# Building Bridges: Customisation and Mutual Intelligibility in Shared Category Management

**Paul Dourish*, John Lamping*, and Tom Rodden†**

*Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto
CA 94304
USA
*{dourish,lamping}@parc.xerox.com*

†Computing Department
Lancaster University
Bailrigg
Lancs LA1 4YR
United Kingdom
*tom@comp.lancs.ac.uk*

## ABSTRACT

Research into collaborative document use often concentrates on how people share document *content*. However, studies of real-world document practices reveal that the structures by which document corpora are *organised* may also, themselves, be important sites of collaborative activity. Unfortunately, this poses a problem. When category structures are used to understand a set of documents, the manipulation of those structures can interfere with shared understanding and intelligibility of the document space.

We show how this problem arises in real-world settings, using a case arising from some recent field work. We outline a solution to the customisation/intelligibility problem, and show how it has been implemented in a system for personal and workgroup document management.

**Keywords**: document management, categorisation, customisation, shared views, shared workspaces.

## INTRODUCTION

Investigations into the end-user customisation of software systems have repeatedly emphasised the fact that customisation is a collaborative phenomenon. These studies have revealed that, rather than being a solitary or individual activity, customisation happens within a complex web of artifacts, people, organisations and practices. Mackay [13, 14] studied patterns of customisation of UNIX software and observed the critical role played by particular individuals, whom she called “translators”, who would act as focal points for customisation activity and for the distribution and reproduction of customisations. Trigg and Bødker [18] encountered a similar phenomenon, and observed that this role was sufficiently important as to be organisationally sanctioned and supported in the government agency that they studied. MacLean et al. [15] discussed the development and deployment of a customisation technology and concluded that the fostering of what they called a “tailoring culture” was, in many ways, as important as the development of the technology itself in creating an environment in which tailoring was not only possible, but acceptable.

However, it is possible to draw a deeper connection between customisation and collaboration. Bentley and Dourish [2] have argued that customisation can be seen not simply as a collaborative activity in itself, but rather as a fundamental, even constitutive, facet of collaboration. From this perspective, it is not simply that customisation occurs within a community of practice, but rather than the very nature of collaborative activity involves the continual adaptation, appropriation and reconfiguration of technologies, artifacts and environment, performed by and within an ongoing collaborative arrangement. Customisation is a natural consequence of the situated, improvised achievement of work that is the basis for much theoretical and practical work within CSCW [20]. Where the situated perspective emphasises how the configuration of the environment and circumstances of action shape the action that arises, customisation reflects the way in which individuals and groups will configure that environment to suit their immediate needs and evolving work practice.

So, the problems of software systems customisation are not simply restricted to questions of how flexible software systems should be, and what knobs and levers they might provide for people to adjust and adapt the interface. Instead, those problems are both fundamental and endemic to the nature of computer-supported cooperative work. Customisation is part of the fabric of collaborative activity.

Customisation does not come without problems, though. In particular, local customisation can interfere with the smooth management of collaborative work. Even in early experiments such as Colab, researchers noticed that local customisations interfered with people's ability to make shared references [17]. In other words, there is a tension between *customisation* and *mutual intelligibility*. When the system is adapted to the needs of one particular user (or group), it becomes less useful or comprehensible to the others with whom the user (or group) might be working.

The work which we will discuss in this paper is focussed on one example of this sort of concern. It arises from an ongoing engagement with a group of workers at a state government organisation. Following from a long-term interest in working document collections [3], the Work Practice and Technology (WPT) group at PARC has been studying the document practices of this organisation with an eye to understanding the potential impacts and opportunities presented by the introduction of digital document technology into workplaces such as theirs. They have deployed a prototype exploring issues of document coding, filing, browsing and retrieval [19]. In contrast, our concern here is with how, first, the task of document categorisation is subject to local customisation and adaptation according to the immediate needs of the project group, and then how, second, the outcomes of these local adaptations can be reflected back to others in a way that makes sense to those who may not be party to the adaptations and customisations that have taken place.

The rest of this paper is organised as follows. We will begin by discussing how the problems of customisation and mutual intelligibility are manifest in the ethnographic material we have been working with. We will use the features of this setting both to contextualise the problem, and to generate a set of specific issues to be addressed in systems for managing workgroup documents. We will then introduce a technique we have been exploring to provide a solution to this problem. We will show how it addresses the needs of the work setting, and how it has been embodied in a prototype system based on a novel approach to personal and workgroup document management. Finally, we will consider some of the wider implications for collaborative document system design.

## STUDYING DOCUMENT PRACTICES

The field work that informs our work was conducted at a state organisation, here called "The Department." The field work, and one particular technological intervention that has resulted from it, has been described elsewhere [19, 21], and so we refer readers to those accounts for a detailed exploration document practices at the site. What we provide here is a brief sketch of the main concerns, and the main lessons we can learn from this investigation for our concerns at hand.

### The Project Files and the Uniform File System

The Department is a state government organisation responsible for (amongst other things) the planning, execution and management of large engineering projects (in the case of our study, building a bridge). Projects may take many years, in the course of which they will move through a number of different phases (such as environmental impact assessment, design, construction, etc.). These phases may be carried out by different groups of people.

Coordination between these different groups, and the different individuals in a group, is achieved in part through the maintenance of a large document collection called the *project files*. The project files (or a subset of them) move from group to group in the course of the project, and will be later archived for legal and reference purposes. The project files comprise a large and heterogeneous collection of documents including letters, plans, memos, reports, minutes of meetings, etc. They are currently maintained entirely on paper; PARC's engagement with The Department has been related to their interest in exploring on-line alternatives and their consequences.

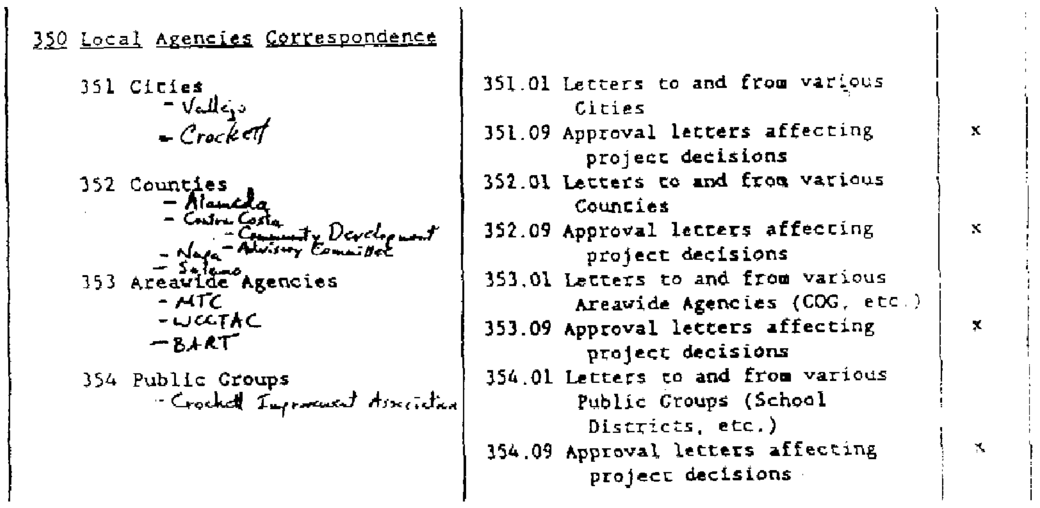
The project files are coded according to an organisational filing scheme called the Uniform File System or UFS, a three-level hierarchical classification system. Each document is assigned a numeric UFS code, which then determines that document's position in the collection of three-ring binders that make up the project files. When more than one UFS category applies to a given document, the organisation stipulates that it should be filed according to the category for the document's source; in practice, engineers often use their best judgement in choosing a category.

### The UFS In Flux

The UFS, of course, suffers from the same sorts of problems that affect all categorisation schemes. As Gerson and Star observe, "No representation of the world is either complete or permanent." [9] So, in this case, the UFS is never sufficiently elaborated for all the specific needs to which each project might put it. For example, the UFS cannot name all the organisations that might possibly send letters to The Department, especially since these organisations may well be grass-roots movements that emerge specifically in response to The Department's activities. Similarly, the issue of how and where a document should be categorised is often a question of how the coder anticipates the document might be used; it is not an absolute evaluation.

As a consequence, one feature of the ethnographic material is the emergence of both problems and solutions arising from the question of "how to apply the UFS to a project." Perhaps the most obvious sign of these issues is artifacts like that pictured in figure 1: a copy of the UFS that has been annotated to show modifications locally appropriate to a specific project. Despite being organisationally mandated, then, the use of the UFS is not stable. The flux surrounding the UFS is part of the problem to which we must address ourselves in considering a move on-line. We can identify three ways in which the UFS is subject to change:

1. *Periodic organisational changes*. The UFS must be updated periodically to account for changes in organisational structure and new patterns of relationships between The Department and the other organisational


350 Local Agencies Correspondence

351 Cities
- Vallejo
- Crockett

352 Counties
- Alameda
- Contra Costa
- Community Development Advisory Committee
- Napa
- Solano

353 Areawide Agencies
- MTC
- WCCTAC
- BART

354 Public Groups
- Crockett Improvement Association

| | | |
|---|---|---|
| 351.01 | Letters to and from various Cities | |
| 351.09 | Approval letters affecting project decisions | x |
| 352.01 | Letters to and from various Counties | |
| 352.09 | Approval letters affecting project decisions | x |
| 353.01 | Letters to and from various Areawide Agencies (COG, etc.) | |
| 353.09 | Approval letters affecting project decisions | x |
| 354.01 | Letters to and from various Public Groups (School Districts, etc.) | |
| 354.09 | Approval letters affecting project decisions | x |

FIGURE 1: A working copy of the UFS, showing local modifications and amendments.

entities with which it interacts. Every so often (on the order of years), then, The Department issues a new version of the UFS to reflect these changes. There is no established organisational procedure for handling these changes in current project files. To re-file according to a new UFS would be prohibitively expensive and time-consuming. Variations in the UFS are handled as a practical element of filing and retrieval by project members.

2. *Local group customisations*. The UFS cannot accommodate the wide variety of needs and purposes to which it might be put for every different project. Each group must, for its own purposes, make local customisations to the UFS: introducing new categories, conflating other ones, and so forth. It should be noted that these changes need not be reflected in an explicit revised form of the category structure, but may be commonly understood practices of document coding and retrieval.
3. *Individual patterns of use*. Since there is some latitude in choosing categories, particularly when multiple categories seem appropriate, there are individual variations in filing practice. Individuals may adopt particular conventions concerning the use of the UFS for document filing.

These are general problems. Garfinkel [8] explored the inherent incompleteness of coding schemes and how their use required the practical working out of the purposes to which the categories were to be put; Bowker and Star studied issues of ambiguity in the evolution of a "standardised" medical classification [4]; and Dourish et al. discussed how shared data is interpreted according to its source and context [6]. Our goal here is not simply to identify these issues, but rather to understand their consequences for the design of collaborative document management systems.

## PROBLEMS OF CUSTOMISATION AND INTELLIGIBILITY

Customisation of the UFS allows members of the project team to solve one set of problems, those concerned with how to use the UFS in their own document filing work. Their customisations bridge the gulf between the UFS and their immediate needs. However, the project files are a shared repository. So, the problem of the tension between customisation and mutual intelligibility arises as part and parcel of this adaptation of the UFS to local contingencies.

This problem emerges in two ways. First, there is the problem that individual approaches to filing may make it difficult for others to retrieve documents if they approach the project files from a different perspective. Discussing the case of a troublesome document, the Senior Project Engineer says:

> *"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."*

Second, this problem emerges on a larger scale. Groups, as well as individuals, can have incompatible views of the document corpus. The project files serve not only for local coordination, but also persist across the different groups involved in the lifetime of the project, as well as serving an archival role. Each project has unique needs and adaptations that they introduce. However, the different elaborations made by one group in, say, the environmental assessment phase may be meaningless to those who are concerned with design or construction, or those who might come to look up documents years later.

### Customisation and Collaboration

We began this paper with reference to research on customisation and tailorability that has been conducted within the HCI community. It has long been recognised that the opportunity to adapt the behaviour of an interactive system to local needs is critical in deploying systems and making them fit into new environments, whether that adaptation is the configuration of a mail system, the provision for special-purpose commands for specific environments, or simply being able to move the mouse from one side of the computer to the other.

It is no surprise, then, that those developing collaborative interactive technologies should also investigate the role that customisation can play.

However, a fundamental problem arises as soon as this shift is made. The classical view of customisation in the HCI literature considers the relationship between a single user and one or more software systems. However, in a collaborative setting, there are many users, and so customisation in the collaborative context introduces new questions. If one user adjusts the system to meet their own needs, how does that affect the other users of the system? Must they also adopt the same adaptations? Can they have their own? What if they select incompatible customisations? How should these be resolved?

So, the provision of customisation in the interfaces of collaborative systems has introduced a tension between the needs of the group and the needs of the individuals, and between the needs of different individuals at the same time [10]. In general, this problem has emerged as a feature of the interface design, and as such, it has been regarded as a form of the "WYSIWIS problem" and the question of degree of coupling to be allowed between interfaces. As such, approaches allowing flexible or variable coupling between the interfaces of different users have been a standard approach to the problem [5]. However, the problems of collaboration and customisation manifest themselves in quite different ways in the case of the project files, requiring a different solution.

The source of this difference is that, in this case, the *object* of customisation is quite different. In most scenarios, what is being customised is the interface to the objects of work, or the procedures that are enacted over those objects. However, in the work at The Department, what is being customised is the very "work object" itself. The object of document categorisation work is not just the documents being categorised, but the category schema that is used to classify the documents. The category schema is a shared resource for the members of the group, which in turn informs, enables and constrains how they work with the documents.

Mackay [13] uses the term "co-adaptive" to describe the phenomenon of software customisation:

> *"'Co-adaptive' emphasizes the dual nature of the phenomenon: that people both adapt to technology (reacting to it) and adapt it to meet their needs (proactively changing it). Both processes occur over time and influence each other." (p13)*

So, the practices surrounding the use of the project files and the local customisations of the UFS cannot be seen as independent, but, rather, are deeply intertwined. Changes to the UFS are introduced to accommodate the needs of practice, and themselves change in the way the group will work. These patterns of use unfold and adapt over time.

Customising the UFS introduces changes not only for the local group but also for other groups who may have to use the project files both immediately and in the future. In order to look up documents, or understand the structure of the document corpus, users need to be able to understand the category structure in use and the practices surrounding it. The problem, then, is one of mutual intelligibility. If the "language" of the category scheme is subject to local modifications or revisions, how can communication take place? More specifically, if someone categorises documents according to a locally accepted version of the UFS, then how are those documents to be interpreted by someone in the next group, or in a later phase of the project, or further up the organisation?

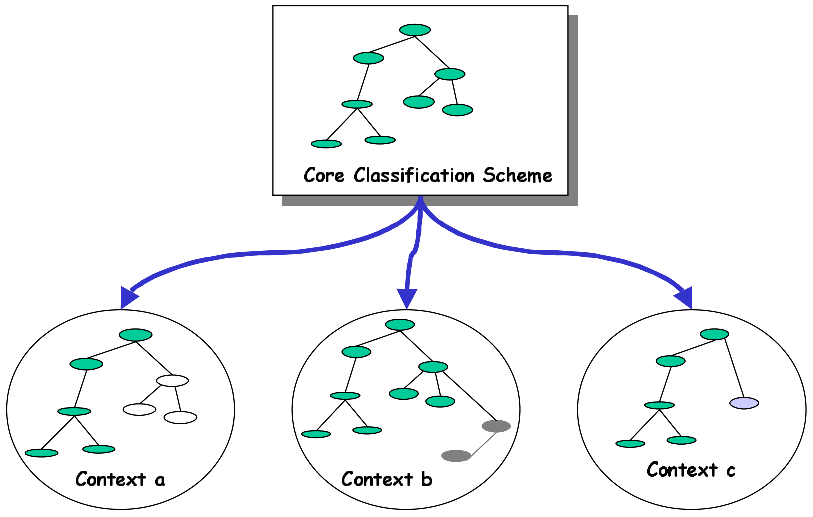
## DEVELOPING A SOLUTION

The problems of intelligibility arise at The Department because the object of customisation, the filing scheme, is also the fundamental mechanism supporting group collaboration. So, in considering the implications of moving from paper to electronic storage of the project files, we need to be able to take into account the interactions between the use and manipulation of the UFS as part of The Departments's work practice.

### Requirements

WPT's engagement with The Department has included the development and deployment of a prototype system for online filing and retrieval of project files [19]. Their experiences in developing and maintaining this prototype has greatly enriched our collective understandings of the use of the UFS in the work of The Department's engineers. On the basis of the document practices observed at The Department, then, we can formulate a set of requirements for a document management system supporting this sort of use.

1. It must allow documents to be categorised according to a hierarchical scheme, such as the UFS. That hierarchy must be accessible and usable not only for document coding, but also for document search and retrieval. This means that, for example, a document filed under "Environmental Protection Agency" must also be retrievable under the more general search term "Federal Agencies".
2. It should support multiple hierarchies of this sort, corresponding to the different sorts of data by which documents might be coded (e.g. sources, budget codes).
3. It must allow customisations to hierarchies. The customisation mechanism must allow cumulative customisation by, for example, the organisation, project, work group and individual.
4. It must make explicit the context under which actions take place. That context determines the currently operative set of customisations.
5. It must be able to "translate" between different levels of customisation. In particular, it must be able to interpret documents according to both the context under which they were filed and the context under which they are retrieved. That is, if a document is filed under Joe's personal customisations, on top of the Bridge project's local



version of the UFS, the system must present the document in a way that makes sense when it is viewed by others with other perspectives.

On the basis of these requirements, we have developed an approach to category representation that addresses the tension between customisation and intelligibility.

## Conceptual Model

The easiest way to imagine our data model is through the metaphor of a stack of translucent sheets.[1] Each sheet holds part of the category structure, and as they are layered on top of one another, they each contribute a part of the whole. As a new sheet is added, it can modify the sheet below, renaming categories, adding new ones or removing those already present. Other components that lie behind it are unaffected; they show through. Different levels correspond to different ways of seeing the data; by layering them on top of each other, we support the gradual accumulation of local customisations by the organisation, project, workgroup individual, etc.

Any group of one or more sheets constitutes a "context", a set of operative features of the working situation, within which a person is working at any moment. For example, a context for someone working on a given project might bring the basic UFS structure together with the relevant customisations for the organisation, the project, and the individual, and combine them into a unified whole. So, at any particular time, there are a variety of contexts at work. Actions take place in a particular context, and each context element (or layer) makes some contribution to an interpretation of the activity that's taking place.

1. "Translucent sheets" are simply a metaphor for how compositions compose. Unlike systems such as CaveDraw [12], we do not provide a layered user interface.

The "translucent layers" model applies not only to the category structure itself, but also to documents that have been categorised according to the model. Documents are coded by someone working in a particular context (with a particular set of layered sheets). When someone else looks at the document, through a different set of sheets, the document's filing structure can be transformed so that it makes sense according to the viewer's context. In other words, the system allows one viewer to look at the documents organised from the perspective of a set of derived structures (see figure 2).

Contexts, then, are the basic element for customisation. When users change the category structure, they do it within the current context. This means that the changes they introduce affect only other people operating in the same context, or a derived context, such as other people working on the same project. As people move between contexts, the relevant sets of customisations are brought into play. Note that the layering approach means that contexts record *changes* to category structures, rather than category structures themselves. This is a key feature of our design, and holds important consequences.

The first is that they can be layered and combined. The ability to layer contexts allows us to support the process of *cumulative customisation*; that is, that there may be organisational customisations, group customisations and private customisations all operating and once, with different scopes. Cumulative customisation is a means to support the situation we find at The Department in which different customisations may have different scopes, applying to projects, or individuals, or after certain points in time, and so on.

A second consequence of the layered approach is *robustness*. Since we have described local customisations as a series of differences from the more global structure, we can re-apply those differences whenever the global structure changes, to yield a new local view. If we had customised the structure by making changes to a private copy, then it would be harder to make those changes stable over time, and changes at the organisational level would eliminate customisations to support local workgroup practice.

The third consequence of our approach is that the changes on each layer can be interpreted *bidirectionally*. Not only can one context be added to another context, to add new levels of customisation, but it can also be subtracted again, to remove them. By combining this feature with the a hierarchy's natural ability to express partial information, we are able to tackle the problem of mutual intelligibility.

### Providing for Mutual Intelligibility

Imagine two users, Adam and Brenda, operating in two contexts, A and B, both derived from a core context, C. Adam has removed some categories from context A, while Brenda has renamed some categories in context B. They have each categorised documents according to their own contexts. What happens when they look at each other's documents?

Because the contexts are recorded as sequences of reversible changes, we can *reinterpret* their categorisations according to each other's contexts, For instance, we can provide a view of Adam's activities in a way that makes sense for Brenda by, first, *reversing* context A and then *adding* context B to the categorisation of the document when we present it to Brenda for viewing. So, if Adam has categorised a document using a unique category that he added in context A, then when we subtract A, that document will be categorised according to the more general category in the core context; and if Brenda has renamed that category in her context B, then we will see the document with the renamed category when we add context B in again.

Recording contexts as changes to category structures, rather than as category structures themselves, supports this by letting us add and subtract categories. Using abstract categories to represent partial information is also required, so that we can substitute a more general category for a specific one that does not appear in the "destination" context. So, by decontextualising and recontextualising documents filed according to customised category structures, the document space can be made intelligible across different contexts.

Since many contexts can be combined, this approach supports a range of situations encountered at The Department, including sharing documents between members of the same project group[2], across different project groups, and at different points in time (e.g. after documents have been archived).

## THE DESIGN OF THE MACADAM PROTOTYPE

As part of their work with The Department, WPT has developed a web-based prototype for managing on-line project files, exploiting richer document codings and image-based search techniques [19]. However, it does not tackle the issues of customisation in the UFS. As an embodiment of the conceptual model described above, we have developed a second prototype, called Macadam, to explore these questions in the context of a possible on-line document repository. The goal of our prototype is, first, to explore the effectiveness of the technique when embodied in an interactive tool and, second, to act as a focus for future design explorations.

### The Presto Infrastructure

As a starting point for development, we decided to base our development on Presto, an experimental document management infrastructure developed in the Computer Science Lab at Xerox PARC [7]. Presto is intended to manage personal or workgroup documents numbering in the thousands. It offers a uniform basis for organising, storing, retrieving and manipulating documents through their *properties*, such as the property of being 14245 bytes long, the property of being a Frame file, the property of being owned by Paul, the property of being related to the Macadam project, the property of being a paper, the property of being a draft, and so forth.

Presto has been developed as a part of a larger project called "Placeless Documents", which is investigating the role of document properties as a uniform means for managing and interacting with document collections. It is designed as an extensible infrastructure upon which applications can be developed and deployed. Its core features, such as flexible properties, persistent storage and an extensible user interface, made it a good match for the needs of our prototype.

However, Presto also lacks a number of the features that we require in our eventual design. Presto properties are completely untyped, and so Presto has no facility for organising document property values according to a hierarchy. Similarly, Presto does not support multiple perspectives on documents, and so it offers no support for moving between different layers of customisation.

### Representing Objects as Abstract Documents

Much of Macadam's design is based on *abstract documents*. Abstract documents are documents in Presto that have no content, but can still hold properties and maintain their identity. We can use these to represent structured data.

Internally Macadam represents each category value as a separate Java object, with the details of the context to which they belong and the relationships between different values represented as object attributes. Instantiating new Java objects creates new category values and existing values can by dynamically manipulated by altering the object's attributes. Each of the Java objects map onto abstract Presto documents with the object attributes mapped onto corresponding Presto document properties. This mapping is maintained by the Macadam prototype with changes in the object structure immediately reflected onto changes in the underlying Presto store.

2. This sharing is largely asynchronous; we have not addressed synchronous access here. Our model does not exclude synchronous access, but a very different user interface would be required.

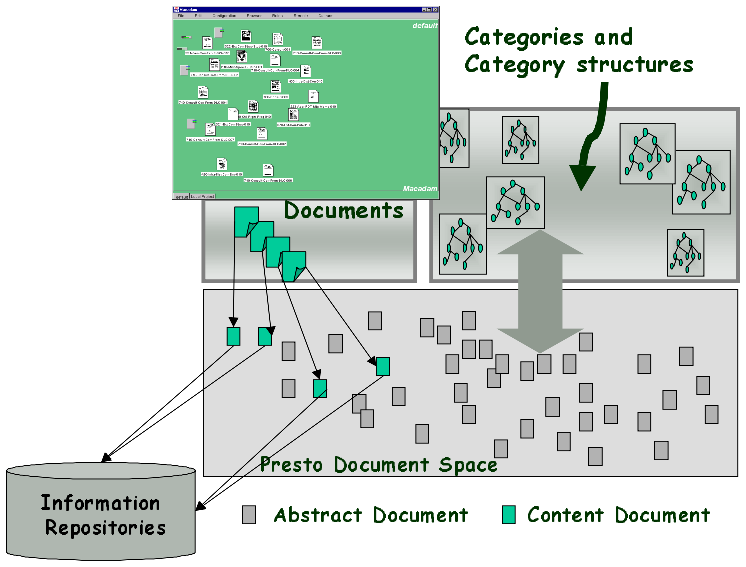


FIGURE 3: The Presto document space stores not only the basic documents in the Project Files, but also the abstract documents representing categories and their structures.

One immediate advantage of this approach is that the application structures created are both external and persistent. The effects of user changes are reflected outside the application space and can be drawn upon by other users. However, equally significantly, a considerable amount of *auditing information* is also recorded about changes and alterations to the Presto document. This auditing information allows us to extend our sharing of the categories values by also representing some details of the activities involved in the construction and alteration of the categories in the information space. Essentially we can consider each abstract document as having two sets of properties, *Macadam properties* that map up to the attributes used to represent the structured value space and *raw Presto properties* that hold auditing information.

### Attaching Categories to Documents

Although the underlying Presto document properties are entirely untyped, the demands of this setting require that we introduce a more sophisticated method for managing document categories in Macadam. Driven by these concerns, we introduce some levels of indirection in order to encode category information.

The most fundamental indirect object is the *property instance*. A property instance is an object that explicitly represents the fact of a document having a particular property. The assignment of a Macadam property is represented as a property instance object. This object is, in turn, represented as an abstract Presto document, and so supports auditing information. We can also record what sort of value this property holds; in other words, the *category type* information.

### Kinds and Categories

The category type information in property descriptions is organised with representations we call kinds and categories. Categories are the terms by which a document can be coded, by being assigned as the value of a property; so, when the property "source = EPA" is assigned to the document, the value EPA is a category. Presto can already perform property associations, of course, but it does not do enough for our purposes, because we need to be able to say things like "the property 'source' should always refer to an organisation'. We need a way to turn categories into properties, which requires a way of understanding how categories can be grouped together according to the type of category they are.

We refer to category types as *kinds*. For instance, "organisation" is a kind. A kind names a set of possible category values. So, possible values such as "Federal Highways Agency", "Environmental Protection Agency", and "Crockett Improvement Association" would all be categories that occur in the "Organisation" kind. Macadam has multiple different kinds, for different sorts of categories (such as organisations, projects, budgets, etc.). Within a kind, categories are organised in a hierarchy. This means that some categories can be more specific than others. For instance, "Federal organisation" is a valid category that can appear in the "Organisation" kind; it would appear above "Environmental Protection Agency", for instance, but not above "The Department", since The Department is a state, not federal, organisation.

Categories that appear in the middle of a category tree, rather than at the bottom, are called *abstract* categories. Abstract categories have four uses in Macadam:

1. Their first use is in searches. When an abstract category is used as a search term, all categories subsumed by the abstract category are included in the search. A search for "source = federal organisation" will return documents that came from any federal organisation.

2. The second, rarer, use is in filing. When there is confusion over the "correct" value for a document property, an abstract category can be used rather than leaving the document uncoded. Confusion might arise because the document itself is unclear, or because the correct answer might involve two different values, and so forth.
3. The third use is in viewing documents. When two users have different views of the category structure, because they each have different local modifications, then an abstract category can be used to give them a consistent view. Modifications to the category structure usually involve refining that structure by adding information to the leaves of the category tree; abstract categories are still shared, and present a unified view of the structure.
4. Finally, the fourth use of abstract categories is to support reinterpretation of information, allowing one user's view of a category structure to be described in terms of another's. We will discuss this in more detail shortly.

When we put the components together, the result is the architecture illustrated in figure 3. All application and user data is persistently maintained and made available for browsing or searching by the Presto data space. Internally, Macadam manages property instance, property descriptor, kind and category objects, and uses these to present the user with a view of a typed, structured space of document properties.

### Contexts

Macadam uses the notion of context, as explored in the earlier section on the conceptual model, to mediate between the different set of local customisations that might be introduced into the categorisation structure. Although Presto offers no native control over the set of possible values that a property can hold, the preceding description outlined how Macadam introduces a series of indirections to create a structured representation of the value space that allows hierarchies of possible values to be created and exploited. One further level of indirection allows us to introduce contexts.

We introduce this level of indirection into the encoding of the category structure rather than the property values themselves. In the representation of a kind, the hierarchical links between categories must be annotated with information that describes the range of contexts in which they are valid. So, when a user introduces a new category into a kind, the link between that new category and its parent category is valid only in the immediate context.

A controller widget allows users to select which context they are working in at any moment. As described above, contexts are layered, so that a personal context can be overlaid on a workgroup context, and so forth. In this way, a series of local customisations are combined into a single view. At any point where the user might want to assign a property value, the opportunity is presented to customise the set of possible values. Switching from one context to another changes the set of category structures available for assignment, as well as providing the means to adjust the view of documents and their encodings according to the currently operative context, addressing the problem of mutual intelligibility.

### Templates

At the same time as the "translucent sheet" model supports local customisation and mutual intelligibility, Macadam offers a complementary mechanism for customising the filing mechanism itself, and for reifying and sharing these customisations. As previous studies of customisation such as those of Maclean et al. [15] or Mackay [21] have shown, simply providing the means to create adaptations over time is not in itself sufficient to support customisation as a common practice. In particular, we need to be able to fit customisation, as a technical phenomenon, into a wider pattern of the mutual evolution of tools and practices. So, Macadam needs to provide a convenient mechanism for collecting together and sharing customisations. We provide this through templates.

A template is, conceptually, an empty document with a set of property assignments. In Macadam, a template can be dropped onto a document, automatically assigning all its properties to the document. Properties assigned through a template are just like any other properties; they can be searched over or changed in the normal way. Users can have any number of template documents on the desktop in the multiple workspaces provided by the user interface.

Templates provide users with a way to conveniently assign a set of related properties to a document in a single step, rather than having to go through the potentially laborious process of assigning each property by hand. More importantly, they provide a means to codify property sets. Templates can be created that correspond to common document forms. A critical aspect of this use is that templates are not simply a "macro" facility in the user interface, but actual, concrete document objects in Macadam, making them both *shareable* and *persistent*. Templates, then, can be passed from one user to another, accumulating changes and customisations as they go. Like MacLean et al.'s Buttons [15] or Mackay's configuration files [21], these templates are both a focus of customisation activity and a currency of exchange.

## RELATED APPROACHES

We have already alluded to research into the trade-offs arising in customisation and collaboration; however, other work is more directly connected with our concerns here.

The most closely related work is that of Simone et al. [16], who discuss the use of an explicit mechanism for mediating between database schema to provide for interoperability and mutual intelligibility between different representations of the same underlying data. In many ways, their solution is similar to ours. However, we use different underlying data models, with different results. The model that Presto offers is essentially one of direct interaction of document objects, rather than the abstract manipulations of document schemas that would characterise work with traditional databases, so we provide a data-centered solution rather than a schema-centered one. At the same time, this also allows us to more easily accommodate multiple different perspectives operating concurrently; this would be more complicated using the pairwise translation mechanism that Simone and her colleagues have developed, at least in its current form.

We have already alluded to aspects of the mutual intelligibility problem considered from the WYSIWIS perspective. Recent explorations in workspace awareness have explored the use of abstract models to mediate between actions and the presentation of their effects [11]. Although this work is directed towards workspace awareness rather than data transformations, it is motivated by a similar set of concerns.

## DISCUSSION

Informed by the field work and experiences with the WPT prototype, the design of Macadam has been organised around three basic issues. The first is the incorporation of meta-information as an element of the shared workspace, on the same level as traditional data objects; the second is the recognition of customisation and adaptation as *intrinsic* features of everyday practice, rather than independent activity; and the third is the importance of mutual intelligibility as a fundamental concern in collaborative information management. We have relied on two mechanisms to support these features, universal representation and incremental customisation. Universal representation allows us to modify data and metadata seamlessly, while incremental customisation allows us to layer, combine and mediate between different perspectives on the information space.

### Uniform Representation

One of the principal design features we have exploited in building this application is uniform representation. Documents, document attributes, and the category structures by which those document attributes are organised are all represented in the same space of abstract Presto documents. This architectural feature also lends itself to particular styles of interface design. In particular, it has the consequence that activities over documents and activities over the category structure can both be made available simultaneously within the same, seamless workspace.

This reflects our understanding of the working domain, and the interaction between using and reflecting upon the category structure. Macadam is designed around the principle that using the structure (e.g. when categorising a document) and reflecting upon it (e.g. when amending it to reflect local needs) are not separable activities, but rather are both aspects of the everyday management of project files. Macadam allows interactions with documents, with properties and with category structures to be mixed fluidly and seamlessly, and the use of Presto as a unifying representational layer supports this. This aspect of Macadam's design is based on a relationship between a particular technical feature of Presto (the generic nature of documents) and a specific observation about the work at The Department (that both documents and document filing structures are shared entities).

These two features work together well. One argues that both data and metadata should be stored in the same place, while the other says that the Presto document space can support exactly this duality. The result is a workspace in which *using* and *reconfiguring* the system are fluidly combined.

### Incremental Customisation

A key feature of Macadam's design is its support for incremental customisation. This reflects the observation that customisation is a collective activity carried out by work groups as a whole. Since customisation affects not only individual but also collective work practices, we need to design collaborative systems to reflect the collective as well as individual perspectives. Macadam's incremental customisation approach reflects an understanding that individual work practices unfold simultaneously in the context of the actions of many others. It lets us draw together a variety of relevant perspectives for both filing and retrieving documents.

### Towards A Common Information Space

We see these features as steps towards a "Common Information Space" [1]. The formulation of the notion of Common Information Space is intended to move beyond the traditional (and technically-defined) "shared workspace" common to CSCW research, and to emphasise a wider set of concerns, such as the:

> *"dialectical nature of these spaces, the frequent need for additional effort in order to put, or use, information 'in common', the need for both closure and openness in representations, [and] their simultaneous portability and immutability, etc." [1:82]*

In other words, the Common Information Space is constituted not only by the information it contains, but by the practices of the generation, use and interpretation of that information. We can see these factors at work in the document management practices at The Department, where the category structure by which the documents are organised is not simply followed, but is *made* to work, practically, against a backdrop of organisational concerns and expectations. Our goal is to explore tools to support "document work"; but we must take this to mean more than simply searching, editing and indexing, but the also the development of shared practices and representations through which the work progresses; in turn, this implies a move from the "mutual visibility" of work, as supported by traditional CSCW awareness techniques, to a focus on the mutual *intelligibility* of work, to which customisation is typically an obstacle.

## CONCLUSIONS

Customisation in collaborative systems is often taken to be an activity in and of itself, separating the *doing* of work (the use of tools) from *reflection upon* work (their customisation). As such, studies of customisation in CSCW have typically followed those in HCI in considering primarily how groups might adjust their tools to more closely suit their needs.

Our work is driven by a different perspective on customisation. We see it as intimately bound up with the activity itself, distinguishable but inseparable. As such, then, we see the need for collaborative tools to incorporate equal control over "data" and "metadata", providing for the mutual management of collaborative activity within the same frame as its concerted performance.

We have described Macadam, a prototype tool for document management that we have developed, drawing on materials from an ethnographic investigation of the document management practices of a group involved in a large engineering project. Macadam provides not only for a variety of local customisations to the structure by which the work is organised, but also for the mutual intelligibility of work across the barriers that these customisations would otherwise impose.

We have been reporting, here, on building three bridges. The first is a bridge between different users' and groups' perspectives on the project file documents. Our prototype introduces mechanisms that allow users to share information despite incommensurate customisations to their organisational schemes. The second is a bridge between field work and technological design. Our design has been developed from reflections on the document practices of a group of engineers and managers, which ultimately serve as both grounding and testing ground for our ideas. Looming over everything else is the third bridge, a real one that The Department is building. We hope that our bridges can last as long as theirs.

## ACKNOWLEDGMENTS

We are indebted to Jeanette Blomberg and Randy Trigg, who advised, guided, encouraged and corrected us throughout the process. Lucy Suchman, Cathy Marshall and David Levy also contributed to the analysis, while Keith Edwards, Anthony LaMarca, Mike Salisbury, Doug Terry, Jim Thornton and Karin Petersen have contributed to the technical development. We are especially grateful to the engineers of The Department. This work was conducted while Tom Rodden was on sabbatical at Xerox PARC.

## REFERENCES

1. Bannon, L. and Bødker, S. (1997). Constructing Common Information Spaces. *Proc. European Conf. Computer-Supported Cooperative Work ECSCW'97* (Lancaster, UK). Dordrecht: Kluwer.
2. Bentley, R. and Dourish, P. (1995). Medium versus Mechanism: Supporting Collaboration through Customisation. *Proc. European Conf. Computer-Supported Cooperative Work ECSCW'95* (Stockholm, Sweden). Dordrecht: Kluwer.
3. Blomberg, J., Suchman, L., and Trigg, R. (1997). Reflections on a Work-Oriented Design Project. In Bowker, Star, Turner and Gasser (eds), *Social Science, Technical Systems and Cooperative Work: Beyond the Great Divide*, 189-215. Mahwah, New Jersey: Laurence Erlbaum.
4. Bowker, G. and Star, S. (1994). Knowledge and Infrastructure in International Information Management: Problems of classification and coding. In Bud (ed), *Information Acumen: The understanding and use of knowledge in modern business.* London: Routledge.
5. Dewan, P. and Choudhary, R. (1995). Coupling the User Interfaces of a Multiuser Program. *ACM Trans. Computer-Human Interaction*, 2(1), 1–39.
6. Dourish, P., Bellotti, V., Mackay, W., and Ma, C.-Y. (1993). Information and Context: Lessons from a Study of Two Shared Information Systems. *Proc. ACM Conf. Organisational Computing Systems COOCS'93* (Milpetas, California). New York: ACM.
7. Dourish, P., Edwards, K., LaMarca, A. and Salisbury, M. (1999). Presto: An Experimental Architecture for Fluid Interactive Document Spaces. *ACM Trans. Computer-Human Interaction*, 6(2).
8. Garfinkel, H. (1967). *Studies in Ethnomethodology*. Englewood Cliffs, NJ: Prentice Hall.
9. Gerson, E. and Star, S. L. (1986). Analyzing Due Process in the Workplace. *ACM Trans. Office Information Systems*, 4(3), 257-270.
10. Greenberg, S. (1991). Personalizable Groupware: Accommodating Individual Roles and Group Differences. *Proc. European Conf. Computer-Supported Cooperative Work ECSCW'91*. Dordrecht: Kluwer.
11. Gutwin, C. and Greenberg, S. (1998). Design for Individuals, Design for Groups; Tradeoffs Between Power and Workspace Awareness. *Proc. ACM Conf. Computer-Supported Cooperative Work CSCW'98* (Seattle, WA). New York: ACM.
12. Lu, I. and Mantei, M. (1991). Idea Management in a Shared Drawing Tool. *Proc. European Conf. Computer-Supported Cooperative Work ECSCW'91*. Dordrecht: Kluwer
13. Mackay, W. (1990). *Users and Customizable Software: A Co-Adaptive Phenomenon*. PhD dissertation. Sloan School of Management. Cambridge, Mass: MIT.
14. Mackay, W. (1991). Triggers and Barriers to Customizing Software. *Proc. ACM Conf. Human Factors in Computing Systems CHI'91* (New Orleans, LA). New York: ACM.
15. MacLean, A., Carter, K., Lovstrand, L. and Moran, T. (1990). User-Tailorable Systems: Pressing the Issues with Buttons. *Proc. ACM Conf. Human Factors in Computing Systems CHI'90* (Seattle, WA). New York: ACM.
16. Simone, C., Mark, G. and Giubbilei, D. (1999). Interoperation as a Means of Articulation Work. *Proc. Intl Joint Conf. Work Activities Coordination and Collaboration WACC'99* (San Francisco, CA). New York: ACM.
17. Tatar, D., Foster, G. and Bobrow, D. (1991). Designing for Conversation: Lessons from Cognoter. *Intl. Journal of Man-Machine Studies*, 34(2), 185-209.
18. Trigg, R. and Bødker, S. (1994). From Implementation to Design: Tailoring and the Emergence of Systematization in CSCW. *Proc. ACM Conf. Computer-Supported Cooperative Work CSCW'94* (Chapel Hill, NC). New York: ACM.
19. Trigg, R., Blomberg, J. and Suchman, L. (1999). Moving Document Collections Online: The Evolution of a Shared Repository. *Proc. European Conf. Computer-Supported Cooperative Work ECSCW'99* (Copenhagen, Denmark). Dordrecht: Kluwer.
20. Suchman, L. (1987). *Plans and Situated Actions: The problem of human-machine communication.* Cambridge: CUP.
21. Suchman, L. (to appear). Embodied Practices of Engineering Work. To appear in *Mind, Culture and Activity*.