
Computer Supported Cooperative Work **12:** 465–490, 2003.

The Appropriation of Interactive Technologies: Some Lessons from Placeless Documents

PAUL DOURISH
Department of Information and Computer Science, University of California Irvine, Irvine CA 92697, USA (E-mail: jpd@ics.uci.edu)

Abstract. Appropriation is the process by which people adopt and adapt technologies, fitting them into their working practices. It is similar to customisation, but concerns the adoption patterns of technology and the transformation of practice at a deeper level. Understanding appropriation is a key problem for developing interactive systems, since it critical to the success of technology deployment. It is also an important research issue, since appropriation lies at the intersection of workplace studies and design.

Most accounts of appropriation in the research literature have taken a social perspective. In contrast, this paper explores appropriation in terms of the technical features that support it. Drawing examples from applications developed as part of a novel document management system, it develops an initial set of design principles for appropriable technologies. These principles are particularly relevant to component-based approaches to system design that blur the traditional application boundaries.

Key words: appropriation, customisation, deployment, design, document management, flexibility, visibility

1. Introduction

For many years, interactive system research has explored the question of customisation – how interactive technology can be specialised to the needs of particular users or settings. For almost as long as we have been developing interactive systems, we have been developing ways to make them customisable. Customisable systems can be adapted and tailored by their users, to fit them to the different situations in which they might be used.

Although customisation was originally studied in the context of conventional single-user systems, it has also been a common topic of concern in multi-user settings and Computer-Supported Cooperative Work. Like individuals, different groups operate in different ways and in different settings, and may need to adapt the technologies to suit those circumstances. However, in addition, groups are themselves made up of individual users, who might each want to work in different ways. So, in looking at customisation in CSCW, researchers have explored the need to pay attention to balancing the needs of the group against the needs of the individuals who make up that group (Greenberg, 1991).

Studies in CSCW have also shed new light on customisation in single-user technologies, revealing that, even in those settings, it is an inherently collaborative phenomenon. Customisations emerge within workgroups and social settings. In developing a tailorable technology called Buttons, MacLean et al. (1990) discovered that a critical element in the successful deployment of the system was the establishment of a "tailoring culture," an understanding on the part of the participants that it was acceptable to tailor their Buttons, and that making changes was acceptable organisationally (tailoring was allowed), technically (the system would not break), and socially (people would not be offended if you changed something they wrote). Similarly, MacLean and his colleagues worked to encourage users to share Buttons with their colleagues, to look for ways to improve them, and so on. Establishing tailoring as an acceptable social practice was critical to the success of the system.

In a similar vein, Mackay (1990) looked at how customisation occurs within social networks. She studied the patterns of diffusion and sharing of software customisations within organisations, and mapped out the paths that particular customisations followed from one organisation to another. By doing this, she traced out the social networks through which customisations were shared, and identified various key roles in the adoption of customisations and customisable technologies within organisations and social systems. Like the Buttons work, this empirical study highlights the collaborative nature of customisation practices. Customisation is always a collaborative phenomenon.

However, there is a stronger position that we can take on the relationship between customisation and collaboration, and that is to see customisation as *inherent to* collaboration (Bentley and Dourish, 1995). From this perspective, customisation is a feature of all collaborative practice. My argument here proceeds from the large body of studies of work practice in CSCW. These studies have repeatedly demonstrated the intricate relationship between work practice and the detail of the settings in which it emerges. This "situated action" perspective, as introduced by Suchman (1987), has become one of CSCW's dominant perspectives on human action. Drawing on ethnomethodological foundations, the situated action perspective looks on the sequential organisation of action as a moment-by-moment improvised affair, emerging in response to the circumstances of its production – physical circumstances, social circumstances, organisational circumstances and so forth. Everyday action continuously incorporates elements of those settings in order to accomplish the work at hand. So, features of the setting and the artifacts around which working practices is organised are continually reconfigured, repurposed and incorporated into the way in which those practices develop. Since interactive technologies are part of these settings, they are also implicated in and subject to this reconfiguring and repurposing. It is in this sense that I suggest that customisation is inherent to collaborative activity. The ongoing, incremental adaptation of interactive technologies is inherent to the emergence of practice, and practice is inherently shared.

This is a broader view of customisation than the traditional perspective within HCI; to avoid confusion, I refer to it as *appropriation*. 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.

In this paper, my goal is not to propose, describe or develop a theoretical framework. Similarly, my use of the term "appropriation" is not the same as that developed in Adaptive Structuration Theory and described elsewhere in this issue. Instead, my goal here is to show how the problems of appropriation are technical problems, and begin to work through some of the consequences of that position.

By technical problems, I mean problems that arise out of the fundamental structure of the technologies from which software systems are constructed. The problems, then, are inscribed into every software artifact. It follows that the solution to these problems must involve some kind of transformation of the technology out of which software systems are constructed. In CSCW, we have typically explored appropriation from a social perspective. So, for example, investigations such as Orlikowski's study of the adoption of Lotus Notes (Orlikowski, 1992, 1995), Grudin and Palen's work on calendar systems (Grudin, 1988; Grudin and Palen, 1995; Palen, 1999), and Button et al.'s study of workflow systems (Bowers et al., 1995; Button and Sharrock, 1997) have uncovered a range of social and organisational issues contributing to the success and failure of CSCW system deployment. However, appropriation relies on flexibility in both practice and technology, and in particular, flexibility in the way in which the technology can be mapped onto user needs. While the social investigations have begun to illuminate issues around the flexibility of working practice, we need at the same time to explore the flexibility that is required of technologies, and how it might be provided. So the question I want to ask here is, what features of technological design support appropriation? And so, how could systems be designed in order to accommodate, support and encourage the process of appropriation?

There are two reasons that these are important questions. The first is as a practical matter. In CSCW, we would like to be able to develop technologies that help people work. Appropriation is endemic to collaborative work; it is a regular part of the picture. So we must be able to build systems that deal with appropriation. Developing a deeper understanding of the technical features that support appropriation can help us build more appropriable technologies, as well as helping us analyse the problems that accrue when technology is not appropriable. The second important aspect is as a methodological matter. Appropriation lie at the intersection of technical design and social practice. Investigating appropriation and developing an understanding of its consequences for technical design is a way to provide a stronger link between sociological studies of working practice and technological investigations of design in CSCW.

As a focus for this discussion, I want to use the example of a document management infrastructure called Placeless Documents (or just "Placeless").[1] Placeless Documents is a technical infrastructure for managing documents and for building document-based applications. It is based on a simple and extensible model that provides a novel architectural approach for application development, as well as a novel conceptual model for end user document management. The Placeless Documents system serves two roles here. The first is that it crystallises the problems of appropriation by letting us focus on one particular area, that of document interaction. Drawing on existing studies of problems in managing document collections, the Placeless system offers a radically different approach to organising information. The second role is that, as a case study, it provides some insight into the problems of appropriation as technical problems. Our attempts to capture difference styles of application use in Placeless have been yielded different degrees of success; some of the lessons that we can draw from it apply very broadly.

Although I will give the Placeless Documents system a central position here, my concern is not, in the end, with Placeless itself. This is not an attempt to evaluate a particular technology. Instead, I want to use this system and the conceptual approach that it embodies as an occasion to reflect on the relationship between technological design and work practice. In particular, I want to explore how the assumptions that lie behind our technical designs can help or hinder users in fitting technical systems into their work. I would suggest that traditional monolithic or application-oriented models of system design embody a set of assumptions that can prove problematic, and that emerging component-based alternatives offer opportunities to incorporate new models of user activity. So this is not simply a question of "this design" versus "that design"; it is a question of the assumptions that lie behind the process of design itself.

2. Information management and placeless documents

Much of what we do when interacting with computer systems is information management. By "information" here, I mean items such as email messages, files, folders, financial records, digital images, web pages, database records, contact information, calendar entries and so on. By "management", I mean the various tasks that these items require – sorting, searching, clustering, creating, monitoring, etc.

Given the range of different information processing tasks that we carry out, and the range of ways that these tasks are integrated into broader individual and collaborative working practices, we find that the tools given to users to perform these tasks are typically very limited. In fact, almost all information systems are based on a single information structure – the hierarchy. Hierarchies pervade information systems of all sorts, from the filesystem (perhaps the most widespread information management device used on conventional computer systems) to email

systems, contact managers, and Web browser bookmarks. Users are forced to find ways to map their information needs onto this single structure.

A variety of studies have shown the problems that users face in trying to perform a wide range of information tasks with this simple structure. Barreau and Nardi (1995) and Kaptelinin (1996), in separate studies, examined how computer users employ the desktop and the filesystem to store personal information and support their individual tasks. Their studies point to generic problems with the use of fixed structures such as hierarchies, such as the requirement that information be placed into the structure before its use has been determined, the difficulty of reorganising information once it has been stored, and the mismatch between the system's insistence that information be in only one place and users' understandings that information can play many different roles for different tasks.

Despite these sorts of problems, Kaptelinin describes a range of creative approaches and work-arounds developed by the users he studied – conventions by which information is managed and the mappings between working tasks and data storage can be maintained. What Kaptelinin's study vividly demonstrates is that the creation of personal information structures and procedures for managing them is an example of appropriation that is carried out continually by computer users who need to find a way to map between the system's features and their own needs. Even just saving files involves creating a meaningful structure of files and folders that in some way accommodates the tasks into which those files might fit.

The object of the Placeless Documents project was to explore architectures for information management that could overcome some of the major problems with hierarchical representations and accommodate the wide range of ways in which people appropriate information structures to accommodate their different needs. Our goal was to develop not simply an application for information management tasks, but also a platform for developing novel information-based applications. Since it is the conceptual model at the heart of this platform that is most relevant here, I will focus my discussion on that. A more technical description of the system can be found elsewhere (Dourish et al., 2000).

2.1. DOCUMENT PROPERTIES

Our paradigm for document management is based on *document properties*. Properties are features of the document that make sense to users (e.g. it's a paper; it's about Placeless; it's from Keith; it's a draft; it's due August 12). In our implementation, properties are arbitrary name/value pairs that can be associated with documents by users either directly or through applications. Placeless itself imposes no structure or requirements on the properties of a document. By "attaching" properties to documents, users can express arbitrary features of the documents that are consequential for the way in which they want to interact with them, and create arbitrary associations or groupings between documents.

Conventional filenames often encode the properties of a document. For instance, when I store a document at "C:\Dourish\Papers\In progress\JCSW\Placeless.fm", I am indicating that the document is owned by Dourish, is a paper, is in progress, is for JCSCW, is about Placeless, and is in Framemaker format. Document properties allow us to capture the same sort of information codings while also addressing observed problems with personal information management. For example, since documents can have many different properties, users can express that documents play multiple roles; since properties can be added at any time, it is easy to file documents incrementally and to revise their codings; since the system allows any set of properties to be attached, different organisational schemes can be applied; and since properties are independent of each other, there is no need to refer to them in some particular order.

2.2. ACTIVE PROPERTIES

We would like to be able to use properties for more than simply organisation. We would also like to be able to use them to control documents. In the everyday world, after all, the features of documents have consequences; the fact that a document has a deadline of August 12 means that I must remember to work on it before that date. So, we would also like to be able to control how documents behave according to the properties that users attach to them.

To do this, the Placeless Documents design incorporates *active* properties. Active properties are name/value pairs just like normal properties, but also include runnable code that will be executed when operations are performed on the document. For instance the property "log access=true" might incorporate code that will intercept read and write accesses and make a note in a log file. There are two sorts of active properties provided by Placeless. "In-line" active properties either existing document operations (e.g. read and write content, add properties, add to collection, etc.), while "delegates" augment those operations to provide specialised behaviours that would not otherwise be accessible (e.g. translate to French, synchronise with laptop, publish on Web). Just like static properties, any number of active properties can be added to documents by users directly or through applications; so, active properties give end users composable control over document behaviour.[2]

2.3. UNIVERSAL AND PERSONAL PROPERTIES

In the everyday world, documents have different relevances to different users. A purchase order on my desk might, to me, represent the new PC I just ordered; to my manager, it is an element for her budget, and for our technical support staff, it indicates an upcoming installation task. As a result, we need to be able to express different features of how the document will fit into our practices, configure different actions for documents, and have documents appear in different parts of our filing structures. Accordingly, in Placeless, documents have both *universal* properties

(which are true for and visible to everyone) and *personal* properties (which are true for and can be restricted to individuals). So, when I say that a document is important, that need not be true for everyone. Different properties can be associated with different users although the underlying document is the same (and maintains its identity across different users).

2.4. CONTENT

Properties are the central feature of the Placeless design. However, properties are simply metadata, while for most users, the most important feature of documents is their content. Clearly, Placeless needs to manage document content in some way. We did not want Placeless to be "yet another place to put your content", along with web servers, ftp servers, intranet servers, and so on. Similarly, we did not want to reproduce the content management features offered by other infrastructures. Instead, we want to be able to incorporate content in external "repositories" (such as existing databases, filesystems or the Web) and incorporate it seamlessly into Placeless.

This presents two problems, the "legacy repository" problem and the "legacy application" problem. The legacy repository problem is the problem of incorporating existing content. Our solution is to attach a specific sort of active property known as a *content provider* to each document with content. The content provider can be thought of as an active property that implements the content read and write operations; it encapsulates an access protocol for an existing repository, including local and remote filesystems, the Web, IMAP mail servers and commercial document management systems. Whenever a user reads and writes content in the Placeless system, the content provider seamlessly mediates between the Placeless application and the external repository.

The legacy application problem concerns what happens to that content; how can it be routed to applications? We would like people to be able to use their existing applications, even though those applications do not use the Placeless programming interface. Some applications access content through well-established network protocols (e.g. Web clients use HTTP, while mail clients use IMAP), and for these, we provide server components in Placeless that provide implementations of those protocols. Many other important applications, like Microsoft Word, do not operate through network protocols; instead, they read and write directly to the filesystem. For these, our solution is, essentially, to *be* the filesystem. By providing an implementation of a networked fileserver protocol, the Placeless system can make its content available *as if it were* a filesystem. Clients "mount" the Placeless document space as part of their filesystem, making all Placeless content available to filesystem-centered legacy applications. This also means that conventional browsing tools, like the Windows Explorer, can be used alongside Placeless browsers to view content, copy files, and so on.

3. Placeless and appropriation

The Placeless Documents system was functional and employed in daily use for around two years. It was used by its own developers as a basis for both programming and for day-to-day information needs (e.g. it provided the primary email infrastructure for a group of around 10 people for some time). In addition, it was used by other research groups as a basis for their own development exercises, within Xerox and externally. Further, it was used in collaboration with others to develop specific application solutions to research problems (some of which will be discussed later).

The discussion to follow will concentrate on how the design features incorporated into Placeless provide a basis for appropriation. This is based in a range of different sorts of experiences with the Placeless Documents system. It plays two roles in these examples. Primarily, it is an infrastructure for developing document and information applications; in addition, it is a conceptual model that is embodied by end-user applications. Even though end users interact with Placeless through mediating applications, the basic Placeless model shows through many of these. So, the lessons from Placeless Documents come from two primary experiences: first, the experience of turning to Placeless as an alternative to conventional information infrastructures for solving development problems; and second, the experience of users with the Placeless Documents model as embodied by specific prototype applications. Through both of these, this paper will explore a more fundamental question: the interactional and collaborative consequences of the alternative information model that Placeless embodies.

Before going on to explore some particular cases of the use of Placeless Documents, it is worth pausing to reflect on the relationship between the Placeless Documents design and appropriation.

We have already seen appropriation at work in the ways in which people adopt and adapt the information structures presented by technical systems to make the appropriate for their work. This is a widespread phenomenon; some researchers such as Mackay (1990) or Trigg and Bødker (1994) have pointed out the ways in which the role of integrating work practice and information systems is sufficiently important as to be systematically worked into organisational arrangements.

Studies such as those previously cited of Barreau and Nardi or Kaptelinin point out the barriers that conventional information and file systems can present to effective appropriation. Consequently, we designed Placeless with an understanding of the need to accommodate appropriation explicitly. By integrating with legacy content and legacy applications, we provide a means to incorporate Placeless applications into existing working practices. By organising documents in terms of properties rather than hierarchies, we allow users to establish multiple different ways of organising documents for the different tasks they may want to perform. By making properties entirely open-ended, we try to allow users to collaboratively and continually revise how they organise and view their work. By separating personal from universal properties, we decouple the interactions that different people's

views would imply. These features are provided to make it easier for users to appropriate Placeless Documents and to incorporate it into their evolving structures of work.

Placeless offers both end user applications and an infrastructure for developing document applications. However, in the course of this paper, I want to consider appropriation neither in terms of the browser applications that Placeless offers, or in terms of the platform features that developers might exploit. Instead, I want to consider how Placeless's *basic conceptual model* – an unstructured soup of documents annotated with arbitrary properties – can support appropriation of Placeless-based systems and can help users find ways to express their information needs in terms of the technical features available to them.

With that in mind, the following sections will present two case studies of information management that we tackled using the Placeless Documents framework. The case studies serve three roles here. The first is to provide concrete illustrations of the role of appropriation in information management. The second is to highlight how problems in appropriation are rooted in conventional approaches to information infrastructure. The third is to show how we can apply the novel approach that Placeless embodies to solve these problems and provide more explicit support for observed practices of appropriation.

4. Case study #1: Collaborative document management

The first case study explores the collaborative document practices of an engineering organisation. This is based on ethnographic materials collected by colleagues at PARC as part of a series of studies of "working document collections" (Blomberg et al., 1997; Trigg et al., 1999).

The engineering group they studied was engaged in the design of a bridge, but at this early stage of the project, the principal focus of their attention is not the bridge itself but the large collection of documents relating to project work. This includes reports, letters, petitions, permits, engineering drawings, maps, newspaper articles and other documents relating to all aspects of the project. A similar set of documents accompanies all projects in this organisation; they are referred to as "Project Files" and an intricate set of organisational practices govern the ways in which Project Files are used and organised. In particular, the organisation lays down a specific categorisation scheme by which documents are coded and located within the project files. This scheme is called the Uniform File System or UFS.

Although the UFS is specifically designed for the needs of this organisation, there are, none the less, a variety of problems that attend its use. In the course of filing documents, members of the organisation need to find ways to make the UFS relevant for their immediate concerns. The fact that they have such problems is not a feature of the UFS per se; it is a feature of categorisation schemes in general. The UFS is a representation of the organisation's world, and, as Gerson and Star (1986) observe, "no representation of the world is ever complete or permanent."

So, in the course of filing or searching for documents, members of the organisation encounter, recognise and deal with a variety of problems putting the UFS to work. For instance, in the course of filing a document, one engineer comments:[3]

> So there's all these categories it could conceivably go under and I have to pick one [. . .] certainly my assessment may be different than the guy next aisle over

Or again:

> Ok now I don't see what I thought I was looking for. So, uhm, I guess I would stick it under uh Floodplain Evaluations. What was the other spot? Drainage is usually done during the design phase and we're not there yet. So that's why I would pick, uhm, but see 231 is Draft Environmental Document which is pretty vague. So I'll never find it. It's just not going to happen. I'd probably be more inclined to stick it under Drainage even though that's not where it belongs? So that's what I'm going to do.

So, in the course of working with the UFS, members of the organisation orient towards a set of problems with the way the UFS organises their work; problems of filing, recognition and retrieval. The problems of using the UFS are also visible in the organisation's workspace, where printed or typewritten copies of the UFS – the official set of categories to be used to file documents – carry handwritten annotations recording how each specific group has amended it or established conventions that govern how they will use it.

The issue of "the guy next aisle over" is particularly salient for this organisation because the project files (or a significant percentage of them) will move with the project through the various phases that make up its lifecycle. These phases may be carried out by different sets of people. So it may well be the guy the next aisle over, or the guy in another building years later, who will need to locate and retrieve the document that the engineer is filing. The local practices that workgroups establish to manage their documents and their practical filing problems may interfere with the organisational need to be able to find documents again later, and to account for and reason about filing practices.

Our engagement with this organisation was as part of their exploration of the consequences and opportunities that online project files might offer. One prototype had already been constructed to help them with the transition from paper to online and to offer them new models for document retrieval (Trigg et al., 1999). As a separate effort, we explored the Placeless Documents model as the basis of a prototype system for this organisation, and focussed in particular on the problem of customisation and mutual intelligibility. Our concern was, given that each individual and group customises the UFS to their own particular needs, how can we support the mutual intelligibility of the document space necessary for effective retrieval?

4.1. THE UFS AS A COLLABORATIVE ARTIFACT

The ethnographic material, and in particular the observation of the variety of ways in which the UFS functions in the work of engineers interacting through the project files, suggests the key design element around which our solution, dubbed "Macadam", is designed. The observation is that the UFS is not simply a resource through which the work of filing is conducted; it is, itself, *part* of that work. It is an object of collaborative activity, in the way in which it is subject to collective adaptation within and between working groups to make it appropriate to the work at hand. Our design reflects this; in Macadam, the structure by which the documents are organised is, itself, part of the same workspace through which the documents are shared.

Our technical approach is described in more detail elsewhere (Dourish et al., 1999b). Conceptually, Macadam's representation of the categorisation hierarchy can be thought of as a stack of translucent sheets, each of which describes part of the hierarchy. Each sheet modifies the sheets below, but when they are viewed together, the sheets collectively present a single category structure.

In this representation, the base "sheet" corresponds to the organisation's view of the category structure – the official UFS. "On top", we place a new sheet that describes the changes that a specific project might introduce to the UFS for its own purposes. Over that, a further set of changes record the ways in which individual work groups might operate, while a fourth sheet introduces personal changes for workgroup members. When a document is categorised, the system records the *context* in which the categorisation took place, that is, the set of "sheets" in play at the time.

The key feature of this approach is that customisations to the category structure are recorded in relative rather than absolute terms. Instead of replacing the existing structure with a new version that incorporates whatever changes the user has requested, the system instead records the set of changes that the user has made, and the structure to which those changes were applied. So, a set of customisations might comprise the renaming of one node, the moment of part of the tree from one place to another, and the addition of a new set of leaf nodes. The advantage of this approach is that specific sets of changes can be turned on and off. The system can show the structure either with or without a particular users' or groups' changes. Using the metaphor of the stack of translucent sheets, the system can add and remove sheets to show different versions of the category structure. This allows the system to create multiple views of the document – perhaps one view in terms of the category structure with the changes, and one view in terms of a category structure without them. These different views correspond to the perspectives of different people and different groups, even though they are all looking at the same set of documents.

By supporting these different views, Macadam balances the problems of customisation and mutual intelligibility. Since project members and groups can create new customisations at any time, the system meets their local need for filing

structures that match the detail of the project they are engaged in and the work they are carrying out. At the same time, since people outside can view the documents in terms of the organisational structure rather than the project's own structure, the problem of intelligibility is reduced. In other words, incorporating new ways of working, and extending the system to accommodate them, need not interfere with the larger organisational requirement for intelligible records. Fluidity in the system supports the evolving nature of work practice and the incorporation of the system into ways of working that is the hallmark of appropriation.

4.2. MACADAM, THE UFS AND APPROPRIATION

What can this example tell us about the relationship between technology and practices of appropriation?

What is being appropriated here is the conceptual scheme by which documents are organised. Although the organisation regards the UFS as a stable feature of its work, we find that it is recruited in a variety of ways to meet the specific and individual needs of each project. However, the natural encoding of the UFS into technical terms – a traditional hierarchical classification – introduces a number of features (such as broad visibility, universal coding, and so on) that interfere with the appropriation of the UFS into local practice.

The design of Macadam allows us to overcome some of these problems, and, critically, it does so through exploiting some of the basic conceptual features embodied in Placeless. In particular, the fact that the static and universal hierarchy of traditional information systems has been replace with a model that is *multivalued* (that is, allowing documents to be coded in many different ways) and *interpreted* (that is, subject to ongoing interpretation and filtering by the system) allows us to accommodate the practices of appropriation that we observe going on around the Project Files. In turn, these both rely on a *separation* between the information being encoded and the encodings (in this case, the properties) that describe it.

These features, then, appear to be important technical concerns that support the process of appropriation. A second case study will reinforce some, introduce others, and examine some more problematic issues.

5. Case study #2: Workflow

Macadam exercised only the static properties in Placeless. The second case study involves using active properties to add dynamic behaviour to documents.

In traditional systems, document functionality is normally locked inside applications. Applications control how we can interact with documents, how they will behave, how their content is managed and so on. In Placeless, by associating this information directly with the document through the use of active properties, we move the locus of document activity from the application to the document infrastructure. The functionality that would otherwise be locked inside the application

is directly associated with the document, which itself becomes active. The active property design allows us to "activate" documents, having them respond to the ways that they are used.

With this model, it is natural to start to imagine how various sorts of document applications, or application areas, can be migrated into the infrastructure and become document services. One candidate is workflow. Workflow is normally manifest in a system as an application.[4] At this level, it offers various features such as routing a document from person to person within an organisation, managing the activity over the document, and coordinating activities according to a process representation. Using active properties, we can provide these functional components in the document infrastructure instead of in application space.

Incorporating workflow functionality into the infrastructure offers a number of advantages. First, it means that workflow functionality can be offered independently of a document's repository. So, for example, the same workflow system can be used to handle documents stored in the filesystem, on the Web, or on a mail server. This last feature is particularly valuable since document practice studies highlight the central role that email messages play as resources for coordinating work (Bellotti and Smith, 2000). Second, the infrastructure approach means that workflow functionality can be offered independently of application. So, rather than having to employ a workflow-specific client, a user can choose to use whatever application they prefer to interact with the document; the application does not need to be enhanced with any workflow features because workflow functionality is added "for free" by the infrastructure. To explore this aspect of Placeless Documents, we developed a workflow system, called Bernoulli, structured in terms of active properties.

Bernoulli is implemented using two active properties, one in-line and one delegate. The in-line active property intercepts the read and write operations. It looks at how people access the content of the document and examines the annotations and changes they make to the content. For structured and semi-structured documents, this analysis allows the system to become aware of significant state changes. For example, when a user fills in the fields of a form, the system can notice this and determine that a work step has taken place; or for program source files, the system can determine when new functions have been added. (For less structured content, an annotation mechanism allows users to express features of the progress of the work.)

When the property recognises a step of this sort, it consults the record of the process to which this document is attached and moves it from one stage to another according to the activity sequence described in the process.

The first property, then, is defined entirely in terms of existing document operations (read and write). The second active property is a delegate. Delegates are active properties that extend the document's API to allow new, specialised operations. In this case, the operations it provides allow the document to be examined in the context of the representation; they allow the process to be queried, and graph-

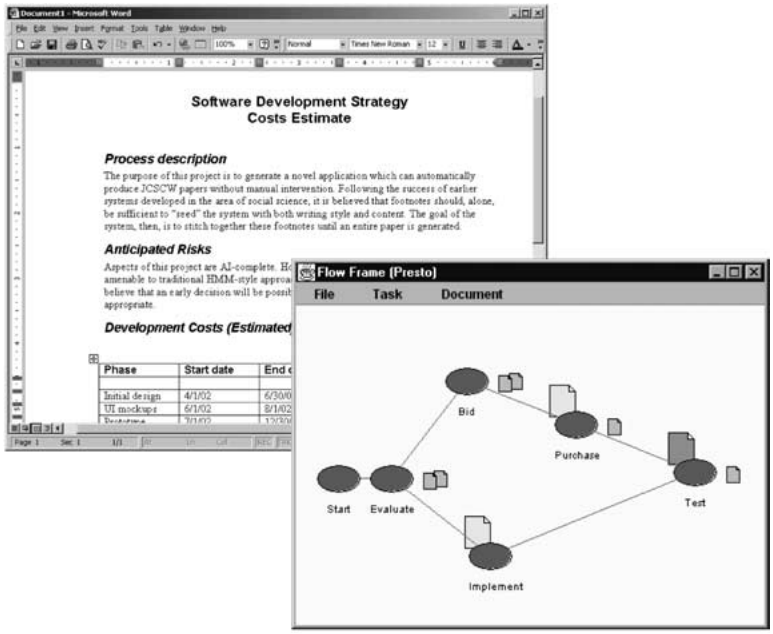

Figure 1. Users can control the workflow application simultaneously through two interfaces, either conventional document tools (left) or a specialised workflow application (right).

ical views with specifically workflow-related information to be constructed. As illustrated in Figure 1, using these two properties lets us provide two sorts of interfaces – traditional interfaces using standard tools (taking advantage of the inline property), and extended interfaces using specialised tools (taking advantage of the delegate property).

5.1. EXTENDING WORKFLOW WITH CONTEXT INFORMATION

The Macadam example in the previous section showed that the flexible property model could be used to capture and represent something of the context in which document operations took place. Exploiting context in this way reflected the findings of work practice studies and made the documents more understandable to people by explaining something of their history. A similar set of features apply to this example.

Traditional workflow systems tend to focus on the content of work but not its context. Process representations highlight the details of *what* gets done (the individual steps or tasks) but exclude information about *how* that work is actually performed (the details of executing each task). The assumption on which they are based is that the performance of one stage of the process is independent of the performance of the other stages; as long the steps are completed in the right order, coordination is accomplished.

However, studies of process-based work in real organisations throw this assumption into question. For example, MacLean and Marqvardsen (1998) report on an investigation of the mortgage process at a British building society. The ethos in this organisation was that they would make every effort to grant a mortgage application if possible. Assessing the loan risk was seen as the process of "building a case" to grant a mortgage application. So, although the process that they followed described the information to be accumulated in support of the application, the critical feature of the process for the organisation was not how information was amassed but how it was *assessed*. What was critical was how the evidence would be weighed, and whether it would make a compelling case. One one the factors contributing to how information is assessed is the question of how it was collected. For instance, an applicant's salary details would support a case in different ways if they were obtained from the applicant's employer or from the applicant himself. Weighing the evidence required an understanding of where that evidence had come from, which was information outside the scope of the process. So, although this organisation was run entirely around formalised processes, the practicalities of the mortgage application process involve a great deal of walking from desk to desk and making telephone calls in order to collect "out-of-band" information about how the process steps had been performed, where information had come from, and so on. The context surrounding the process is as important to the work of the organisation as the process itself, but this contextual information is often missing from traditional process support systems.

Since the Placeless Documents system offers the opportunity to associate arbitrary information with documents through document properties, we incorporated some features to address these problems into Bernoulli. Our solution is called "Perspectives" (Dourish et al., 1999a). At each stage of the process, we record the "resources" from which the process information has been derived. Resource categories can be included in the process description or added by users as they work with the system. Resources are primarily other documents, but may also be links to other people, information sources, and so on. Resources are linked directly to the document, so that they travel with the document as it moves through the process. At any stage, a user can see the resources that were associated with a document at a particular stage in the process. So, this mechanism uses the process representation as an index into the document's history, allowing the context in which each activity was performed to be re-established.

The case of Bernoulli shows that, by capturing contextual information and allowing it to be linked into the application's structures, we can provide better support for the informal practices surrounding a formal description of working processes encoded in an information system. By looking at how a system is embedded within a wider set of practices and organisational needs, we attempt to accommodate the different ways in which it might be appropriated. In this case, then, support for appropriation lies in the use of a single, integrative, open-ended platform through which applications can be related and linked.

5.2. BERNOULLI AND APPROPRIATION

The "Perspectives" example shows appropriation in a different way. The issue here is not how users can impose their own structures on shared information, but rather, how users can incorporate information into their patterns of work.

As before, in order to understand appropriation in this case, and to see the ways in which the technology is incorporated into working practice (such as at the Building Society studied by Maclean and Marqvardsen), we need to assess the way that the technical infrastructure is configured. In this case, the way that the traditional application models creates a separation between the use of a document in one application (e.g. the workflow system) and another (e.g. email, word processor, etc.) interferes with this.

So the key technical feature supporting appropriation in this example is the revision of the conventional relationship between document, activity and application. Traditionally, documents and files are passive entities, while the potential for activity is locked within applications. Further, most documents are designed to operate with a single application. By breaking application functionality into smaller pieces and allowing them to be associated directly with documents (through active properties), we achieve two benefits. First, end users can customise the behaviour of documents directly by associating active properties with them, and hence gaining finer-grained control over the ways in which documents are incorporated into their environments and their work. Second, the functionality becomes *persistently* associated with documents, so that they carry these specialisations with them at all times, even when no "application" is running. Third, it allows documents to reflect and participate more directly in the many different activities in which they may be involved. This component model, then, allows us to break away from an application-centric approach to information processing, and move towards an artifact-centric approach which offers much greater flexibility to information users.

6. Designing for appropriation

The primary concern of this paper is to explore the technical foundations of appropriation. Having identified a number of features of information systems as problematic for the incorporation of technologies into more flexible forms of practice, I described the alternative model embodied in the Placeless Documents system, and presented two examples to illustrate how its conceptual model relates to the problems of appropriation. What we have seen is that Placeless's assault on traditional information models is focused largely on "moving the boundaries" – making information available outside of traditional application boundaries, separating application and activity, and allowing information structures to be more flexibly designed.

One of the Placeless Documents system's design goals was to use this greater flexibility in order to support appropriation and adaptation. So, at this point, it is appropriate to look back at the assumptions that the design embodies and how they fare in light of developing working prototypes for various information tasks.

6.1. ORGANISING INFORMATION

The first hypothesis was that properties would be a good way for users to organise information. Our experiences in a range of applications, including those discussed here, demonstrate that the property model provides a means for people to create information structures that are incremental, revisable, and overlapping. Although it was not our original design intent, we have also found that properties are also a valuable way for *applications* to organise information in ways meaningful to users. So, for example, Bernoulli uses properties to manage document history; by using a property model, document history can be easily combined with other document features without interfering with the inherent functionality associated with the documents being manipulated. The benefits of property-based organisation include compositionality (different schemes can be brought together over the same information without conflicts) and extensibility (new properties can always be added without any requirement for pre-defined structure). However, we have also found that, while properties are a natural conceptual model for information storage and retrieval, they are frequently an inappropriate model for interface design. Increasingly, our applications use more traditional or task-specific forms of presentation for information that is internally represented as properties. So, for example, our email client uses properties on documents to record the state of mail messages, but the user interface presents that information in more conventional terms, using folders, flags and font features to display the organisation and the state of messages. Similarly, although Bernoulli stores process representations and document relationships as properties, the user interface displays a traditional workflow graph diagram and uses proximity to indicate document links.

In a similar vein, Macadam illustrated that the flexible, composable information management structure that Placeless offers allows different structures to be built up over the same underlying information. The principle use of this was to allow the system to accommodate the different perspectives that occurred through the organisation. However, it also supports the evolution of information structures over time, so that as the categorisation structure changes, the information that it describes can continue to be accessible. Being robust in the face of evolving information and practices is critically important in supporting appropriation, since appropriation inherently involves transformations of this sort.

The appropriation design principle we take away from this is that designing for appropriation implies *supporting multiple perspectives on information*. Appropriable systems need to support the different perspectives that different people might have on information, and support them in moving fluidly from one view

to another. In turn, this implies a separation between information and the structures that describe it, so that many structures can apply to the same information. The property model is one means by which this can be achieved.

6.2. COMPOSABLE FUNCTIONALITY

Our second hypothesis was that active properties would allow users to control document behaviour composably.

This is both a technical question and an interactional question. Technically, Placeless clearly supports incremental customisation of behaviour through active properties. Interactionally, we have encountered some design difficulties which would limit the usefulness of active properties in real use. Certainly, active properties – if properly designed – can be combined on documents and provide the end-user with control over the behaviour of the documents. We have demonstrated this with a range of small applications that allow documents to render themselves variably in different circumstances, or to monitor activities (as in Bernoulli). Since this control is associated with the document itself, rather than with a specific application or server, then it persists in the infrastructure, allowing users to state behaviours that they would like the documents to maintain even when a user is not there to be "in the loop."

We have also found that active properties can be exploited by applications, which can essentially delegate some of their behaviour to the document, too. Again, the persistent feature of active properties means that these behaviours stay with the document even when the application is not running, and so the boundaries between application and infrastructure begin to blur.

This blurring means that, essentially, the infrastructure can be configured and adapted to more naturally support particular styles of interaction in which users might want to engage. Similarly, the composable nature of active properties helps support the fact that a single document might be involved in a range of tasks; the process of appropriation might mean extending the range of ways in which a document is used, so this compositional feature is important. However, it comes at a cost. Since active properties insert themselves into the execution paths of pre-defined document operations, the behaviour of those operations is no longer well-defined. Standard document operations may involve non-standard code running in associated active properties. The result is that it is harder for end-users to understand what sequence of code elements their actions might invoke.

The design principle we take away from this, then, is to *preserve visibility* in an appropriable system. People need to be able to understand how a system works in order to understand how to make it work for them. So, we need to allow them to see not just the opportunities for action (the affordances that characterize traditional user interface design), but also the consequences of those actions. Being able to understand the consequences of action is critical to being able to incorporate the system into patterns of work.

It might seen obvious that interaction design should make the activities of the system visible to users, and of course, so it is. However, our traditional approaches to making the system's actions visible depend on knowing in advance what those actions are. Component-based architectures call this into question. When intermediate designers, tinkerers and end-users can combine active components to create new kinds of behaviours, and when those active entities can be appropriated and incorporated into working activities in a variety of ways and a variety of settings, the privileged position of the "designer" dissolves. There is no position from which the actions of the system can be determined – not even, as the case of Bernoulli shows, from within the system itself, given that system boundaries are frequently opaque. The requirement for visibility, then, is transformed into a requirement of a different sort. It becomes a requirement that the components of the system be amenable to some form of internal examination and reasoning, which would allow some model of the system's activity to be constructed "on the fly" (Dourish, 1995).

6.3. GROUP WORK

The third hypothesis we wanted to explore was that properties would provide a good basis for group work, allowing a document repository to support the collaboration and coordination needs of a group of users.

The question here is whether the features of the property model provide benefits that can resolve some of the tensions inherent in personal versus collective information use. The work on Macadam was directed particularly towards one of these tensions – the tension between customisation and mutual intelligibility. Macadam demonstrates that the property model can provide some relief from this problem. So, in attempting to use the system to support workgroup document activities, we have found that properties do indeed offer a number of benefits over more traditional approaches. Again, the compositionality of properties, both static and active, allow members of a group to organise a common document corpus for their different individual needs, while properties also act as points of coordination between them.

However, although we had made explicit provision for collaborative document management in the infrastructure design, through the separation of personal and universal properties, we found the issues to be quite different when we came to build applications for specific settings. Our experience in these settings has been that the issues of information sharing and group access are ones that are intimately influenced by the way in which the group chooses to work. The structure of sharing is task-specific, and the coordination of shared information is an *achievement* of group work, rather than an prerequisite for it.

On the basis of these experiences, then, we draw the conclusion that appropriable technologies should *make information sharing an application matter rather than an infrastructure matter*. Workgroup information sharing is a matter of prac-

tice, not of design, and so applications must be able to present and employ a variety of task-specific sharing mechanisms rather than a universal general model.

This may seem an unlikely conclusion given that systems such as Placeless tend to emphasise the expansion of what the infrastructure can accomplish and diminish the role of applications. In fact, it can be read as a different requirement, to enrich the boundary between application and infrastructure so that applications and users can express (directly or indirectly) their needs to the infrastructure. This allows us to move decisions about "what will happen next" closer to the user. It ensures, then, that the part of the system with which the user is directly interacting is the one which is in control of how the system's resources are deployed in support of the user's activities.

7. Appropriation, meaning and boundaries

Although this paper has spent a good deal of time exploring design aspects of the Placeless Documents infrastructure and application, the primary topic has not been the Placeless Documents system itself, but rather the process of appropriation, the process by which technologies are adopted and adapted by work groups.

A range of studies in CSCW have encountered questions of appropriation in exploring the adoption of collaborative technologies. Harrison and Dourish (1996) use the term to explore the different adoption patterns of two superficially similar applications of video technology, as part of a larger exploration of the relative roles of "place" and "space" in collaborative settings; while Orlikowski's studies of the adoption patterns of Lotus Notes similarly observe patterns of mutual adaptation of work practice and technology (Orlikowski, 1992, 1995).

From a more theoretical perspective, Poole and De Sanctis propose appropriation as one of the elements of their Adaptive Structuration Theory (Poole and De Sanctis, 1990). Adaptive Structuration Theory is a form of Giddens' Structuration Theory (Giddens, 1984) that is specifically adapted to deal with questions of technology adoption and use. Structuration Theory, as a general social theory, is largely concerned with the mutual interactions of social systems, embodied in regularised social practices, and social structures, the rules and resources that guide and sustain action within the system. Structuration Theory argues that structure both constrains and enables social action, and is concerned with the way in which social structures both sustain action and are themselves transformed through it. In Adaptive Structuration Theory, the role that technologies plays in this process is highlighted and elaborated. Poole and De Sanctis introduce concepts such as the *spirit* of the technology (loosely, the expectations on the part of a technology's designers about how it will be used, and the structures that it encodes) and *appropriations* of technology into practice, through which users incorporate specific structural features from the technology into their practices, more or less faithfully to the designers' intent. Adaptive Structuration Theory, though, has primarily been used to analyse the adoption patterns of group technologies. In this paper, I have been making

a more general use of the term "appropriation," and have been concerned with technical and design consequences. What does it mean to say that a technology is appropriated by an individual or a group? What relationship between technology and practice does appropriation propose? What does this relationship imply for interactive and collaborative system design?

The solution is to start not with technology but with practice, and in turn to see practice as more than simply how things get done. Instead, practice concerns the meaning of action. In *Communities of Practice*, Etienne Wenger (1998, p. 51) comments, "Practice is, first and foremost, a process by which we can experience the world and our encounters with it as meaningful." Practice reflects the sets of meanings that can be ascribed to objects and actions over those objects as part of a larger enterprise. As individuals become members of a community of practice, they come to understand and incorporate this system of meaning into the ways in which they see the world and organise their actions within it.

So, appropriation concerns the way in which technology comes to play a role within this system of meaning. In particular, it plays two roles. The first that features of the system itself become meaningful; people develop ways of understanding how the representations that the system might offer are consequential for their work, and how these representations incorporate and refer to other meaningful entities (people, documents, appointments, or whatever). Secondly, technology conveys meaning. That is, the system is a means by which people can see (and then interpret and understand) the actions of others, whether that action is represented explicitly (as is often the case with "awareness" technologies in CSCW) or implicitly (reflected in the state of the system).

Placeless Documents, then, turns out to be a particularly useful vehicle to explore questions of appropriation, since its principal role for end-users is to create information structures. Information structures inherently reflect the meaning of the stored information for the specific set of purposes to which a user or group will put it, and simultaneously lend meaning to information according to its place in the structure. The active properties in the Placeless system reflect the consequences of those meanings, and the particular ways in which users would like the system to respond. However, the same principles (supporting multiple perspectives, making action and its consequences visible, and making information control an application matter) clearly apply in a range of other application and collaboration contexts, from communication systems to process support tools to interface generators. Lessons from Placeless concerning the relationship between appropriation, practice and meaning have a much broader application than simply information management.

What aspect of the Placeless Documents design allows it to support the creation of new meanings? Perhaps the most critical is that it allows users to organise their work independently of the boundaries that conventional system models present. Rather than offering a fixed set of applications, with rigid boundaries between them, Placeless offers an approach which is fine-grained and compositional, so that

work can be distributed across a number of areas of activity that would traditionally be located within separate applications (such as workflow system, email system, word processor, etc.). The fixed boundaries of conventional application models impose structure and meaning on the activities of users, rather than allowing them to create and communicate their own. By blurring those boundaries, Placeless's component-oriented model allows meaning to emerge from the practices into which the technologies are incorporated. The blurring of boundaries comes from a separation between information and the structures and encodings that allow people to manage it. This separation between information and representation is a familiar idea, which has been employed in the past to solve traditional technical problems such as interoperation and generality; for example, Kelly et al. (1996) employ a similar approach in their collaborative CASE environment. Placeless differs first in applying this approach much more deeply (there *is* no underlying core representation) and in making it an intrinsic part of the user experience. Ongoing work highlights the critical role that decoupled representation can play in coordinating heterogeneous work (Reddy et al., 2001).

What this suggests is that recent uses of component-oriented approaches in CSCW (e.g. Litiu and Prakash, 2000; Roussev et al., 2000) may have a broader application than simply to the internal architecture of collaborative systems. Although component models are seen as a route towards flexibility, responsiveness and maintainability at an infrastructure level, they may also offer a route towards a more flexible user experience and, in turn, to a better match between the capabilities of technical systems and the needs of practice.

8. Ongoing and future work

Our experiences developing the Placeless Documents infrastructure and looking at the experimental deployment of applications on top of it suggests that the basic conceptual model which has been the focus for this paper is largely successful. Properties provide a natural and compelling way to organise information resources, and their natural compositionality supports information sharing and collaborative interaction.

The Placeless Documents infrastructure is now in its third major revision. This allows us to focus more directly on the needs of applications and interaction styles. Current work is looking at the applicability of spatial hypertext models to the design of interfaces for fluid information spaces. Spatial hypertext systems exploit spatial arrangements and visual features of information objects for clustering and relating them, rather than using formal and explicit linkage structures. These models lend themselves naturally to a more fluid style, particularly for sensemaking tasks (Shipman and Marshall, 2000). Our goal is to use this approach to support a wider application deployment that can in turn provide richer understandings of the practical issues surrounding property-based information management and the appropriation of fluid information stores.

9. Conclusions

Studies of customisation in interactive systems have typically concentrated on the way in which explicit features of the system's configuration can be adjusted to suit the different settings in which the system might be used, or the needs or preferences of different people who might use them. However, customisation is a broader phenomenon than this perspective would suggest. In this paper, I have used the term "appropriation" to refer to the ways in which people adopt and adapt interactive technologies, fitting them into working practices and evolving those practices around them. Most of the papers in this special issue present discussion theoretical frameworks for appropriation and show how, in a variety of settings, these frameworks can be used to explain how the process of appropriation leads to transformations in the use of technology. In contrast, this paper explores the technical foundations of appropriation. Clearly, both of these – a theoretical account of the process of appropriation, and a foundational understanding of the consequences of technological design – are necessary if we are to be able to produce software systems that fit more naturally into adaptive patterns of practice.

Our experiences with a range of applications built on top of the Placeless infrastructure point to some common design principles. In this paper, I have discussed three: supporting the interoperation between multiple perspectives or organisations for the same information, making action and the consequences of action visible in an interface, and making control over information a matter for the application rather than the infrastructure. These principles each reflect part of a larger exploration of the potential uses of composable component-based architectural models not only for infrastructure development but also for a radically different collaborative user experience, and resonate with the experiences of appropriation in other settings reported in the research literature.

These explorations of appropriation, which began with the attempt to take a broader view of the role of customisation, have suggested that appropriation is best thought of as the incorporation of technology not simply into practice but into systems of meaning. Appropriation is the creation, management and communication of meaning, within a community of practice. Viewed in this light, then, we can see how the three principles put forward here can have application to a much broader set of tasks and applications than I have discussed here. At the same time, it also shows how much more work remains to be done to elaborate appropriation as a fundamental aspect of system adoption, and as a critical bridge between design and studies of practice in CSCW and HCI.

Acknowledgements

Many people contributed to the development of the Placeless Documents system. The other members of the Placeless Documents group – Keith Edwards, Anthony LaMarca, John Lamping, Karin Petersen, Mike Salisbury, Doug Terry and Jim Thornton – made it all possible. The work on Macadam was conducted in collabo-

ration with John Lamping, Tom Rodden, Lucy Suchman, Randy Trigg and Jeanette Blomberg, while the Bernoulli work arose from a collaboration with Richard Bentley, Rachel Jones and Allan MacLean. Mark Ackerman, Jonathan Grudin and the issue editors and reviewers provided valuable feedback on earlier presentations of these ideas.

Notes

1. Placeless Documents was developed by a team of researchers, including myself, at the Xerox Palo Alto Research Center.
2. Active properties in Placeless are similar to, but more general than, related approaches such as active (dynamic) values or "triggers."
3. These quotations are taken from transcripts of ethnographic interviews conducted by Jeanette Blomberg, Lucy Suchman and Randy Trigg.
4. I am construing "application" broadly here. Typically, workflow is organised as a client/server system; however, it is manifest to a user as an application rather than as an infrastructure service.

References

Barreau, D. and B. Nardi (1995): Finding and Reminding: File Organization from the Desktop. *SIGCHI Bulletin*, vol. 27, no. 3.

Bellotti, V. and I. Smith (2000): Informing the Design of an Information Management System with Iterative Fieldwork. In *Proc. ACM Conf. Designing Interactive Systems DIS 2000*, New York, NY. New York: ACM.

Bentley, R. and P. Dourish (1995): Medium versus Mechanism: Supporting Collaboration through Customisation In *Proc. Fourth European Conf. on Computer-Supported Cooperative Work ECSCW'95*, Stockholm, Sweden. Dordrecht: Kluwer.

Blomberg, J., L. Suchman and R. Trigg (1997): Reflections on a Work-Oriented Design Project. In Bowker, Star, Turner and Gasser (eds.): *Social Science, Tehnical Science and Cooperative Work: Bridging the Great Divide*. Nahwah, NJ: Laurence Erlbaum.

Bowers, J., G. Button and W. Sharrock (1995): Workflow from Within and Without: Technology and Cooperative Work on the Print Industry Shopfloor. In *Proc. Fourth European Conf. Computer-Supported Cooperative Work ECSCW'95*, Stockholm, Sweden. Dordrecht: Kluwer.

Button, G. and W. Sharrock (1997): The Production of Order and the Order of Production. In *Proc. Fifth European Conf. Computer-Supported Cooperative Work ECSCW'95*, Lancaster, UK. Dordrecht: Kluwer.

Dourish, P. (1997): Accounting for System Behaviour: Representation, Reflection and Resourceful Action. In Kyng and Mathiassen (eds.): *Computers and Design in Context*. Cambridge, MA: MIT Press.

Dourish, P., R. Bentley, R. Jones and A. MacLean (1999a): Getting Some Perspective: Using Process Representations to Index Document History. In *Proc. ACM Conf. Supporting Group Work GROUP'99*, Phoenix, AZ. New York: ACM.

Dourish, P., J. Lamping and T. Rodden (1999b). Building Bridges: Customisation and Mutual Intelligibility in Shared Category Management. In *Proc. ACM Conf. Supporting Group Work GROUP'99*, Phoenix, AZ. New York: ACM.

Dourish, P., W. Edwards, A. LaMarca, J. Lamping, K. Petersen, M. Salisbury, D. Terry and J. Thornton (2000): Extending Document Management Systems with User-Specific Active Properties. *ACM Trans. Information Systems*, vol. 18, no. 2, pp. 140–170.

Gerson, E. and L. Star (1986): Analyzing Due Process in the Workplace. *ACM Trans. Office Information Systems TOIS*, vol. 4, no. 3, pp. 257–270.

Giddens, A. (1984): *The Constitution of Society: Outline of the Theory of Structuration*. Cambridge: Polity.

Greenberg, S. (1991): Personalizable Groupware: Accommodating Individual Roles and Group Differences. In *Proc. European Conf. Computer-Supported Cooperative Work ECSCW'91*, Amsterdam, Netherlands. Dordrecht: Kluwer.

Grudin, J. (1988): Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces. In *Proc. ACM Conf. Computer-Supported Cooperative Work CSCW'88*, Portland, OR. New York: ACM.

Grudin, J. and L. Palen (1995): Why Groupware Succeeds: Discretion or Mandate? In *Proc. European Conf. Computer-Supported Cooperative Work ECSCW'95*, Stockholm, Sweden. Dordrecht: Kluwer.

Harrison, S. and P. Dourish (1996): Re-Place-ing Space: The Roles of Space and Place in Collaborative Environments. In *Proc. ACM Conf. Computer-Supported Cooperative Work CSCW'96*, Boston, MA. New York: ACM.

Kaptelinin, V. (1996): Creating Computer-Based Work Environments: An Empirical Study of Macintosh Users. In *Proceedings of the ACM SIGCPR/SIGMIS'96 Conference*, Denver, Colorado. New York: ACM.

Kelly, S., K. Lyytinen M. and Rossi (1996): MetaEdit+: A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment. In Constantinopoulos, Mylopoulos and Yassiliou (eds.): *Lecture Notes in Computer Science 1080: Proc. Eighth Intl. Conf. Advanced Information Systems Engineering CAiSE'96*, Heraklion, Crete. Berlin: Springer, pp. 1–21.

Litiu, R. and A. Prakash (2000): Developing Adaptive Groupware Applications Using a Mobile Component Framework. In *Proc. ACM Conf. Computer-Supported Cooperative Work CSCW 2000*, Philadelphia, PA. New York: ACM.

Mackay, W. (1990): Patterns of Sharing Customisable Software. In *Proc. ACM Conf. Computer-Supported Cooperative Work CSCW'90*, Los Angeles, CA. New York: ACM.

MacLean, A., K. Carter, L. Lovstrand and T. Moran (1990): User-Tailorable Systems: Pressing the Issues with Buttons. In *Proc. ACM Conf. Human Factors in Computing Systems CHI'90*, Seattle, WA. New York: ACM.

MacLean, A. and P. Marqvardsen (1998): Crossing the Border: Document Coordination and the Integration of Processes in a Distributed Organisation. In Wakayama et al. (eds.): *Information and Process Integration in Enterprises: Rethinking Documents*. Boston: Kluwer, pp. 109–124.

Orlikowski, W. (1992): Learning from Notes: Organizational Issues in Groupware Implementation. In *Proc. ACM Conf. Computer-Supported Cooperative Work CSCW'92*, Toronto, Ontario. New York: ACM.

Orlikowski, W. (1995): *Evolving with Notes: Organizational Change around Groupware Technology*. Working Paper 186, Center for Coordination Science. Cambridge, MA: MIT.

Palen, L. (1995): Social, Individual and Technological Issues for Groupware Calendar Systems. In *Proc. ACM Conf. Human Factors in Computing Systems CHI'99*, Pittsburgh, PA. New York: ACM.

Poole, M. and G. De Sanctis (1990): Understanding the Use of Group Decision Support Systems: The Theory of Adaptive Structuration. In Faulk and Steinfield (eds.): *Organizations and Communication Technology*. Newbury Park: Sage, pp. 173–193.

Reddy, M., P. Dourish and W. Pratt (2001): Coordinating Heterogeneous Work: Information and Representation in Medical Care. In *Proc. European Conf. Computer-Supported Cooperative Work ECSCW 2001*, Bonn, Germany. Dordrecht: Kluwer.

Roussev, V., P. Dewan and V. Jain (2000): Composable Collaboration Infrastructures Based on Programming Patterns. In *Proc. ACM Conf. Computer-Supported Cooperative Work CSCW 2000*, Philadelphia, PA. New York: ACM.

Shipman, F. and C. Marshall (2000): Formality Considered Harmful: Experiences, Emerging Themes, and Directions on the Use of Formal Representations in Interactive Systems. *Computer Supported Cooperative Work*, vol. 8, no. 4, pp. 333–352.

Suchman, L. (1987): *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge: Cambridge University Press.

Trigg, R., J. Blomberg and L. Suchman (1999): Moving Document Collections Online: The Evolution of a Shared Repository. In *Proc. European Conf. Computer-Supported Cooperative Work ECSCW'99*, Copenhagen, Denmark. Dordrecht: Kluwer.

Trigg, R. and S. Bødker (1994): From Implementation to Design: Tailoring and the Emergence of Systematization in CSCW. In *Proc. ACM Conf. Computer-Supported Cooperative Work CSCW'94*, Chapel Hill, NC. New York: ACM.

Wenger, E. (1998): *Communities of Practice*. Cambridge: Cambridge University Press.