has a broader scope which concerns how people adopt and adapt technology and integrate the customised technology into their context through continue (re-)configuration of the technology. As Dourish (2003) put it:

> *"Appropriation is the way in which technologies are adopted, adapted and incorporated into working practice. This might involve customisation in the traditional sense (that is, the explicit reconfiguration of the technology in order to suit local needs), but it might also simply involve making use of the technology for purposes beyond those for which it was originally designed, or to serve new ends." [ibid, p.463]*

Within this broadened scope, appropriation becomes an inevitable process for groupware adoption and adaptation. As Dourish (2003) pointed out, the success of appropriation not only depends on the flexibility of the technology but also the social practice in which the appropriation takes place. Indeed, adoption and adaptation are determinant factors for the success of failure of a groupware (Grudin, 1994b; Palen, 1997). Dourish (2003) asks us to view the appropriation of technology in relation to communities of practice, a context in which technology plays important roles. To paraphrase:

i. The features of the social-technological system become meaningful to people as they develop understandings about how the representations the social-technological system offer may affect their work and how the representations

[33] Andriseen et al. (2003) write "The use of advanced information technology depends on the features and spirit of the tool, but also on structures provided by the task and the environment. Moreover, when a technology is used, its output – e.g. proposals, new ideas, but also emerging ways of using the technology in interaction – becomes in turn a new source of structures. This is the process of 'structuration', i.e. the production (and reproduction) of emergent social structures. When emergent rules or behaviours are accepted over time they become institutionalised and will determine subsequent behaviour. This process of institutionalising meanings and rules in a group is called 'appropriation'".

relate to other people or entities.

ii. The social-technological system provides a means for people to can interpret and understand others' actions, both implicit and explicit.

Evolution

Evolution is not another buzzword to describe the process of adoption or adaptation of groupware. Instead, it is a perspective at the holistic level that emphasizes a focus on the evolutionary process of groupware rather than giving exclusive attention to the technological imperative perspective or the organisational imperative perspective (cf. Andriessen et al., 2003).

Evolution can be characterized by its emphasis on the gradual and mutual adjustment, both in social and technical aspects, instead of sudden changes which may deliberately be imposed by other factors, e.g. politics in the organization, rather than the actual need derived from day to day work practice (cf. Andriessen et al., 2003). This perspective is compatible with many existing concepts, e.g. adoption, tailoring, customization, incorporation, and appropriation (Andriessen et al., 2003). However, it contradicts Orlikowski and Hofman's (1997) improvisational model for change management for groupware which suggests anticipated changes interleaved with emergent and opportunity changes[34].

Since the evolution of the groupware occurs in a context in which the social, the organization, and individual aspects are all evolving and affecting each other, it has therefore sometimes been viewed as a co-evolution process (Rogers, 1994). I will discuss the co-evolution in more detail in relation to collaborative modelling in Chapter 5.

[34] Orlikowski and Hofman's (1997, p.13) describes anticipated changes as "changes that are planned ahead of time and occur as intended", emergent changes as "changes that arise spontaneously from local innovation and that are not originally anticipated or intended", and opportunity-based changes as "changes that are not anticipated ahead of time but are introduced purposefully and intentionally during the change process in response to an unexpected opportunity, event, or breakdown".

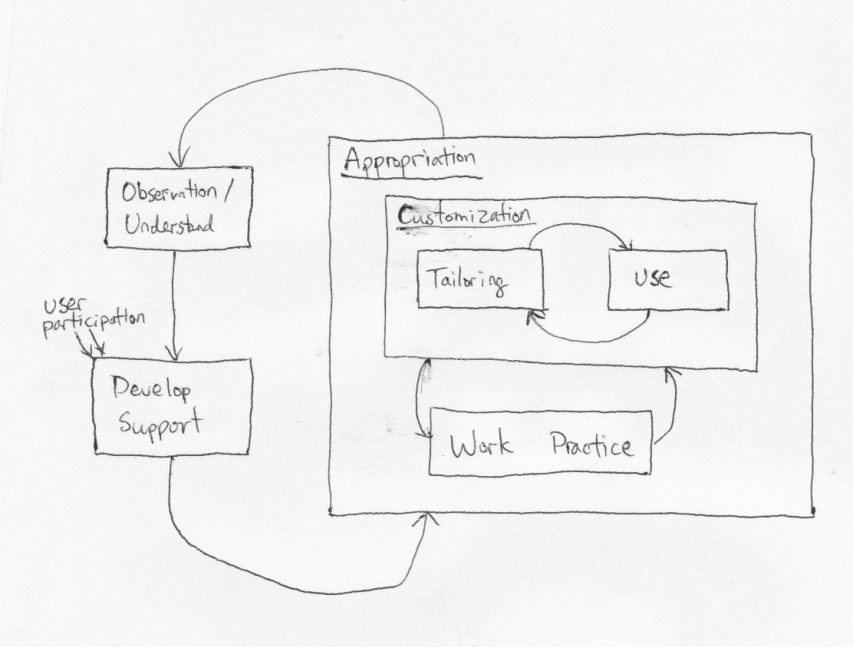

Figure 2.2 – A model for groupware evolution

A model for groupware evolution

To help the reader to understand what I meant by groupware evolution in relation to the existing concepts in the literature and in the context of this thesis, I have developed a simple model which integrates all the aforementioned concepts, i.e. tailoring, customization, appropriation, adoption and adaptation, into a coherent model. This model takes inspiration from the iterative process to groupware development proposed by Tang (1991) that I discussed in section 2.3.1, the evolutionary process described in (Wulf and Rohde, 1995), and the STEPS process model (Floyd et al., 1989).

Figure 2.2 illustrates how these concepts can be put together and shows the scope of their concerns. Individual members of the group can tailor the groupware to her local context of use. However, the local context of use may change from time to time, therefore may require further tailoring to suit the evolving needs. All these tailoring activities are circumscribed within the customization process[35]. Individuals can customize the groupware according to the communities of practice to which they belong. As in tailoring, there is a continuous

[35] MacKay (1990, p.23) asserts that "Customizations are preserved in a form accessible to users, which may be shared". This suggests that MacKay considers customization to be an individual process rather than a collective process.

interplay between customization and the working practices. According to MacKay (1990), individuals' customizations can be shared and therefore can be viewed as a collaborative activity within the organization. All the customizations take place within the context of appropriation, and the appropriation is carried out at the group level.

By integrating all these concepts into a coherent framework, I do not intend to disregard or be disrespectful of the theoretical grounds or traditions behind accepted concepts, such as Adaptive Structuration Theory behind the concept of appropriation.

2.5 Towards efficacious groupware development

So far in this chapter, I have studied the challenges in groupware development – i.e. why groupware is so difficult to develop compared to other species of software. I have also discussed a number of concepts and techniques that attempt to make groupware easier to adapt to the human context and enable it to cope with the evolving nature of group work – i.e. the changing context of group work in different aspects, which includes the social, the organizational, and the work practices.

In the final section of this chapter, I argue that an approach to groupware development that is *efficacious* must adopt a human-centred approach[36]. Section 2.5.1 discusses respects in which current approaches to groupware development fail to be human-centred. In section 2.5.2, I then critique the notion of participation in the current participatory approaches to systems development. I use the term "efficacious" here in the sense in which it is used by the French philosopher François Jullien. Jullien (1995) adopts the term in relation to his lengthy discussion of the interpretations of the Chinese word *shi* (勢) as *efficacious disposition* in his inquiry into ancient Chinese philosophy. I will briefly discuss this concept in section 2.5.3.

[36] By emphasising a human-centred approach, I do not mean to discard the group-centred approach to groupware development that we discussed in section 2.3. Instead, I regard human-centred thinking as transcending group-centred thinking.

2.5.1 From group-centred to human-centred for groups

As I discussed in section 2.3, there is a strong connection between the development of user-centred design and the development of the group-centred approach to groupware development. Research into user-centred design in HCI shares many common themes with research into groupware development, e.g. social aspect and the appropriation process. To some extent, the group-centred approach addressed many concerns regarding the limited scope of the notion of *user interface* that the first generation HCI did not take into account (cf. Grudin, 1990; Bannon, 1992b). It was in this context that the research field CSCW received much attention.

The major difference between user-centred design and the group-centred approach is their scope – the group-centred approach pays particular attention to the interaction between group members. In this sense, groupware development may follow an expanded-scope user-centred paradigm. More importantly, current group-centred approaches may share a common perspective on *user* as the term is interpreted in user-centred design. This perspective puts it emphasis on *a human as a technology user in a particular domain* rather than *a human who has a felt life* (cf. Gasson, 2003; Bødker, 2006; McCarthy and Wright, 2004). This narrower perspective precludes developers examining the issues that are central to human-centred design (Gasson, 2003). According to Gasson (2003), human-centred design promotes the following features:

1. Flexible computer technologies that cope with people and allow them to manage and to shape their work.
2. Priority given to human communication and collaboration over computer-based information processing, which is the opposite of the traditional technology-oriented approach.
3. Computer technologies that allow the use of implicit knowledge, i.e. tacit and skill-based, together with the explicit, e.g. rule-based, knowledge. If the computer technologies merely support explicit knowledge and not its implicit counterparts, the implicit knowledge becomes marginalised because the explicit knowledge is no longer contextualised (Gasson, 2003).

4. The design of computer technologies that address the normative expectation of the technology (cf. Kuhn, 1996). Computer technologies are intertwined with the social expectations that circumscribe the design of the technologies.

5. Designs that gives joint consideration to the questions of "What is technically feasible?" and "What is socially desirable?"[37] (cf. Kuhn, 1996).

6. Socio-technical computer technologies that support meaningful and enriched work (cf. Gill, 1991). This precludes the separation of the planning tasks from the doing tasks, since such separation may deskill those workers who are ill-equipped for meaningful decision making or exception handling (cf. Cooley, 1987).

Approach	Intended Focus	Actual Focus
Traditional IS design approaches	The structuring of ill-structured problems: goal-driven decomposition.	Explicit (management) focus is goal-driven and decompositional. Implicit strategies are opportunistic, to deal with goal-emergence.
Prototyping and participatory design	Negotiation and exploration of ill-structured problems.	Iterative and cyclical process of stakeholder involvement, limited by political selection of user-representatives and technology-centered requirements focus.
Interaction design	Exploration of IT-supported user work-processes.	Technology-centered, individual user focus. Assumes consensus among system users, with well-understood IS goals.
UML and Use-Cases	Modeling of business processes and user-interactions with intended IT system.	Models formal information-processing (business processing rules). Technology-centered and decompositional (so no opportunity to redefine goals as these emerge through design process).
Agile Software Development	Adaptation of an evolving system design, based on user interaction and scenario generation.	Technology centered prototyping, accomplished by the development of individual user-scenarios.

Table 2.3 – Summary of the intended focus and actual focus of contemporary systems development approaches[38]

[37] The socio-technical gap, as described by Ackerman (and as discussed in §2.3.2), points to a similar but larger divide. Instead of “what is socially desirable?”, Ackerman asks “what must be supported socially?”.

[38] This table is adopted from (Gasson 2003)

Based on the above principles, Gasson (2003) examined five contemporary systems development approaches[39] from an Information Systems Development (ISD) perspective, and concluded that contemporary systems development approaches put too much emphasis on the technology-focused perspective and do not satisfy human-centred design principles fully (cf. table 2.3). Gasson (2003) argues that:

> *"… each of these approaches focuses on user-centeredness at the expense of human-centeredness, because of their technology-centered focus … system stakeholders -- the intended "victims and beneficiaries" of the proposed information system (Checkland, 1981) – should be enabled to negotiate the role and purposes of the system with other stakeholders, non-technical as well as technical." (Gasson, 2003, p.39)*

As Gasson (2003) pointed out, this idea is not new – it was part of the early movement of PD (cf. Mumford, 1983). However, it has been difficult to implement due to the goal-driven and technological-focused traditions in ISD (Gasson, 2003). Gasson (2003) further argues that:

> *"most user-centered approaches are concerned with closing down a technology-centered and goal-directed IS problem-definition, not about exposing (or opening up) the social and organizational context (the design "problem") to examination and debate … The most human-centered of the methods discussed, participatory design approaches, do not change the fundamental nature of the "circular" system development life-cycle. They merely "rotate" the life-cycle through 90°, so that the cycle is driven by user-evaluation of system design requirements, rather than by technical evaluation of system design requirements. This rotation does not question many of the essential contradictions of the traditional perspective, because it inherits the "problem closure" life-cycle emphasis." (ibid, p.39)*

[39] These include Participatory Design, two varieties of agile development (Adaptive Software Development and Extreme Programming), use cases in UML, and the "Interaction Design" paradigm in HCI. However, Gasson (2003) did not mention the group-centred approach that we discussed in section 2.3.

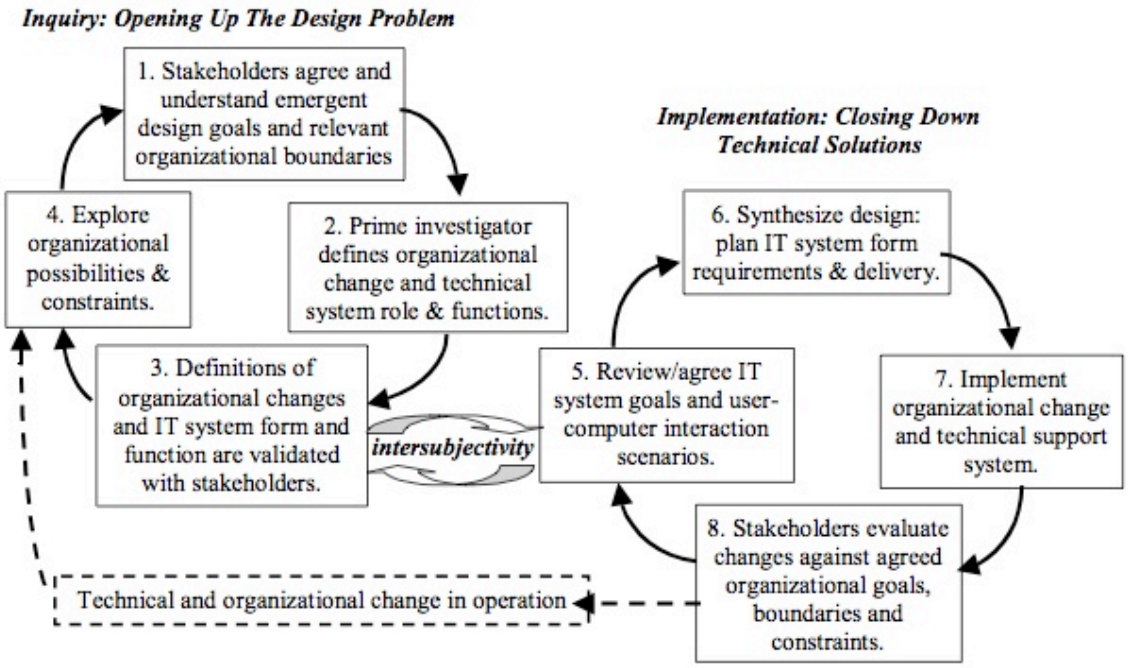

Figure 2.4 – Gasson's dual-process model[40]

To satisfy the human-centred concerns, Gasson (2003) proposed a dual-process model (cf. figure 2.4), which is based on a case study of the systems development in an engineering firm. Gasson (2003) claims that the model "represents an optimal way of managing the dialectic between subjective, organizational problem inquiry and goal-directed, process and technical solution design." The two cycles, "opening up design problem" and "narrowing down potential solutions" can be carried out inter-changeably to cater for design inquiry and technical implementation. In contrast to the traditional waterfall model for ISD, Gasson's (2003) dual-process model fills in the missing pieces of the "systems development puzzle" with the "opening up" cycle (on the left) that is aimed at guiding the "pre-requirements" activities. However, I argue that the dual-process model merely represented the system development process of the engineering company where the study was carried out, and, at best describes a process that caters for, addresses, and balances the concerns raised from both the user's and the developer's worlds. It hardly convinces one to believe that this model can address the human-centred concerns or be applicable to other forms of systems development, such as groupware development. From my perspective, the main concerns

[40] This figure is adopted from (Gasson1998)

with Gasson's proposal are:

i. In stage 2 of the "opening up" cycle, the prototype of the organisational and technological change is produced by the principal investigator. Although other stakeholders may have been consulted in the first instance (as depicted in figure 2.4), the orientation of this "opening up design problem" process suggests that the inter-subjective design perspective is vulnerable to being overridden from a managerial perspective. This is somehow contrary to the principles of PD and Gasson's initial argument for human-centredness.

ii. The non-technical stakeholders, e.g. workers and their managers, can only discuss or review the systems implementation at the transition stage (stage 3 to stage 5 and *vice versa*). It is also the only point of contact between the technical and non-technical teams. Further, the discussion and review around the design goals and boundary between the technical and non-technical teams is likely to be carried out at managerial level in practice. The dual-process does not pre-empt such "representative meetings".

iii. The technical development process seems to be hidden out from the scene. It more or less follows a mini-waterfall model, i.e. the development process looks like a black box process. The dual-process model seems to promote minimal contact between the developers and the non-technical stakeholders. It does not promote an ongoing interaction between these two parties, which a human-centred approach should promote.

To some extent, the CSCW and groupware development research community has gradually taken the aforementioned human-centred concerns into consideration over the last twenty years. However, as Gasson (2003) observed in the IS and HCI literature, the CSCW literature rarely discusses how these concerns are considered, especially in a holistic view at the methodology level, during the development of groupware. Experience from field studies usually reports the scattered results but does not report how they actually made it work.

The problem with the group-centred approach, or user-centred approach in general, is not

with its promises, its philosophy, or its intended outcome. The *real* problem is the over-emphasis on the organisational and technical goals rather than the emphasis on each stakeholder's *felt-life* (cf. McCarthy and Wright, 2004). Because of this misaligned emphasis, many so-called user-centred, group-centred, and human-centred approaches fail to live up to their billing. If these approaches cannot fulfil their promise, they may become no more than labels for research funding.

Although Gasson's dual-process model potentially violates some basic tenets of human-centred design, there is little doubt that the model has opened a debate about whether the "pre-requirements" activities (in the traditional ISD sense) should be considered as part of the systems development. Indeed, the human-centredness considerations (especially the bridging between the technical and the non-technical worlds) behind Gasson's dual-process model are worth reiterating, and group-centred approaches should learn from these principles. In the next section, I discuss the notion of participation in relation to the human-centredness concern.

2.5.2 The notion of participation

User participation is thought to be a remedy for relieving the tensions between stakeholders (e.g. developers, workers, and managers) of the computing technologies, and for guiding the development towards a successful direction. It is suggested that user participation can increase the democracy of decision-making during the development process (Wulf and Rohde, 1995) and it can improve the relationship and the mutual understandings between stakeholders (Karasti, 2001; Béguin, 2003; Cluts, 2003; DePaula, 2004). From the developer's perspective, this increases the likelihood of getting the user to accept unanticipated changes, when that happens, during the systems development. From the user's perspective, this results in a 'better system' because of their participation (Pekkola et al., 2006). Although early studies (e.g. Cavaye, 1995; Olson and Ives, 1981) argued that there was insufficient evidence to establish a positive correlation between user participation and system success, recent studies do show a positive correlation between user participation and the likelihood of system acceptance and adoption, both of the design and the actual system (Woods, 2002; He and King, 2008), which is a critical factor in determining

the success and failure of groupware development (Grudin, 1994a). Indeed, user participation is a key component in the principles of human-centred design (cf. Gasson, 2003). However, user participation alone may not be enough to ensure the success of a systems development project (He and King, 2008). Some researchers even suggest that user participation can increase the complexity of managing the systems development project, and may result in its abandonment if user participation is not properly managed (Gasson, 1999; Howcroft and Wilson, 2003). Howcroft and Wilson (2003) argue that, in the worst case, the systems development process becomes an arena for negotiating private interests. For instance, it is possible that users abuse the participatory opportunity to realise crazy ideas or want to explore ideas so ambitious as to be outside the scope of the systems development. For this reason, some researchers have suggested that the intervention of the users is problematic and should be avoided in systems development (cf. Howcroft and Wilson, 2003). Despite the concern that user involvement may invite political conflicts among stakeholders, it is generally believed that user involvement may increase the overall satisfaction of systems development.

The degree of *user participation* varies in different styles of systems development, and in different projects: from 'participation' merely in requirement analysis, to making joint design decisions, to joint appropriation of the system, to end-user development. It is surprising that research into user participation has proliferated so rapidly without agreement on a precise definition of the concept. Perhaps the most satisfactory *definition* of user participation is given by Barki and Hartwick (1989), in which the authors distinguish *user participation* from *user involvement* in systems development. Barki and Hartwick (1989) suggest that *user participation* should be used to refer "the behaviors and activities that the target users or their representatives perform in the systems development process" (ibid, p.59) and *user involvement* should be used to refer to "a subjective psychological state of the individual and defined as the importance and personal relevance that users attach either to a particular system or to IS in general, depending on the users' focus" (ibid, p.59-60). With this distinction between two concepts, it is possible that a user is participating but does not necessarily feel involved in the systems development process, no matter whether the participation is voluntary or forced (Barki and Hartwick, 1989).

In order to get the users to enrol in the development process, systems developers often emphasize its human-centred aspect (Howcroft and Wilson, 2003). Howcroft and Wilson (2003) write:

> *"The promise of empowerment is to give employees more power to use their judgement and discretion in their work, thereby encouraging them to utilise their skills and experience for the benefit of the organisation." (ibid, p.9)*

Genuine participation implies a degree of democratisation by way of equal opportunities and responsibility. In the context of systems development, this implies high degree of joint ownership and joint responsibility for the system and its development process among all stakeholders. However, this is easier said than done. As Howcroft and Wilson (2003) write:

> *"User participation, because it brings together a broad range of actors with varying interests, potentially carries with it more latent conflicts than other forms of systems development." (ibid, p.15)*

Research into user participation in systems development (e.g. Howcroft and Wilson, 2003) suggests that, apart from users' attitudes and methodologies, *structural constraints* in the organisation play substantial roles in the business of participation. Howcroft and Wilson (2003) observed that user participation is often being used as a managerial token gesture for various non-technical purposes, e.g. to promote their humanitarian image or to enrol users' commitment to the organisational changes early in the systems development project. When this tokenism exists, participating users are often carefully selected according to compliance to the managerial perspective (Gasson, 2003; Howcroft and Wilson, 2003). This is because the managers, not the users or the developers, often have the power to initiate user participation, to determine its extent, and to select which approach adopted (Howcroft and Wilson, 2003).

Apart from the imbalance of power distribution in systems development, the mismatch of skills and expertise between developers and users also contributes to the difficulty of achieving genuine participation (cf. Howcroft and Wilson, 2003). Despite the fact that a number of approaches have been developed to promote a higher degree of user participation with this mismatch in mind, however, none of them comes close to fulfilling all

the human-centred principles that I discussed in section 2.5.1. In her NordCHI 2006 keynote speech, Susanne Bødker (2006) remarked "the human actor needs to step out of the role as worker in a particular practice, and participate in design as a person who brings her entire life to the design." When considering how far the current implementations of PD fall short of Bødker's ideal notion of participation, we are led to question whether it is the current approaches that are ineffective or whether the conceptual constructs that underlie current approaches (or more generally, modern systems development) are appropriate for promoting genuine participation.

2.5.3 The efficacious disposition perspective

Earlier in this chapter (in section 2.4.2), I synthesized an evolutionary process model to describe groupware evolution. This model is, in fact, compatible with an emerging perspective in interactive product design, and in HCI and CSCW in general, which argues that the design process does not stop when the piece of computer technology is handed over to its consumer (cf. e.g. Henderson and Kyng, 1992). In this sense, the boundary between design and use become blurred as the consumer will continue to *design-in-use* (Henderson and Kyng, 1992). That is, the role of the computer technology is continuously shaped through continued reconfiguration during its use. Consequently, designs of interactive products have aimed at providing flexibility for their consumers to *appropriate* its use rather than prescribing use (cf. Suchman, 2007).

My perspective on groupware development here is similar to, though different from, the *design-in-use* interactive product design perspective. My perspective is neither that associated with professional software development (in the sense of traditional software development) nor that associated with end-user software development. It transcends the notion of end-user software development but not to the extent that it encompasses professional software development with semi-formal (e.g. UML) or formal methods (e.g. B-method) (this will be explained in more detail in chapter 7). While I agree that the scope of the *design* process of groupware has now been extended into the use context, I argue that the entire artifact cycle should be thought as a process in which some or all of the stakeholders are continuously searching for *efficacious dispositions* for the artifact and its

stakeholders throughout the artifact's life.

In his book "The propensity of things", the philosopher François Jullien describes *efficacious dispositions* as one of the meanings of the Chinese concept *shi* in the context of ancient Chinese philosophy. He argues that the Chinese interpretation of reality "appears to proceed through the understanding of the disposition of things" (Jullien, 1995). Jullien then writes[41]:

> *"One starts by identifying a particular configuration (disposition, arrangement), which is then seen as a system according to which things function: instead of the explanation of causes, we have the implication of tendencies. In the former, one must always find an external element as an antecedent, and reasoning can be described as regressive and hypothetical. In the latter, the sequence of changes taking place stems entirely from the power relations inherent in the initial situation, thereby constituting a closed system: in this case we are dealing not with the hypothetical but with the ineluctable. In the context of natural phenomena and in first philosophy, this ineluctability of tendency can be expressed by the term shi, translated as either "tendency" or "propensity" depending on the word chosen by the first Western interpreters of Chinese thought as they tried to convey its originality." (Jullien, 1995, p.221)*

Therefore, obstacles (including social, organisational, and technical) and the changing nature of these obstacles in groupware development become "natural things" and "natural occurrence" when viewed through the lens of the *efficacious disposition*. The focus of the groupware development is then shifted to finding a suitable configuration for the situation by configuring and arranging existing ingredients in the situation from time to time, taking the

[41] Jullien's (1995) work was originally published in French. The quote used here is taken from the English translation version by Janet Lloyd. In her translator's note (Jullien 1995, p.9), Lloyd expresses the difficulty in translating the French word *dispositif* into English, where in turn Jullien had also mentioned his difficulty in translating the Chinese word *shi* (勢) (ibid, p.16). According to Lloyd, "*Dispositif* refers to the efficacy of a *disposition* (whether strategic, aesthetic, etc.), its capacity to function spontaneously and inexhaustibly. Every configuration or disposition possesses an inherent potential or propensity that is fulfilled by the *dispositif*. In other words, a *dispositif* is a "setup", a colloquial expression to which I have often resorted in preference to longer, heavier phrases such as those above." (ibid, p.9)

evolving nature of all these ingredients into account. All activities in the groupware development can then be viewed as being concerned with *efficacious dispositions*, including both technical and non-technical activities.

The efficacious disposition perspective does not attempt to draw a hard boundary between the context of development and use, and does not divide the roles between stakeholders. Instead, this perspective blends the contexts and roles of the stakeholders in relation to the *development* of the artifact (i.e. groupware in the context of groupware development). The treatment of stakeholders in this perspective is not the same as in traditional software development, but closer to that in Participatory Design, which views *user* as "categories describing persons differently positioned, at different moments, and/or with different histories and future investments in the project of technology development." (Suchman, 2007, p.278-279) Furthermore, the efficacious disposition perspective is human-centred. It puts its emphasis on the support for meaningful and enriched human work by disposition, that is by (re-)configuration, of objects within the context of work. This shifts the focus from development as an exercise in prescription to development as exploration of the most *efficacious* configuration for the work.

In the next chapter, I discuss the role of developers and users, their relationships, and the conflation of development and use contexts due to their role-shifting behaviour. I also reformulate the notion of participation and the development process upon which the rest of this thesis will be based.