Beyond Design: Social Learning and Computer-Supported Cooperative Work: some lessons from Innovation Studies

Rob Procter
Department of Computer Science
Edinburgh University
Scotland, EH9 3JZ
rnp@dcs.ed.ac.uk

Robin Williams
Research Centre for Social Sciences
Edinburgh University
Scotland, EH8 9JU
r.williams@ed.ac.uk

Keywords:

CSCW methodologies, innovation, organisational issues.

Introduction

The concept of Computer-Supported Cooperative Work (CSCW) offers a potentially important contribution to improving the utility and usability of Information Technology (IT). CSCW takes as its starting point the fact that many of the working activities in which we take part are collective. In suggesting that IT systems design should take on board the social character of work, the CSCW community is at the forefront of attempts to improve design methodologies through the application of social sciences. Much recent thinking has focused upon the need to incorporate social complexity into CSCW design, and has flagged the value of ethnographic and other sociological approaches to this end. This paper argues that these attempts are important, but do not represent the sole contribution of the social sciences to the conception, creation and adoption of CSCW.

We explore possible lessons from Innovation Studies for the development and prospects for CSCW, and argue that the experiences of existing processes of IT systems design and implementation have important implications for the ways we conceptualise the social learning processes involved in the creation and adoption of technologies. In particular this bears upon:

1) how we conceive CSCW and its relationship with conventional systems design;
2) the relative importance of the initial design of CSCW versus the more widely distributed processes of social learning (or *innofusion*) involved in its subsequent implementation and diffusion.

These insights raise some critical uncertainties surrounding the future development of CSCW and the ways in which social scientists might best contribute to it.

We explore some of the ways in which social learning in CSCW could be supported, considering in particular the scope for customisation by end-users, and argue that these must be analysed in the context of broader processes, of change within the organisation, and of feedback from use to future technological supply.

Problems in the Usability of IT

Many of the problems experienced in the organisational application of IT stem from the overly simplistic concepts of the nature of organisations and their activities that have underpinned IT design. Considerable difficulty has been experienced in appropriating for the purposes of design the specific circumstances and requirements of organisations, which derive from their particular activities and context, and which are rooted in organisations' distinctive practices and cultures. There has been an extensive discussion of the problems of systems design, and in particular, of traditional approaches which exhibited a marked technocratic approach (see for example Briefs et al. 1983). Systems design practice tends toward a model of the organisation that is socially homogeneous and simplified: of actors with clear and common goals; engaged in easily specifiable actions and transactions which can readily be represented in terms of formal decision models and abstract information flows. This is in contrast with real organisations in which information flows and decision making are often highly informal, in which the purposes and activities of the organisation are contested and differentiated, and are subject to divisions of labour, knowledge and control (Greenbaum and Kyng 1991).

Success in the design of IT systems has been greatest when developing tools for discrete tasks and functions to support well-delineated organisational activities. Initial commercial applications were for areas of routinised clerical labour (for example in record-keeping). Generic computer-based tools have emerged where stable sets of activities can be generalised across a range of organisations (e.g. word-processors, spreadsheets etc.). Exploitation of the growing body of human-computer interaction (HCI) design principles and techniques has enabled such tools to be used effectively by technically unsophisticated end-users, and they have been taken up in large numbers. Such tools have proved extremely attractive to users, because of their low price, capabilities and ease of use -- particularly in the rapidly growing area of personal computing. Such discrete applications are now available as highly standardised commodities -- and marketed on an increasingly global scale, to exploit economies of scale in development[1].

The other thrust of IT development has been to increase the range and complexity of organisational activities supported. On the one hand we note the shift of systems development towards the sharing of information between hitherto separate organisational activities and functions. These are directed not just towards cost savings (e.g. by eliminating double-entry of data), but towards greater flexibility and effectiveness by closer integration of organisational activities. On the other hand, as routine tasks have become automated, IT systems have become increasingly focussed upon higher level organisation activities such as planning and decision-support. In

[1] There are in addition network externalities from standardisation which include, importantly, the benefits available from industry standard interfaces that allow interconnection and inter-operability between different components, and the savings in training and adoption costs for user firms arising from standard user interfaces.

this way, increasingly comprehensive IT systems are becoming vehicles for the communication and coordination of information and activities across -- and beyond -- the organisation. By their nature, such systems must be designed to fit particular organisational contexts and activities. Under these circumstances generic solutions may be inappropriate. Attempts to implement 'standard' packaged solutions in these circumstances have often proved unsuccessful because of their lack of fit with the particular exigencies of the user organisation (Webster & Williams 1993). Integrated systems have continued to be developed, by and large, on a custom or bespoke basis.

Figure 1 highlights the dilemmas in the supply of solutions. It summarises the volume/variety characteristics of different kinds of IT applications, in terms of the range and complexity of their scope, and their market size (Brady, Tierney & Williams 1992). At one extreme are the commodified standard packages for discrete and widespread activities. At the other are highly complex organisational solutions geared towards the specific characteristics of an individual user organisation. The arrows indicate the dynamics of development -- towards the increasing complexity on the one hand and towards their standardisation/commodified supply on the other. It shows some intermediate supply strategies which go some way towards resolving these contradictory tendencies -- for example by exploiting specific market niches of similar users, or through generic systems or 'pick and mix' configurations of more or less standard components that can be configured and adapted to meet particular user requirements.

One of the major challenges for IT design today arguably lies in the achievement within integrated systems of the same flexibility and usability experienced with discrete applications. We argue that the significance of CSCW should be viewed in this context; its approach is centrally bound up with the development of tools to support such activities.

The Distinctive Claims of CSCW

The HCI community has tended to focus upon usability as it relates to individual tasks, and its implications for user interface design. In this, it has achieved a considerable degree of success -- usability considered at the level of the single user has been improved quite dramatically. CSCW has emerged as the focus of those within the systems design and implementation community who are concerned to tackle long-standing problems with the usability and acceptability of IT systems, and which lie beyond the traditional remit of HCI (Procter & Williams 1992). CSCW is an acknowledgement of the need to apply user-centred design principles and methodologies within the domain of organisational IT. In particular, such work activities are conducted in collaboration, rather than being an agglomeration of individual tasks.

The concept of CSCW is welcome insofar as it emphasises the need for richer accounts of the social contexts of systems use. However, questions remain about whether CSCW has gone far enough. In particular it has been argued that the CSCW

community will fail to achieve its goals if it persists with a restricted and inadequate conception of the organisation (Randall, Hughes & Shapiro 1993)[2].

There is confusion in the literature as to the exact meaning of CSCW and its terms of reference. It is often used in a restricted sense to denote a new class of IT applications that are explicitly oriented to the requirements of group-based work, and hence raises the need to understand *how* group work is carried out. The broader interpretation is simply that all work is socially organised and should be understood as such (Hughes, Randall & Shapiro 1991). From this perspective, therefore, the concerns of CSCW are co-terminous with the concerns of conventional systems design and implementation, whilst signalling a paradigm shift in how these are conceptualised. It is this latter interpretation which we believe to be most appropriate to the issues we raise in this paper. CSCW applications themselves may help bring to the fore some important issues and opportunities for systems design. However, they are only part of the expanding range of IT applications in the workplace.

As an application area, CSCW is now several years old, and commercial systems are now available. There have been negative responses in many trials, and utilisation rates have been low (Carasik & Grantham 1988; Kling 1991). This failure cannot be blamed on the lack of attention paid to the standard HCI usability issues -- on the contrary, this has often been exemplary (Kling 1991) -- but rather the incongruence between design practice and organisational realities. It is in areas where CSCW involves pre-suppositions about organisational activities that the problems of design, use and acceptance appear to be greatest. These have been attributed to "designers' naive assumptions about the use of the technology" (Ellis, Gibbs & Rein 1991). Lacking a one-to-one correlation with existing social activities, the use of many CSCW applications requires further technical and social changes. In CSCW, the problem is not the complexity of the technology (though it may well be complex), but complexity of the social interaction between end-users, systems designers and implementors (Greenbaum & Kyng 1991). And one important reason for this complexity is that end-users and designers don't inhabit the the same environment and share a common practice. Early CSCW applications were very much a reflection of the perspectives and concerns of the IT design community (for example the importance of joint-authoring systems reflect their academic context). Where CSCW designers have moved beyond this, their products have often displayed somewhat naive assumptions about user contexts, which have limited their utility. It is notable that the most successful examples of CSCW -- electronic mail[3] and bulletin boards -- make few presumptions about the nature of the tasks being supported. As their names suggest, they are "mere" electronic substitutes for existing paper-based communications. The ways in which these tools are used vary from place to place; patterns of use depend upon the establishment of behavioural norms, and generally rely upon social codes ("netiquette") rather than "hard-wired" mechanisms. Indeed, where attempts have been made, as in the Coordinator (Flores et al. 1988), to

---

[2] One important element in many discussions of CSCW is a normative concern with how work might be organised which, for example, emphasises non-hierarchical forms of organisation and autonomy in relation to roles. Whilst we are sympathetic to these goals, they are sometimes espoused in a naive manner which ignores the fact that many organisational systems are designed explicitly to control ways in which organisations operate. For example finance and accounting applications seek to restrict opportunities for fraudulent and other non-standard ways of using systems. This raises some broader issues which we do not have space to discuss here.

[3] Itself an application which emerged as an unanticipated function with the development of IT networks for file-transfer.

incorporate stronger models of communicative behaviour into such tools, these have met with little success (Bullen & Bennett 1991; Suchman 1993).

In contrast to CSCW, the world of current industrial applications of IT contain a plethora of systems designed or customised to match collective working processes within and between organisations (Fleck 1993). The workplace experience also highlights real constraints in the way that systems are developed and used, and in particular, the conflicts of interest which surround the design and use of IT systems at work. For example, powerful players may not favour CSCW tools or approaches; equally CSCW tools may be resisted because they transform existing power and control relationships. These must be addressed if we are to understand the potential of CSCW. For example, it has been difficult to encourage people to use computer-based diary systems where they have been seen to involve an uneven distribution of costs and benefits for different players (Grudin 1988)[4].

Existing organisational systems could serve as a reservoir of ideas for CSCW development. Yet, to date the CSCW community does not appear to have tapped this rich resource of systems that have been adapted to match particular workplace objectives and cultures. The suggestion that CSCW might benefit from an examination of existing systems raises questions about the degree of distinctiveness of CSCW. Does CSCW represent a radically different way forward, or is it just one point in a spectrum of design and implementation approaches? Is CSCW distinctive, or does it merely represent 'good design'?

Social Learning in Technological Innovation

CSCW reflects a new concern with end-user requirements. However, in addressing these, CSCW remains solidly within conventional supply-driven concepts of how new technologies emerge. This criticism applies both at the level of methodologies, in which a traditional structure of systems design implicitly privileges the expert designer, and in the broader policy context, in terms of the ways in CSCW is promoted and is seen as being applied. In so doing, CSCW has neglected the importance of more distributed 'social' processes, involving a wide range of users as well as designers of technology. This process of innovation during the adoption and diffusion of technologies has been described as *innofusion* (Fleck 1988).

In making this point, we do not wish to deny the importance of design, or the significance of new initiatives in CSCW design practices. Ethnography, in particular, has much to recommend it in this context (Randall, Hughes & Shapiro 1993). The primary concern of all such initiatives is to improve design processes, however, and whilst this is commendable, they continue to insist that the problems arise through some failure of design, and can be fixed by increasing the attention paid to design. What they ignore are the important processes of social and technical innovation that arise when technologies are implemented -- the struggle to get technologies to work, and to adapt them to the social and technical exigencies of use. For example, standard packages supplied as integrated solutions to manufacturing and service firms alike

[4] For example they are more likely to be attractive to managers -- who are most likely to use the information, and who may well have secretaries to maintain their diary entries. Other staff may be less enthusiatic at using systems where they are seen as offering little advantage: constraining their autonomy by making their work schedule more visible, while imposing unwanted costs of data entry.

have often proved difficult to implement -- their standard presumptions about the user context were poorly matched to the particular history, practices and structure of the firms adopting them. User firms were consequently forced to embark upon (an often unanticipated process of) extensive customisation of the packages, together with a modicum of organisational adaption to get them to work. In this process some elements of the packages were unpicked; in effect the technologies were un-invented and re-invented (Brady et al. 1992; Fleck 1993; Williams & Webster 1993). Important new innovations were thrown up, some of which had wider applicability and were incorporated into subsequent technological designs.

Innofusion is a challenge to the conventional 'linear' model of innovation, which views technologies as emerging as stable solutions that can simply be plugged in and switched on in subsequent technological diffusion. For the individual user, the search for such a 'technological fix' has often been frustrating and unsuccessful. However innofusion also has a more general implication for public and commercial policies for the promotion and use of technology. In particular it draws attention to the interactions between supply and use in the successful adoption of technologies, and the importance of the implementation site as an arena for innovation.

Recognition of innofusion opens up our concept of where social and technical learning takes place. It raises choices about where resources should be placed within the overall systems life cycle. Conventional systems design and more progressive HCI and CSCW approaches alike privilege the initial design. Although the latter emphasise the role of the end-user, and may have an explicit democratic agenda of end-user participation, they nevertheless continue to privilege the computer scientist, the social scientist or other design practitioners as the final arbiters of end-user requirements. In contrast, it could be argued that the implementation process offers opportunities for the direct involvement of end-users in systems development in their own 'natural' working circumstances. In this way a wide variety of players have the opportunity become involved in a highly-distributed learning process.

This has a number of implications. Processes of technological design go hand in hand with processes of organisational restructuring. These must both be addressed. Following on from this, it should not presumed that the best, or indeed only, way forward is to emphasise initial technological design -- instead it may be better to promote processes of experimentation and innovation in the user domain. From this perspective, attention should be directed towards *collective* rather than *individual* design processes. Rather than trying to design new kinds of CSCW solutions, the conflicting commercial and technological dynamics surrounding IT development (as illustrated in Figure 1) may favour strategies of seeking to sell-on existing custom-built applications as niche-specific offerings, of supplying generic systems that can readily be customised for (and perhaps by) particular users, or of offering more-or-less standardised technological components that can be configured together on a pick-and-mix basis to meet user requirements.

Supporting Social Learning in CSCW

We have suggested that these processes of social learning and innofusion are of potential significance to the future evolution and the success of CSCW. Certainly, examples of innovations arising *in use* are already documented in the CSCW literature

(Robinson 1993). For example, the Information Lens system had been designed to serve end-users as an automatic filter to prioritise email *before* it was read; its users *re-invented* it as an aid for archiving email *after* it was read (Mackay 1990).

In this section we therefore begin to address how best to support social learning in CSCW. This involves two elements: adaption or innovation in use, and feedback from use to future supply. The former has been widely considered within the CSCW and wider systems design community. Thus Henderson and Kyng (1991) describe three basic approaches to adaptation in use:

1) adopting a 'construction set' approach to design, with the aim of providing users with a well-founded set of basic parts -- incorporating minimal assumptions of the circumstances of use -- from which users can then proceed to build a system to satisfy their requirements;
2) anticipating different scenarios of use and designing alternative behaviours into the system, together with a means to specify the one desired;
3) directly altering the system e.g. through changing the source code.

These, of course, are complementary, rather than mutually exclusive.

Construction Sets and Commodification

This approach conforms to the commodification strategy that has been already successful in relation to discrete applications -- re-packaging innovations in a form that can be subsequently transferred with the minimum of effort into a multitude of different application contexts. Through this process, a range of stable products is made available in the market place, reducing the turbulence of technical change, and thereby diminishing the risks for both producers and consumers (Brady 1992). Such commodifications serve as the building blocks for subsequent end-user innovations. In relation to CSCW applications, packages such as Lotus Notes and MicroSoft's Windows for Work Groups already provide examples of the commodification process at work.

End-User Innovation

All three approaches bear on the question of whether end-users can be empowered to become masters of their tools. The first enables end-users to pursue a pick-and-mix strategy with the confidence of technical compatibility. The second approach gives end-users a limited capacity to adapt -- or customise -- generic applications to meet their specific needs and circumstances. The third approach would be beyond the skills of the typical end-user, but may conceivably be re-packaged in a form which overcomes this problem (MacLean et al. 1990).

End-user customisation is becoming an increasingly common practice, with many applications now having customisation facilities built-in. Providing this kind of adaptability for a wide range of end-users, adds to the complexity of design, but is nevertheless a tractable problem. More fundamental is the issue of whether end-user customisation can address anything other than the more superficial kinds of adaptation. By definition, the concept of innovations in use contradicts the premise that different scenarios of use can be adequately anticipated at the design stage, and the evidence of such innovations is proof that this is indeed the reality. Where design

can aid users is in situations where there is a history of innovations which can be fed back into the design process to add to the understanding of requirements. Nevertheless, the designer can still only set the scene within which new innovations may subsequently emerge in use. Uncertainty and incompleteness is therefore a fundamental characteristic of the design problem. This is highlighted in CSCW because users have as yet little experience on which to base their expectations and requirements (Mantei et al. 1991) and the history of innovations in use is short.

Lowering Barriers to Innovation by End-Users

Many forms of customisation are easy for the average end-user to accomplish (e.g. Jorgenson & Sauer 1990). Personalising the look and feel of an application through changing fonts, colours, key bindings, menu organisation etc. are good examples. But in other instances, this simplicity may be more of a reflection of how the customisation mechanism is presented as a facility for the end-user rather than of its effects, and here the designer has an important contribution to make. For example, customising the X Window system through the default mechanisms (editing resource files) is not a task for the inexperienced user. There are various tools available, however, which provide a simpler and more intuitive approach. One issue for consideration, therefore, is to what extent tools can provide a way to overcome the skill barriers to end-user innovation (MacLean et al. 1990).

End-users vary enormously in the kinds of technical skills they possess. Typically, however, they lack the skills demanded for conventional programming tasks. End-users can circumvent immediate skill barriers by copying (and perhaps later modifying) the customisations and adaptations of those more skilled, and this is increasingly observed (Mackay 1990). With the growth of 'freeware' and 'shareware', of bulletin boards to find out about, or publicise, availability, and of file transfer facilities to expedite acquisition, the sharing of customisations and application software amongst end-users -- both within and between sites -- has become extremely easy. As this illustrates, CSCW tools themselves have much to contribute to the support and improvement of social learning processes. Sharing is not free of pitfalls, however, and end-users can easily come unstuck as a result of other changes (e.g. hardware or operating system upgrades) within the system. End-users may then find that things suddenly stop working and that they are unable to deal with the problem themselves.

Spreadsheets are cited as an example to follow of how the technical skills barriers to 'end-user programming' may be lowered (Nardi & Miller 1990). Unfortunately, they also appear to be a good illustration of the potential pitfalls of putting powerful tools in the hands of users who lack the skills to reason in depth about how they work. One recent survey estimated that over 90 percent of end-user spreadsheet programs in use within commercial organisations contained significant errors, and that 12 percent were so poor that their output was meaningless (Bray 1992). Like customisation, the sharing of spreadsheet programs also carries hidden costs for the end-user. Users often remain dependent upon the original author for maintenance and extensions, thus many spreadsheet programs become useless once their authors have left the user organisation.

Both the sharing of customisations and programs point to failures in social learning processes within organisational IT. First, sharing is a weak mechanism for learning

because it doesn't require the transfer of expertise -- indeed that is what makes it successful from a more immediate perspective. Second, the often informal nature of the interaction between author and user means that the level of commitment entered into is fragile.

End-users who do possess programming skills may still lack in-depth knowledge of their computing environment (which may be very complex, and site-specific). As a consequence, they may fail to anticipate the full implications of the changes they would like to make. For the user of a 'stand alone' PC, customisations will have strictly localised effects. In any shared computing environment, however, the activities of one user may affect others. Present-day trends in technologies favouring distributed IT have increased complexity, and with it the chances that changes may have unexpected side effects. Such systems often support several hundred users, and so the impact of side effects can be considerable. End-users often appear to be unaware of other users sharing their environment, as reflected in difficulties in distinguishing between local and system-wide problems "they complain that *their* machine isn't working, often not realising that the problem is one which is often affecting many others too".[5]

End-User Computing

End-user innovation raises many similar issues to those thrown up by the emergence of PCs and 'end-user computing' (MacLean, Kappelman & Thompson 1993). Indeed, the move of many organisational end-users from mainframe to PC-based systems can only be fully understood by considering the opportunities which PCs gave them to exercise more control over their IT practices and strategies. PCs provided organisational end-users with an alternative source of hardware and software which was cheap, flexible, and easy-to-use, and which in turn allowed them the opportunity to lessen dependence on what they often viewed as slow to act, and conservative, IS departments. In effect, this represented a move by groups of organisational end-users.to create greater opportunities for adaptability. End-user computing has failed, however, to achieve the total liberation that some predicted. PCs have to be integrated into more conventional IT environments to utilise corporate data. Moreover, current trends in technology towards distributed 'client/server' IT effectively reverse those of the stand alone PC era. Organisational end-users remain dependent upon IS departments -- and their technical staff -- for the provision and maintenance of IT services.

Trends towards end-user programming have been closely associated with end-user computing. Tools such as Fourth Generation Languages (4GLs) have been produced for -- or sometimes more accurately appropriated by -- end-users to reduce the need for conventional programming skills. According to recent figures, end-user programming is growing as a proportion of application software development, though still failing to match the predictions that first accompanied it (Sedacca 1984). This may be explained in part by the resistance of IT support staff.

There may also be other motivations for limiting end-user autonomy for which such technical arguments are a convenient smoke screen, the more so because it is

[5] This is a finding from a preliminary study of social learning processes in CSCW. Systems administrators of a large UK University Computing Department were interviewed.

generally difficult for end-users to challenge them. These reflect the politics of organisational divisions of labour, knowledge and control (Greenbaum and Kyng 1991).

Relationships Between End-Users and IT Support Staff

It makes little sense to draw the line between what users can do to change their IT systems (which for simplicity we will henceforth refer to as customisation) and implementation -- the role performed by IT support staff -- purely on the basis of some set of abstract criteria relating to the respective technical complexities of the tasks, or to assume that technical complexity is the only barrier. More relevant to our concerns here is that what constitutes end-user customisation is defined through the set of practices that evolve within a particular environment of use: customisation may be defined to be whatever actions end-users are officially sanctioned to undertake on their own initiative; the rest, by default, must be considered to be implementation by virtue of being the prerogative of support staff.

It is important to consider just how customisation differs from other kinds of change that goes on during the system life-cycle. When analysed from the perspective of factors that trigger them, these changes appear to be of many different sorts. Some may be necessitated because "bugs" have been discovered, others may be triggered by technology updates and developments, and yet others by individual and social learning. From the perspective of sustaining a viable service for end-users, however, it is not the triggers that are important, but that each change event is dealt with in an effective manner. Whether end-users or support staff take responsibility reflects the practices of a particular organisation. Similarly, this division of labour may be rigidly enforced within some organisations, but extremely flexible within others.

What is lacking in discussions of customisation practices is an account of how this division of labour -- "tailoring culture" (MacLean et al. 1990) -- gets defined, and the alternative means end-users have for pursuing changes which are beyond their compass. These are questions about the relationship between end-users and support staff, and the power end-users may have to achieve customisation of their environment indirectly through the agency of support staff. In this context, human facilitators like the "handyman" -- someone who is able to "communicate users needs to programmers for longer term or more complex development" (MacLean et al. 1990, p. 176) -- would seem to have far greater relevance than customisation tools.

It is the practice in most UNIX[6] systems for only systems administrators to have so-called 'super-user' or *root* access privileges since this grants the power to make changes to any part of the system. With some notable exceptions (e.g. de Leon, Rodriquez & Thompson 1993) most UNIX systems administrators would regard giving end-users root access as the first step towards chaos. The history of UNIX's development, and the original philosophy behind it, makes an interesting contrast with present-day attitudes. UNIX was conceived as an adaptable operating system which would provide its users with a small, but powerful, collection of software tools (Kernighan & Pike 1984). These were designed on a construction set principle so that the user could easily create a new tool from the existing set to meet a particular requirement. In addition, in UNIX users were actively encouraged to contribute ideas

[6] UNIX is a trademark of Bell Laboratories

and software to its development. In many respects, therefore, UNIX is the product end-user innovation[7].

By limiting end-users' access privileges, systems administrators ensure that their cooperation is indispensable for a significant range of changes, irrespective of end-users' technical competence. Effectively, therefore, systems administrators are in a position to act as gatekeepers of change, and so where end-users' needs bring them into conflict with formally or informally agreed norms, then the barriers identified by Mackay (1990) take on an entirely different form. End-user behaviour can be controlled in ways which stop short of directly curtailing their freedom act. For example, technical support and advice may be limited to a specific set of applications software; end-users are then free to deviate from these implied defaults, but can only do so at the cost of having to fend for themselves when problems arise.

Tailoring culture varies from site to site. Some systems administrators in our study take the view that end-users should (in principle) be free to do what ever they like; others believe that it is necessary to police and control end-users closely. Clearly, factors such as the type of organisation (e.g. academic or commercial), end-user profiles and the day-to-day interactions between end-users and systems administrators contribute to the degree of trust and understanding between the two groups, and hence to the ease with which they cooperate. For some systems administrators, ensuring that end-users get to know about -- and to take advantage of new developments -- is a key part of the job. These are likely to take a more pro-active role in assisting end-users to adapt and improve IT systems, and may even wish to take a lead in advocating changes. For systems administrators, the main priority is to merely to maintain an adequate service day-to-day. These may be more likely to take the view that end-users' aspirations have to be controlled.

By way of a summary of these issues surrounding innovation in use, we can point to a number of issues that deserve closer examination: in particular, how customisation is negotiated between end-users and support staff. Mackay's (1990) analysis of innovations in use largely focused upon as an activity taking place between end-users'. We suggest that this picture is incomplete without consideration of the relationships between customisation and other forms of change, and of factors that are treated as exogenous within Mackay's account. The evolution of organisational computing environments is a social process, and end-users are not the only, or even necessarily the most important, contributors.

More profoundly, there is little information about the other critical element of the innofusion process in relation to CSCW: feedback of useful innovations arising within the implementation arena to future supply of artefacts. The lack of supplier-user links that might facilitate such feedback has been identified in relation to other technologies (Fleck 1988; Webster & Williams 1993). Their lack in relation to CSCW may not be surprising -- particularly given the short history of CSCW. However in the absence of such mechanisms, the lessons learnt by users and others trying to adapt technologies to their purposes is not being properly utilised. This is an issue that therefore requires further attention in both research and policy terms.

---

[7] According to some of its critics, this is root of many of UNIX's shortcomings (see for example Norman 1981).

Discussion about customisation has frequently seemed to be presented as a panacea for problems of usability, and has failed to specify the scope for, and limits to, customisation. For example, the rhetoric of customisation has been applied to relatively superficial personalisation of interfaces by end-users, with the misleading implication that everything could be up for change. However, it is essential in many organisational IT systems for some limits to be placed on end-user innovation -- for example to protect key information structures and systems. The way forward for CSCW will not be found solely by resort to end-user customisation, nor by fully comprehensive sociologically-informed design. Instead, what is needed is a well-judged combination of different measures. For example, the benefits of commodified supply will involve the standardisation around some established CSCW components. This is particularly the case when we consider the advantages of standardisation around user interfaces (for example the success of MicroSoft points to the increasing importance of the user-interface as a arena of standardisation).

If we are to assess the prospects for a technology such as CSCW, and the way it can best be carried forward, we must take into account not just processes of initial design but also the important (albeit highly-distributed) social learning processes involved in its implementation, and the mechanisms whereby innovations in use can be carried forward into future supply. This should also include also market processes.and their role in selection of products. While the bulk of attention around CSCW has focussed upon improving design models for CSCW technologies, we must bear in mind that current knowledge does not allow us to determine which will carry the greatest prospects for the commercial success and widespread adoption of future technologies, between on the one hand, the expenditure of a major effort in generating richer accounts of users and uses and embedding these in in the design of technologies, and on the other, the launch of a variety of different options, and allowing end-users to express their preferences through the market.

Our exploration of the complex character of social learning processes in the adoption of technology highlights the enormous difficulties and uncertainties inherent in attempts to predict which products and approaches will succeed. We need merely consider the very uneven success of different instances of IT. For example the dramatic success of fax in recent years (surprising since this technology had been available but not widely used for many years) contrasts with the slow initial spread of email (despite predictions that this technology would take off rapidly for several decades, email has until recently remained the preserve of a small technophilic elite). Complex organisational IT applications, such as CAPM systems, frequently failed, despite their widespread promotion in the late 1980s as a ready technological fix to current commercial problems and as a stepping stone to Computer Integrated Manufacture (Webster & Williams 1993). The difficulties we experience in conceiving future technological capabilities are substantial, but arguably much less intractable than our abilities to pre-conceptualise changes in society and in organisational requirements for technology.

References

Brady, T., Tierney, M. & Williams, R. (1992) "The commodification of industry-application software", Industrial and Corporate Change, Vol. 1, No. 3, pp. 489-513.

Bray, P. (1992) "The Software Time Bomb", Which Computer, September.
Briefs, U., Ciborra, C., & Schneider, L. (eds.) (1983) "Systems design: for, by and with the users", North Holland, Amsterdam/New York.
Bullen, C. & Bennett, J. (1991) "Groupware in Practice: An Interpretation of Work Experiences", in Dunlop, C. & Kling, R. (eds) "Computerization and Controversy", Academic Press, New York, pp. 257-287.
Carasik, R. & Grantham, C. (1988) "A Case Study of CSCW in a Dispersed Organisation", Proceedings of the ACM Conference on Human factors in Computing Systems (CHI'88), ACM Press, pp. 61-66.
de Leon, L., Rodriquez, M. & Thompson, B. (1993) "Our Users Have Root!", Proceedings of the USENIX Systems Administrators Conference, Monterey, ACM Press, pp. 17-24.
Ellis, C., Gibbs, S. & Rein, G. (1991) "Groupware: some issues and experiences", Communications of the ACM, 34, 1, pp. 39-58.
Fincham, R., Fleck, J., Procter, R., Scarbrough, H., Tierney, M. & Williams, R. (1994) "Expertise and Innovation: Information Technology Strategies in the Financial Services Sector", Oxford University Press.
Fleck, J (1988) "Innofusion of diffusation: the nature of technological development in robotics", Edinburgh PICT Working Paper No. 4. Edinburgh University, Edinburgh.
Fleck, J. (1993) "Configurations: Crystallizing Contingency", International Journal of Human Factors in Manufacturing, Vol. 3, No. 1, pp. 15-36.
Fleck, J., Webster, J. & Williams, R. (1990) "Dynamics of Information Technology Implementation", Futures, July/August, pp. 618-640.
Flores, F., Graves, M., Hartfield, B. & Winograd, T. (1988) "Computer Systems and the Design of Organisational Interaction", ACM Transactions on Office Information Systems, 6, 2, pp. 153-172.
Greenbaum, J., & Kyng, M. (eds) (1991), "Design at Work: Cooperative Design of Computer Systems", Lawrence Erlbaum.
Hamilton, R. (1986) "DDP -- Management Opportunities and Consequences", Omega, 14, pp. 475-481.
Grudin, J. (1988) "Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces", Proceedings of the Conference on Computer-Supported Cooperative Work, ACM Press, pp. 85-93.
Henderson, A. & Kyng, M. (1991) "There's No Place Like Home: Continuing Design in Use", in Greenbaum, J. & Kyng, M. (eds) "Design at Work: Cooperative Design of Computer Systems", Lawrence Erlbaum, pp. 219-240.
Hughes, J., Randall, D. & Shapiro, D. (1991) "CSCW: Discipline or Paradigm?", Proceedings of the Second European Conference on Computer-Supported Cooperative Work, Amsterdam, pp. 309-323.
Kernighan, B. & Pike, R. (1984) "The UNIX Programming Environment", Prentice-Hall, New Jersey.
Kling, R. (1991) "Cooperation and Control in Computer-Supported Work", Communications of the ACM, 34, 7, pp. 83-88.
Mackay, W., (1990) "Users and Customizable Software: A Co-Adaptive Process", Doctoral Dissertation, Sloan School of Management, MIT.
MacLean, A., Carter, K, Lovstrand, L. & Moran, T. (1990) "User-tailorable systems: Pressing the issues with buttons", Proceedings of the ACM Conference on Human Factors in Computing Systems, Seattle, ACM Press, pp. 175-182.
MacLean, E., Kappelman, L. & Thompson, J. (1993) "Coverging End-user and Corporate Computing", Communications of the ACM, 36, 12, pp. 79-92.

Mantei, M., Baecker, R., Sellen, A., Buxton, W., Milligan, T. & Wellman, B. (1991) "Experiences in the Use of a Media Space", Proceedings of the ACM Conference on Human Factors in Computing Systems, New Orleans, ACM Press, pp. 203-208.

Nardi, B. & Miller, J. (1990) "The Spreadsheet Interface: A Basis for End User Programming", Proceedings of the Third International Conference on Human-Computer Interaction, Cambridge, North Holland, pp. 977-983.

Norman, D. (1981), "The Trouble with UNIX", Datamation, November, pp. 139-150.

Procter, R. & Williams, R. (1992) "HCI: Whose Problem is IT Anyway?", Proceedings of the IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction, Ellivuori, North-Holland, pp. 385-396.

Quintas, P. (ed.) (1993) "Social Dimensions of Systems Engineering: People, Processes, Policies and Software Development", Ellis Horwood, Hemel Hempstead.

Randall, D., Hughes, J. & Shapiro, D. (1993) "Systems Development - the Fourth Dimension: Perspectives on the social organisation of work", in Quintas, P. (op. cit.), pp. 197-214.

Robinson, M., (1993) "Design for unanticipated use ...", Proceedings of the Third European Conference on Computer-Supported Cooperative Work, September, Milan, Kluwer, pp. 187-202.

Sedacca, B. (1984) "Fourth Generation Languages", Systems International, November, pp. 113-115.

Suchman, L. (1993) "Do Categories Have Politics? The Language/Action Perspective Reconsidered", Proceedings of the Third European Conference on Computer-Supported Cooperative Work, Milan, Kluwer, pp. 1-14.

Webster, J. & Williams, R. (1993) "Mismatch and Tension: Standard packages and non-standard users", in Quintas, P. (op. cit), pp.179-196.