

**Four principles for groupware design:
a foundation for participative development**

Andy Cockburn
Steve Jones

Technical Report COSC 14/93

Department of Computer Science
University of Canterbury, Private Bag 4800
Christchurch, New Zealand

Four principles for groupware design: a foundation for participative development

Andy Cockburn
Dept of Computer Science
University of Canterbury
Christchurch
New Zealand
andy@cosc.ac.canterbury.nz
Tel: +64 3 364 2774

Steve Jones
Dept of Computer Sciences
Dundee Institute of Technology
Dundee
Scotland
mctsrj@uk.ac.dct.cc.vaxb
Tel: +44 382 308619

November 1993

Abstract

Participatory design amalgamates the expertise of interdisciplinary specialists with the task-specific expertise of end-users. Groupware design is widely recognised as benefiting from participative approaches. Recognition of this ideal, however, does not preclude the failure of groupware design due to poor communication and inadequate understanding between participants. We provide a grounding in the problems affecting groupware success, and introduce four design principles that guide all those involved in design around the pitfalls that have been encountered, some repeatedly, by groupware.

Keywords: groupware, participatory design, principles, user-acceptance.

1 Introduction

The failure of early groupware systems is well recorded [Gru88, Gru90]. The design approach adopted in these development projects was often characterised by computer scientists intending to radically in-

crease the efficiency of organisations through deterministic models of cooperative activity. Such design strategies have been shown to be inadequate: they fail to account for the social factors in group work. As a consequence, research into computer supported cooperative work (CSCW) has broadened beyond computer science to include the social sciences that study the subtle factors that encompass collaborative work.

CSCW research focuses on a range of goals: from providing an understanding of social factors involved in support for group work, to the development of point systems that demonstrate the potential of new and innovative technologies. Evaluation and explanation of systems is frequently a casualty of the “next system trap” with developers eager to eliminate current failings in their next implementation. Mistakes and misguided developmental decisions are frequently unreported, resulting in a lack of guidance for the next generation of system developers. Consequently the same, or similar, design errors are replicated and rediscovered.

The latest, and most promising, group-

ware development methodology is participative design [MK93]. Interdisciplinary development teams work with the end-users to co-determine the support they receive. Yet participatory design is no panacea: the mis-guided intuitions that caused the failure of early systems may be collectively held by the participatory design team. All involved need to share a common understanding of the issues relevant to groupware design.

In this paper we examine and record the major causes of groupware failure, and provide four groupware design principles that encapsulate these problems and guide design teams around them.

2 Why principles?

In 1983, Donald Norman [Nor83] argued for “more fundamental approaches to the study of human-computer technology.” He alerted human-computer interaction (HCI) researchers to the “tar pits and sirens of technology,” referring to the temptations of system development and the self-serving enticements of new technology. He argued the need for fundamental principles to “broaden our views, sharpen our methods, and avoid temptation.”

CSCW research is now in a similar situation to that of HCI in 1983. It can be viewed as research territory into which exploration has only just begun, rife with uncharted tar pits and sirens. The lessons of CSCW development are akin to folklore with design issues permeating local research communities, but often going no further.

The principles presented in this paper provide system designers¹ with a guiding “chart,” noting the relevant pitfalls, problems, and barriers to the development of successful cooperative work support tools. Per-

¹In this paper we use the term “designers” to mean all those involved in (participatory) design. The shared understanding that is the aim of the principles is equally important to all participants, regardless of their educational backgrounds.

sonal intuitions, and a “feel for the right way to do things” remain a major part of design (and the key factor distinguishing good designers from bad ones), but intuitions, regardless of their foundations, are fallible. These principles alert designers to groupware’s problems, and to the mis-guided intuitions encountered (some repeatedly) in collaboration support.

2.1 A pragmatic approach

The design guidance provided in this paper is derived from a number of sources. Much of it is gleaned from the fragmented lessons of groupware development that are recorded in CSCW literature.

The social implications of groupware are often cited as the fundamental barrier to its success [SK91]. We do not contend with this claim; rather, we note that a society’s rejection of groupware is driven by an accumulation of individual rejections. If principles for groupware design make systems more acceptable to individuals, without hindering their value in group support, they are likely to be more acceptable to the user society.

The three major issues hindering the success of groupware (detailed in section 3) are:

- The lack of integration — this applies to several aspects of computer systems including interface differences, and incompatibility caused by information format requirements.
- The additional effort required to use systems — partially due to lacking integration, but effort is also imposed on users by explicit system requirements for information, and by the enforcement of specific usage styles.
- The vicious circle of groupware adoption (section 3.5) — if all system borne benefits are dependent on critical mass there will be little to encourage adoption until critical mass has been achieved.

Each of the four principles for groupware design addresses a combination of these problems.

Maximise the likelihood of personal system acceptance — aiming to increase the perceived and immediate benefits. By making systems appealing to individuals, regardless of the number of other users, systems become less dependent on critical mass and provide the “kick-start” necessary to overcome the vicious circle of adoption.

Minimise the requirements imposed on users — explicit system requirements are the actions that *must* be executed by users for correct system operation. Such requirements impose an additional work burden on users (someone must carry out actions for benefits to be realised), and reduce system compatibility.

Minimise the constraints imposed on users — constraints imposed on users restrict and frustrate their natural styles of working, and limit their flexibility in customising information input/output formats.

Maximise the potential for external system integration — this principle examines the wider work environment that is beyond the direct control of system designers. Issues include the different hardware platforms supported by organisations, and how to draw together disparate resources contributing to efficient cooperative work.

The causes of groupware failure that motivate these principles are examined in the following section.

3 Causes of Groupware Failure

Forming and maintaining collaborative relationships is difficult. Even in ideal work environments continual trade-offs, give and take, between collaboration participants is required. The inclusion of computer support in this complex balance is frequently counter-productive. Rather than enhanc-

ing group efficiency and cohesion, computers hinder it. Typifying the extreme level of discontent with groupware, users called the *Coordinator* “worse than a lobotomised file clerk” [CG88], and reactions like the following about the Colab meeting support system are not uncommon:

“...they found it so frustrating that they put their heads in their hands, raised their voices, and ultimately threatened to walk out. They expressed astonishment that anyone would build such a tool,” [TFB91] page-190.

What can be done to improve the poor performance of groupware? The issues of interest to groupware developers are “what can be done to avoid the failure of previous systems?”, and “why have previous applications failed?”. As a starting point, Grudin’s widely cited paper (1988) identifies three major causes of failure in CSCW applications²: the disparity between who does the work and who gets the benefit; the breakdown of intuitive decision-making in design; the underestimated difficulty of evaluating CSCW applications. These three points can be generalised into three levels of failure: system-use, system-design, and system-evaluation.

In the following sections we examine the causes of groupware failure at the system-use level, pinpointing the components of Grudin’s cost/benefit disparity and analysing the relationship between them. The observations made motivate the principles which assist designers in overcoming failure at the system-design level.

3.1 Effort in Collaboration

The costs or undesired aspects of collaboration are the overheads of *effort* beyond that required to execute personal work tasks.

²Grudin later extended these three points to include conflicts with social norms and inadequate facilities for exception handling.

Naturally, there are a plethora of social factors which can inhibit and discourage collaborative work, regardless of its supporting mechanisms. Many of these factors can be considered to increase “effort”: personality clashes, for instance, make collaboration burdensome. Social complications of such a fundamental nature are, however, beyond the scope of this investigation.

When collaborators are physically remote, communication mechanisms must be used to mediate the interaction. All non face-to-face interaction mechanisms are limited by their bandwidth, reducing the richness of interaction³. The reduction in communication richness necessitates greater *effort* in completely and accurately transferring information [HS92].

These issues are inherent in collaboration and its mediation. When computers are used to support group work there is a further imposition of effort due to explicitly required user-actions, constraints on flexibility, and transitions between methods of task accomplishment. It is these issues that the following sections address.

3.2 Effort imposed by system requirements

Many systems explicitly require additional effort from users in order to support their functionality [CT93]. Usually this effort takes the form of structured information which we will term *guidance*. Guidance-dependence is best exemplified by the wide range of applications that support and enhance asynchronous messaging through the use of semi-structured message templates [MLF92]. If guidance is not supplied then, for successful operation, some other user must provide it on the sender’s behalf. When systems attempt to maintain an active role in collaboration (coordinating activities, for example), failure to capture

³Whether this will always be the case is discussed in [HS92], and is the subject of much futuristic virtual reality research.

guidance detrimentally effects *all* users: the knowledge base on which active assistance is based becomes corrupt causing problems such as redundant or mis-timed reminders.

3.3 Effort imposed by lack of flexibility

Several CSCW applications have been based on explicit theories of cooperative tasks: examples include speech-act theory [FGHW88] and IBIS [CB88]. Systems based on rigid theories will be inflexible and impose specific styles of use that may conflict with those preferred by users.

Inflexible and constraining systems are likely to be unpopular: they enforce a form of “work to rule,” a phrase synonymous with inefficient, restricted and *inflexible* working practices. Although users may find ways of working around system-imposed restrictions (perhaps using alternative mechanisms to record personal views—a paper note pad for instance), such work-around strategies illuminate system inadequacies.

3.4 Effort imposed by lack of integration

Sources of additional effort derived from groupware go beyond the requirements and the flexibility constraints imposed by each *independent* system. Effort is also required to manage work environments in which various tools, facilities, and communication mechanisms are used in conjunction.

In computer supported *personal* work people are required to make *transitions* (changes in their styles and methods of working) between the facilities used in their everyday work. When computer support is used for *group* work, additional transitions are required: between single- and multi-user applications, and between alternative communication mechanisms. If groupware is used infrequently, the effort required to re-learn its interface may be sufficient to discourage participation in group-work altogether.

Inadequate integration between groupware and other computer tools is not only a source of user-effort, it is also a missed opportunity. Computers can integrate access a variety of information about communications and collaborators, they can actively initiate collaborations, and can carry out autonomous processing to establish suitable colleagues or communicants [CG93].

3.5 Adoption and Critical Mass

The additional effort required by a system will discourage users from adopting it. Users will also be discouraged by the imposition of inflexible working methods. Although the promise of work enhancement may encourage system use, groupware tools are prone to a vicious circle that restricts the realisation of system borne work enhancements (figure 1).

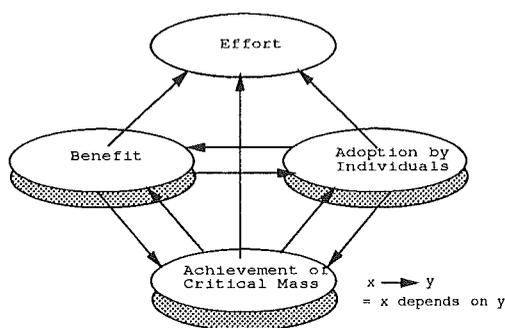


Figure 1: The “vicious circle” of dependencies in groupware adoption

The factors in this chain of dependencies are the system’s perceived *benefits*, its *achievement of critical mass*, and its *adoption by individuals*. The key determinant in the realisation of each of these components is the level of *effort* involved in system use.

Benefit and benefit-lag. Willingness to adopt a system is dependent on the benefits derived from its use, and during adoption this is primarily determined by *immediate gains*. All computer systems, however, suffer

from “benefit-lag,” the period during which the effort put into mastering a system outweighs the benefit received. Mantei (1989) notes that “... a high learning threshold would cause meeting participants to reject the technology”.

Overcoming benefit lag is a complex problem for groupware. Not only must each individual undergo the learning process necessary to master new mechanisms, but the benefits encouraging this learning burden may not be available until critical mass has been established.

Attainment of critical mass. Achieving critical mass depends on adoption by a *sufficient* group of individuals. Sufficiency in this context is contingent on the group, individual, and task requirements: in one group-task the main factor for overcoming critical mass might be the number of collaborators, while in another, the involvement of particular individuals might be the main determinant.

Adoption by individuals. Individuals will be encouraged to adopt a system if there is an established base of regular users. A community of users ensures that information about the system and how to use it will be readily available, and consequently helps to break the inertia-driven maintenance of current working practices. Personal use of systems will be further encouraged if the rewards for doing so are clearly apparent: *personal* use is most likely to be stimulated by *personal* benefits.

The vicious circle of adoption. The vicious circle relating benefit, critical mass, and personal encouragement requires that all these properties are simultaneously available *before* groupware can become successful: critical mass depends on adoption by individuals which is encouraged by benefits, but the benefits are contingent on a critical mass of users. This situation appears to foretell a gloomy future for groupware!

What is required is some sort of kick-start,

a break in the vicious circle allowing, for instance, benefits without the achievement of critical mass. The dominant and discouraging role of effort throughout system adoption and subsequent use must also be minimised.

The groupware design principles, described in the following sections, further encapsulate groupware problems, and offer some generic strategies that work towards the breakdown of the vicious circle.

4 Maximise personal acceptance

Maximising personal acceptance is concerned with encouraging individual users to incorporate new systems into their work routines. There is a similarity in how users view systems for personal and group work (for example, a word processor and a collaborative writing system). A common question users ask about both types of tool is “what can it do for me?” During initial system use, this question will carry an additional component, “*now*.” We therefore stress, in accordance with Borenstein and Thyberg (1991), the importance of interface issues in groupware. In addition to underlying the need for close attention to groupware user-interfaces, implementation strategies for encouraging personal acceptance include “feature ticking,” and the use of “champions.” The “reflexive perspective” notes the group-like requirements of personal work coordination and argues that these similarities should be exploited by groupware.

Catchpenny systems. Feature ticking is a sales ploy used to add instant appeal to a wide range of modern products. Attractive features and additional facilities supplement the core functionality, turning attention away from the key task and onto fancy bells and whistles. While not condoning the design of poor (but feature rich) systems, a form of feature ticking could be used to sup-

ply instant user-appeal.

A “Reflexive Perspective” of CSCW. Personal work is distributed through time and space. Single users may work on several machines, one at the office, one at home, a lap-top, and an assistant’s machine, and several projects may be pursued at different times. The group-like properties of the single user are further illustrated when the separate roles undertaken in personal work are examined [CT91]. These roles include the following: a *management role* in which decisions about work coordination are made; a *worker role* in which the actions necessary to advance or complete the work are executed; a *meta-management role* in which personal assistants (human or computer) are instructed about appropriate actions.

With multiple tasks, roles, and work places, the individual’s coordination requirements are similar to those of asynchronously collaborating co-workers. The aim of the reflexive perspective is to use similarities in personal and group work to blur the distinction between support mechanisms. By doing so, the user benefits from familiarity and predictability arising from a consistent interface to the personal and collaborative work environments. Skills transfer from one environment to the other, and the effort of learning and remembering separate interfaces are shared over a wider range of tasks.

Champions and encouragement. Studies of CSCW adoption [Ehr87, FRCL91] have shown that enthusiasm for new systems is greatly enhanced by “champions” or “evangelists” who promote the use of the technology, raise awareness of what it can achieve, and generally encourage system use. Fafchamps *et al* (1991) noted this in their study of decision making through email conferences, “...the single most important factor for a successful computer conference is the activity level of the organiser of the conference,” page-220.

5 Minimise requirements

User effort plays a pivotal role in system adoption (see figure 1) but a system's *dependence* on user effort has detrimental effects beyond issues of system adoption. These include the imposition of a cost/benefit disparity between those carrying out actions and those receiving the benefit, and an increased likelihood of system incompatibility due dependence on specific information structures/format.

The intention of minimising requirements is to reduce the disparity between groupware's costs and benefits to user-acceptable levels. Strategies for achieving this goal, summarised below, include avoiding dependence on additional work, exploiting information inherently available through communication, and enabling shifts between the provision of benefits and the imposition of costs so that those most willing, able, or in need receive appropriate support⁴.

Avoid dependence on user actions. The problems caused by system dependence on information that is explicitly provided by users (guidance) were discussed in the section on user-effort. Rather than depending on guidance, a more acceptable approach would use guidance when available, but not depend on it for system operation. It has however been argued that a relaxed approach of this nature is impractical due to the inter-relations and dependencies inherent in collaborative work:

“Can a CSCW application succeed if doing the extra work is left to individual discretion? Unfortunately, probably not.” Grudin (1988) page-86.

Unfortunately, depending on and requiring actions from users is as likely to cause system rejection as leaving the work to individ-

ual discretion. Certain forms of user-support will, indeed, be dependent on structured information. The major problem for system designers in supporting this structured information is *how* to retrieve it. The user is the most readily accessible and accurate resource, but research and experience has shown that systems can profitably look elsewhere [CS93, KM93, CT93].

Use what's available “for free.” Avoiding dependence on user actions raises conflicting aims for system designers: how to leave users free from requirements, and yet still provide enhanced facilities.

Although guidance information is required to provide certain types of benefit, there are information sources other than that explicitly provided by users. Information is often accessible to computers through the process of communication: for instance, email messages contain header information revealing at least the who, the when, and the where information about a communication, and can also detail the subject matter, the direct relationship with previous messages, and so on. Existing “for free” approaches are wide ranging and include the following: the use of information in standard email headers to infer conversational relationships between email messages (*Mona* [CT93]); adaptive and learning systems that modify their performance dependent on user characteristics [KM93] or interaction pace [Dix92]; and techniques such as Latent Semantic Indexing [FD93] which infers the semantic distance between text documents.

Enable shifts of cost and benefit. Designers and managers who strive for efficiency-enhancing groupware have, typically, assumed that people are willing to work for the benefit of others [Gru88, Nag90]. This assumption ignores social affects, including the users' reluctance (or inability) to carry out actions that provide no *personal* benefit.

By shifting the provision of guidance (the cost) onto users gaining the benefit the

⁴For a complete discussion of the strategies used to implement minimised requirements, and point systems demonstrating their use, see [Coc93].

cost/benefit disparity is reduced—users execute additional actions when they are willing and able. Tapestry [GNOT92] underlines such an approach: through “collaborative information filtering” it exploits those people who are willing and eager to carry out the additional work.

The applicability of a cost-shifting approach depends on the politics and hierarchical structure of the organisation in which it is implemented. Although it may be reasonable to expect subordinates to work on behalf of a manager, the reverse may not be true. Social protocols should be allowed to resolve conflicts between expectations of actions and execution of actions. Enforcing rigid dependencies on the work of others will, for many groups, precipitate system rejection.

6 Minimise Constraints

Minimising requirements is concerned with the implementation stage of groupware development. It focuses on *how* systems retrieve the information they require. *Minimising constraints* attends to problems arising at an earlier and more abstract stage of system development. It examines the models and theories underlying groupware support. The aim is to avoid inflexible and constraining styles of use.

There is a causal relationship between the imposition of user-constraints and resultant user-requirements. Explicit models of group work processes constrain user flexibility, and require explicit guidance for correct operation. Designers implementing such models must ensure that their system can secure the required information, and the user is the most obvious source—hence the imposition of requirements.

Although rigid working practices can, in principle, support highly efficient organisations, in reality few organisations operate according to such deterministic methods; furthermore, they cannot be made to do

so [Nag90]. Minimising constraints concurs with the sentiments of Dykstra and Carasik (1991) who, during development of the *Amsterdam Conversation Environment*, considered “...what it was that we really wanted to support: processes or people?”, page-420. Their definition of support for *people* as “non-dependency-creating enablement” argues for groupware that leaves users free to develop protocols governing collaborative work as *they*, rather than their systems, see fit.

Strategies for achieving minimal constraints, summarised below, primarily aim to increase designers’ awareness of problems arising from inflexible and rigid systems. Specific and detailed strategies are likely to be inappropriate due to the diversity of the models implemented by groupware.

Be aware of the two level perspective of technology. Sproull and Kiesler’s (1991) two level perspective of technology examines conflicts between increased efficiency available through computer support and its negative social implications. The first level addresses the increased efficiency enabled by particular styles and uses of technology. The second level is concerned with social effects, raising issues such as user acceptance, personalised views of information, and individual preferences. The distinction between these levels can be expressed by the contrasting questions “what is *possible* with technology?” at the first level, and “how will it be used?” at the second.

Groupware designers, and all those involved in system development, must be aware of the social implications inherent in group work support. Technology capable of enhancing organisational efficiency will fail if relevant social factors are ignored. Design alterations based on projections of a system’s social implications may temper the efficiency improvements achievable, but it is better to provide acceptable mechanisms providing some benefit than unacceptable ones which, despite great potential, fulfill

none.

Beware of rigid models and theories. CSCW research into collaborative activity promises to yield workable explicit models for groupware in the future. However, the lack of maturity and incomplete state of this research makes the use of explicit models in *current* groupware largely inappropriate.

Open, unconstrained enhancement. While models and theories of collaborative activity are under development, open and unconstrained systems allow users to develop protocols as they see fit. User-specific models might be used to supplement an open system, but they should not impose constraints on collaborative tasks or on their mediation.

Several existing groupware applications exemplify various “open” approaches. In investigating the *Capture Lab*, Mantei (1989) cites the inadequate understanding of social exchange protocols; the system therefore supports a socially negotiated protocol for access to the system’s shared workspace. Similar observations have led to toolkits such as GROUPKIT [RG92] supporting a range of open protocols for floor control and other facilities. *Milo* [Jon92] avoids modeling specific writing styles/roles in order to free co-authors from constraints.

7 External Integration

The first three groupware principles are primarily concerned with design and use of groupware *in isolation*. In contrast, maximising external integration requires designers to consider their system’s role within, and relationship to, the entire work environment. In this extended collaborative context, group members use competing systems to execute similar tasks, and a variety of tools (computer and non-computer based) are drawn on to support and assist collaboration.

Enabling external integration attends to

the user-effort that results from the changes (transitions) made between differing applications or communication environments. Its primary aims are twofold:

Curative — to reduce the number and magnitude of transitions between tools and facilities employed in collaborative work.

Augmentative — to improve and integrate access to resources that serve communication and collaboration requirements.

In this section we describe groupware approaches that reduce transitions (also termed “seams,” and “barriers” [Bae93]) in CSCW.

Video fusion. Best exemplified by *Clear-Board* [IKG92] video fusion systems reduce the lack of compatibility between personally favoured tools, and ease the disruption to communication efficacy in distributed work caused by the loss of communicative cues such as gesture. Video fusion allows separate video images (perhaps a computer screen, and a human face) to be simultaneously displayed “on top of” each other, like layers of transparent acetate. In this way two (or more) otherwise incompatible applications, or work environments, can be used together.

Heterogeneous environments. Video-fusion techniques are primarily curative, overcoming problems brought about by the lack of integration between existing work support tools. In contrast, heterogeneous platforms augment collaboration by drawing together access to, and information about, collaboration resources in a way impossible without computer support. They work towards an “integrated portfolio of media” [Bai89]. For example, TELEFREEK [CG93] provides an extensible CSCW environment based on, but not limited to, standard networked computers. By drawing together information sources, communication mecha-

nisms, and collaboration applications, TELEFREEK users are provided with a platform for communication and collaboration.

Minimise dependence on structure and format. Dependence on system specific information formats and structures reduce the potential for system integration. Minimally, groupware that is intended for general release (rather than research point systems) must follow relevant standards. Many standards allow flexibility and additional structure to be added within their specifications, but designers must consider the impact of such structures on colleagues who do not have access to the same structuring mechanisms [LM90]. Systems built on existing communication media should do so monotonically: new facilities and enhanced features should not affect those already in use.

Implementation platforms. Incompatibilities between hardware and interface platforms are a primary concern for groupware developers. To avoid these problems designers must either replicate some of the implementation to support a variety of hardware, choose a suitable hardware-independent development platform (such as the X Window system), or use an interface development application that can generate code for several Graphical User Interface environments.

Cross-platform interface generators are becoming available and are likely to substantially ease the implementation of integrable groupware. Groupware development toolkits, such as GROUPKIT [RG92] and OVAL [MLF92] also promise to increase the integrability of systems.

8 The principles effect on designers

Research and experience has shown that technology is able to offer novel, efficient, and work-enhancing facilities. How-

ever, providing systems *capable* of enhancing group work is insufficient, they must also (in many ways, primarily) be acceptable. Surface system issues such as interface quirks, or failure to provide adequate appeal to new users are likely to prompt system rejection. Designers may argue that such superficial problems will be overcome once enhancements in work efficiency are recognised, but rejection at an early stage means that the benefits will never be attained.

Employing these principles in system development will make substantial demand on development teams. The question asked by designers, however, should not be “which approach is the easiest?”, but rather “which *potentially successful* approach is the easiest?” We contend that these additional development demands are not purely a consequence of the principles; rather they are qualities required for the development of successful and (necessarily) acceptable group work applications.

9 Conclusions

Participatory design offers a potential solution to many of the problems of groupware development. It also threatens to complicate and confound it if the team has a lacking appreciation of the complex design issues. In this paper we have provided a cross disciplinary perspective on groupware development that focuses on those most greatly affected by its deployment: the users. We have noted the common failings of groupware, and from those observations we have forwarded a set of design principles that encapsulate both the problems and ways to avoid them.

Three systems specifically adhering to the principles have been developed by the authors. *Mona* [CT93] supports a conversation-based email platform that does not require explicit user guidance, *Milo* [Jon92] is a minimally-constraining collaborative writing tool, and TELEFREEK [CG93] supports an extensible and customisable

platform for communication and collaboration resources. These point systems are freely available from the authors⁵.

References

- [Bae93] RM Baecker. The future of groupware for CSCW. In RM Baecker, editor, *Groupware and Computer-Supported Cooperative Work*, pages 851–853. Morgan Kaufmann, 1993.
- [Bai89] JH Bair. Supporting cooperative work teams with computers: Addressing meeting mania. In *Proceedings of the 34th IEEE Computer Society International Conference. San Francisco, CA. Feb 27–Mar 3.*, pages 208–217, 1989.
- [CB88] J Conklin and ML Begeman. gIBIS: A hypertext tool for exploratory policy discussion. *ACM Transactions on Office Information Systems*, 6(4):303–331, October 1988.
- [CG88] RP Carasik and CE Grantham. A case study of CSCW in a dispersed organisation. In *Proceedings of CHI'88*, pages 61–65, 1988.
- [CG93] AJG Cockburn and S Greenberg. Making contact: getting the group communicating with groupware. In *Proceedings of COOCS'93 Conference on Organisational Computing Systems. November 1st to 4th. Milpitas, California.*, 1993.
- [Coc93] AJG Cockburn. *Groupware design: principles, prototypes, and systems*. PhD thesis. University of Stirling, Scotland. 1993.
- [CS93] D Crow and B Smith. The role of built-in knowledge in adaptive interface systems. In *Proceedings of IWIUI'93, Florida.*, pages 97–104. ACM Press, January 1993.
- [CT91] AJG Cockburn and HW Thimbleby. A reflexive perspective of CSCW. *ACM SIGCHI Bulletin*, 23(3):63–68, July 1991.
- [CT93] AJG Cockburn and HW Thimbleby. Reducing user effort in collaboration support. In *Proceedings of IWIUI'93, Florida.* pages 215–218. ACM Press, January 1993.
- [DC91] EA Dykstra and RP Carasik. Structure and support in cooperative environments: the Amsterdam Conversation Environment. *International Journal of Man-Machine Studies*, 34:419–434, 1991.
- [Dix92] AJ Dix. Pace and interaction. In *Proceedings of HCI '92: People and computers VII*, pages 171–190, 1992.
- [Ehr87] SF Ehrlich. Strategies for encouraging successful adoption of office communication systems. *ACM Transactions on Office Information Systems*, 5(4):340–357, 1987.
- [FD93] PW Foltz and ST Dumais. Personalized information delivery: An analysis of information filtering methods. *Communications of the ACM*, 35(12):51–60, 1993.
- [FGHW88] F Flores, M Graves, B Hartfield, and T Winograd. Computer systems and the design of organisational interaction. *ACM Transactions on Office Information Systems*, 6(2):153–172, 1988.
- [FRCL91] E Francik, SE Rudman, D Cooper, and S Levine. Putting innovation to work: Adoption strategies for multimedia communication systems. *Communications of the ACM*, 34(12):53–63, 1991.
- [FRK91] D Fafchamps, D Reynolds, and A Kuchinsky. The dynamics of small group decision-making using electronic mail. In JM Bowers and SD Benford, editors, *Studies in Computer Supported Cooperative Work: Theory, Practice and Design*, pages 211–224. North-Holland, 1991.
- [GNOT92] D Goldberg, D Nichols, BM Oki, and D Terry. Using collaborative filtering to weave an information

⁵They are written in C and run under Unix and X Windows.

- tapestry. *Communications of the ACM*, 35(12):61–70, 1992.
- [Gru88] J Grudin. Why CSCW applications fail: Problems in the design and evaluation of organizational interfaces. In *Proceedings of CSCW'88* September 26–28 1988. Oregon., pages 85–93, 1988.
- [Gru90] J Grudin. Groupware and cooperative work: problems and prospects. In B Laurel, editor, *The Art of human-computer interface design*. Addison-Wesley, 1990.
- [HS92] J Hollan and S Stornetta. Beyond being there. In *Proceedings of CHI'92* Monterey, May 3–7 1992, pages 119–125. Addison-Wesley, 1992.
- [IKG92] H Ishii, M Kobayashi, and J Grudin. Integration of inter-personal space and shared workspace: Clearboard design and experiments. In *Proceedings of CSCW'92* October 31 to November 4 1992. Toronto, Canada., pages 33–42, 1992.
- [Jon92] S Jones. MILO: a computer based tool for (co)authoring structured documents. In M Sharples, editor, *Computer Supported Collaborative Writing*. Springer-Verlag, 1992.
- [KM93] R Kozierok and P Maes. A learning interface agent for scheduling meetings. In *Proceedings of IWUI'93, Florida*. pages 215–218. ACM Press, January 1993.
- [LM90] J Lee and TM Malone. Partially shared views: A scheme for communicating among groups that use different type hierarchies. *ACM Transactions on Office Information Systems*, 8(1):1–26, 1990.
- [Man89] M Mantei. Observation of executives using a computer supported meeting environment. *Decision Support Systems*, 5:153–166, 1989.
- [MK93] MJ Muller and S Kuhn. Introduction to the special issue on participatory design. *Communications of the ACM*, 36(4):24–28, 1993.
- [MLF92] TW Malone, K-Y Lai, and C Fry. Experiments with oval: A radically tailorable tool for cooperative work. In *Proceedings of CSCW'92* October 31 to November 4 1992. Toronto, Canada., pages 289–297, 1992.
- [Nag90] M Nagasundaram. Style and substance in communication: Implications for message structuring systems. *ACM SIGOIS Bulletin*, 11(4):33–41, 1990.
- [Nor83] DA Norman. Design principles for human-computer interfaces. In *Proceedings of CHI 83*, pages 1–10. New York: ACM Press, 1983.
- [RG92] M Roseman and S Greenberg. Groupkit: A groupware toolkit for building real-time conferencing applications. In *Proceedings of CSCW'92* October 31 to November 4 1992. Toronto, Canada., pages 43–50, 1992.
- [SK91] L Sproull and S Kiesler. *Connections: New ways of working in the networked organization*. The MIT Press, 1991.
- [TFB91] DG Tatar, G Foster, and DG Bobrow. Designing for conversation: Lessons from cognoter. *International Journal of Man-Machine Studies*, 34:185–209, 1991.