HCI in the wild mêlée of office life – explorations in breaching the PC data store

Richard Harper, Siân Lindley, Richard Banks, Phil Gosset, Gavin Smyth

Socio-Digital Systems, MSR Cambridge

INTRODUCTION

'HCI in the wild' was meant to be a call to get HCI investigations out of the lab into the mêlée of real life. This is of course a commendable suggestion, though begs questions about what kinds of methods and topics are suited for exploring in this mêlée as against in the lab. Claims by some experimentalists that they seek ecological validity in lab studies are largely missing the point since the thing that studies in the wild seek are essentially only those things that occur outside the lab – and hence are not things that can be replicated, modelled, or emulated. But in any case, some of those who have taken up the call for studies in the wild have taken this rather too literally – they have sought wild places, places where HCI researchers have not gone before. Needless to say this being HCI, the places in question are not often that wild, woods near Brighton, for example, street life in south Cambridge. What they ignore as they venture into these settings is the mêlée of office life, the place where the bulk of computer systems are located and the place in which, oddly enough, increasingly little HCI research gets done.

Office life is wild if one looks properly, if the label 'wild' is meant to highlight the complexities of and hybrid organisation in real-life practicalities. Consider the 'desktop', the fundamental interface of the PC in the workplace. While this might have been the apogee of HCI research thirty years ago, does it still fit the office? The technology that supports the desktop has changed, the networks that connect desktops to each other are different, the work practices the desktop supports are different, but little HCI research looks at the wild of this new office life. One wants to ask what is the mêlée of contemporary office life? How does this show itself in desktop behaviours? What impact would new HCI concepts have in this setting? In this paper, we report a project that sought to inquire into precisely this question.

Its focus, in particular, is the file and its role as an anchor of the desktop. File abstractions were defined many years ago [5]. They reflected the infrastructure and purposes of files, as documents in the workplace and provisioned by networked work-stations, as originally imagined by Xerox PARC researchers. From this beginning, the experience of digital objects has been almost entirely through the interactive architectures that these researchers devised: what is now known as the PC, and through that, the WIMP interface and interconnections to other users on their PCs via Ethernets and other protocols. We have – all of us – grown to understand what the terms files (and subsequently folders and directories) mean through our experience of them in this ecology.

Key to this context are a number of foundational design principles [5, op cit]. The file as represented by the system in the GUI is subject to direct manipulation, for example. The thing-as-seen by the user, moved and copied by their use of the mouse is pretty much the thing-that-is-stored by the computer. Also, this entity, the file, is represented at an abstract level but not merely as a thumbnail; this abstract form represents what is stored *and* things that can be done with that file. Files have powers, in other words, and hence the term 'Icon': something representing hidden functionalities.

It is perhaps worth reminding ourselves of what these powers are after all these years of daily use. It is not just that they are somewhat mundane when used to evoke the religious setting from which they are drawn but their ubiquity today makes them almost invisible. They are made up of a set of operations – Xerox defined these as Move, Copy, Open, Delete, Show Properties, Same (Copy Properties). This scoping remains startlingly similar to what is possible with current desktop machines and the Icons representing entities manifest there; there are some remarkable similarities too with new platforms – smart phones, tablets and so on.

Even so, today there are many types of files, not just office documents, spreadsheets and messages, but also images, sound files and social data, where the notion of files seems ill-fitting if not wrong – irrespective of the platform. Furthermore, the ways in which we experience and interact with these digital objects is changing. The things we do with our digital files has expanded, in other words, with sharing, copying, viewing and tagging practices being of a different order than before. And the thing that we understand our digital stuff to be has changed too – files are just a part of that stuff and even here the form these files has is now greater, more diverse.

This is nicely illustrated with image files. Over the past few years, we have gone from a situation where a digital image that exists as a single file on our PC can become a large nebulous collection of data on a Web page once posted upon a site like

Flickr. This collection can include basic details about the image (a name, description, date taken); complex data derived from the camera, mechanisms for sharing (if it is added to a 'Group'), and information about structure (both through time – the image is part of a stream of photos ordered by chronology – and through user action, if the picture is added to a 'Set'). The picture can also be tagged (by the owner or by others, and even by software), commented on, marked as a favourite, and it can have items and people within it identified, either by the person who uploaded it or by others. In computer file storage terms, this large nebulous collection of data is usually described as a graph. In some Web based systems, all data are treated as elements in a graph – and not just image files and associated data. This is explicitly the case with Facebook, for example, where the Facebook Graph File System affords new ways of interacting with one's digital data.

One interpretation of this situation would be to conclude that traditional files on the PC lack richness; that they have a coarseness in their relationship with other entities when compared with online files. Online files are formed from a cloud of metadata and file streams that hang together, but also overlap, sharing properties with other files, and connecting to other people, places and events through their attributes (for example, through friends, location and time). Therefore, one might argue, the model of the file as an entity on the Web should supersede the model originally applicable to the world of the PC.

This might miss something of importance however: it could ignore some of the values of digital entities on the PC, a value that is to be found in their relative simplicity. This simplicity might afford benefits that the apparent richness of the Web elides.

For example, it is commonplace to hear complaints from people about how they no longer feel as if they 'own' the pictures they put on the Web – on Facebook, say. As Odom *et al* note [11], users observe that what was once a thing on their desktop, an object in their file directory, becomes something different on Facebook. The former is an object they can keep or delete, put away as they feel fit; it is a directly manipulable abstraction to put it in Xerox terms. When the object goes on Facebook, it seems to lose some of these properties – some of its powers as an Icon - and starts to obtain a life of its own over which the users have less or even no control. Terms that are often coined by users to account for their relationships with digital files, like ownership, possession and responsibility, and even more crude terms like 'thinginess', seem to lose their valence; certainly the operations defined by Xerox seem pallid and inappropriate when articulating what the Icon of a file on the web might stand for.

Of course, one would be remiss if one neglected to note that it is in this respect that the value of applications (or services) like Flickr or Facebook is to be found. Their transformation of the digital entity from its form on a PC into something else allows Webs of annotation and comments to accrete around a picture, say, and thus allows the creation of a kind of social life – one for the participants, the commentators and the original creator (or poster) of the image. One might even say, following Appadurai, for the object itself [1]. The point is that these benefits, substantial though they are, come at a cost, a cost that users have no alternative but to pay: their sense of a loss control. Certainly the powers that they have come to understand are vested in the Icons they associate with their files is somehow corrupted, some of its capacities being removed while new ones are added that are unclear, opaque to view.

It might be, then, that the two types of digital world, one a domain of singular entities, files, objects as we describe, and the other a world of more hybrid, networked and interrelated entities, has been poorly integrated. Movement between the two worlds does not support the virtues of each – presuming for the moment that there are virtues to be found.

At the current time, these virtues all appear to reside on one side: the fervour for Web-based experiences more or less obscures any values that the architecture of digital entities on the PC enable. These values might relate to a sense of control that users feel is lost when they move away from this context, as just mentioned. Though, as we point out, it was many years ago when Xerox asserted the benefits of direct manipulation of an Iconographically rendered abstraction, that this could be key to future evolutions of files that might move beyond the PC is something that has been picked up by other researchers over the years. Dourish and Button argued this in their *Technomethodology* paper [3], for example, where they urge designers to make accountable (or visible) the operations that could apply to a file wherever it might be. Odom *et al's* user studies paper mentioned above echoes this claim albeit unwittingly. The complaints of users reported there might not be merely a reflection of these users struggling to learn how to deal with new experiences; it might flag *correctly identified* issues to do with systems design. Their complaints point toward their interaction with diverse architectures made manifest in abstractions and the operations applicable to them, architectures that seem to compete with one another in confusing ways. Users do not find accountable what their files can do when those files move from the PC to the Web and they are right to note this – even if they find the words to articulate their concerns awkward and fear their complaints likely to be mocked by the 'more knowledgeable'. One should not forget either, that just as the values of the PC have lacked valorisation, so the virtues of Web-based applications have not been applied or investigated in relation to PC-based architectures. If files on a PC were more like files on, let us say, Flickr, with a complex Web of data associated with each that allowed new interconnections between them all then this could give added meaning to individual files. Placing them in a Web of relationships would not just provide new ways of viewing and organizing particular ones, then, it could also facilitate understanding those individuals in richer ways. Meaning could be multifaceted, with files being viewed and managed in diverse ways, in sets and singularly; in arrays or freestanding. Of course, something

might be lost in this move toward the web-based architectures on PCs, just as something might be lost when design principals from the PC start invading the Web.

A Step along a Research Path

Many issues are thus evoked by the changing nature of digital objects, the interplay between their constitution on PCs and on the Web. In the project we report in this paper we examine some of these possibilities, in particular what is afforded when a PC-based system is given some of the virtues of how the Web treats digital objects. Whereas PCs have been architected to treat digital objects as singular, directly manipulable entities, we have been exploring what is enabled when these entities can also be engaged with as part of graph relationships – as part of associations between (or across) files where the nature of that manipulation alters and where ways of rendering digital objects alters too.

The goal of this research harks back, then, to the opening paragraph of this paper: to the question of how we define and understand the digital entities we engage with. As Dourish and Button note, the abstractions that articulate these entities, whether those entities be composites within a graph store, singular objects in a file repository or any hybrid of these (and other possibilities), should be tractable to the user and engineerable for the computer scientist. As we ourselves have argued in a prior companion paper [12], they should allow users to grasp what it is they are handling and thus ascertain how they can act with it; they should allow the engineers to embed those actions in computational specifications; and they allow also designers to render visual forms (icons) that reflect the actions applicable to digital things.

Engineering a Method

To explore these topics we developed a new, low-level data store on a PC, called CamFS. This combines the current file-centric model associated with Windows and its associated data store, NTFS with notions of files taken from the Web, particularly, graph models that describe relationships between files. These can be delivered as required, with a name value store being used as an index of these files and graph relations that allows fast access to digital data and the delivery of that content to a new GUI.

It is very important that it be understood from the outset that CamFS is not a 'solution', however. We have built it to let us inquire into the issues of concern. For example, the name value store enables us to design a new user interface that combines the rendering of files as singular entities alongside files when seen as combinations or sets. These sets can be dynamically reconstituted in real time and thus can allow a re-specification (in theory) of the meaning of individual files. In other words, CamFS allows us to visit what interacting with files through or with reference to graph relations engenders – interactions with files and their associations that were not possible or even imagined before – whilst confining that concern to a world we are deeply familiar with, our own PCs.

Nevertheless, our research does have its limits. The possibilities that CamFS enables relate to how single users interact with their digital stuff, not with how multiple users deal with shared entities on the Web. But CamFS does allow us to consider how users might mediate their files between their PC and the Web; how a new store on their PC might pattern and document their social systems of exchange. If the forms of interaction traditionally possible with files was defined by Xerox and we are considering how this might be altered by some of the styles developed on the Web, then CamFS is also allowing us to consider what is implied when users share or give a file, or associate it with other content. These considerations point towards new operations that act on digital files and the icons that articulate them, operations that lead the power hidden in the icon to be greater (or at least different) from that specified all those years ago in Palo Alto.

Related Work

For reasons of space, we can only point to our own prior review of papers that discuss research on file directories on the PC [7]. This research is substantive and very rich, as we remark in our assessment. Over the years, investigations have considered how to improve the efficiency and ease of use of files by placing them as entities in a database, for example, and has explored how threading between different file types can better reflect work tasks – linking emails and Word documents, PPT and spreadsheets. Tagtivity [12] also pointed towards how interactions on the PC are bleeding into interactions on the Web. Research looking at the Web itself has tended to avoid addressing the issue of file abstractions as a topic, though as we note above there is considerable research on how users struggle to make sense of their experiences of digital file management on the 'domestic' devices and in the semi-public world of social networking sites – and thus the Web more generally. This points back to the need to reconsider what the concept of a file might be, and how it ought to evolve in ways that allows users to better manage both their own data stores and those of applications such as Facebook.

Our Research

It was with this as background then that our research entailed building a new data store that would allow Web-like interfaces to be developed and then 'lived with' on our PCs for a period of several months. Two of the research team committed to doing this full time during this period, as many of the files these individuals were routinely dealing with would be stored within and rendered by the application. The resulting experiences of use was documented. This documentation entailed noting particular problems of understanding and use, system errors and failures, as well as more encompassing reflections on how file entities

and their rendering in the GUI were experienced. Weekly meetings were held to share those experiences and to guide future note taking. In all, the system has been used continuously for six months; it is still deployed.

Given what we say about the status of the system – its nature as a research enabling tool – we propose that our study consists of a kind a breaching experiment, a formulation originally coined by Garfinkel [6]. This has been extensively reported in the HCI work of Poole, for example [13], Crabtree [4], Tolmie & Crabtree [15]. It has its echoes in other studies that draw on 'reflections in use' [2] and 'autobiographical design' [10].

Key to breaching is the claim that much of the interaction that is possible on and through computing entails tacit and taken for granted common sense practices. These constitute a body of know-how that enables users to focus on certain aspects of interaction while taking for granted others and also provides a resource for reference and constitution of meaning. Breaching entails uncovering some of the 'taken for granted' through altering the ease with which activities can be done – through altering the interface say, or making the functionalities of a data function in a visibly different way.

The system itself, CamFS, works as follows. It augments the PC file system, allowing the benefits of a graph relationship to be tested within an environment that the user knows well, namely their own computer. To begin with, existing NTFS file structures have two elements added. The first is the ability to connect arbitrary items together on the same or on different pieces of hardware through the implementation of a graph system. The second is the ability to define simple layers of data through the use of a name value store. These two elements, the graph and the name store, enable us to build a file system that allows for some of the subtle interconnectivity and extended properties highlighted in Web-based experience, allowing us particularly to play with aspects of *property* and *structure* that are unified under the term name value. These aspects have a fluidity and ambiguity in online 'files' (such as a Flickr photo) which, through CamFS, we can bring to bear on a more traditional file system, on a PC.

Our goal with CamFS is thus is to show a simple evolution from a world in which the *folder* is paramount as a structural concept, to one in which other structures are allowed, and from a world in which the properties of a file have to live *in* that file in order to exist, to a world in which properties can be layered on top of files as needed but without the requirement to abandon the essential characteristics of a file itself.

As a simple example, we might want to connect multiple files together because they are part of the same project that we are working on. We might then give the connections between those items the same name (the name of our project), tying them together, in essence, with a tag; a *name value*. These are concepts that are common online, but rare in our operating systems. As it happens, users can create such connections on PCs through adroit management of folder hierarchies; as we mention above, too, some effort has gone into affording threading of files on the PC already. Our system however embeds what have typically been interface designs into the data store. In the following, we describe the back-end to CamFS in more detail, before outlining the user interface.

The system

CamStore consists of a file system, a graph store and a name value store. In a typical file system, digital objects are single files. The file system itself is constructed as a series of nested folders, and the digital objects or files sit within one of the folders. The folder structure is usually either created for the benefit of the Operating System, for the applications that run on it, or for the benefit of the user to be able to make sense of and find their files. In a graph file store, the graph consists of a mesh of connected objects, where the objects are the nodes of the mesh, and the relationships between the objects are the connections between nodes, usually called edges. It is common in a graph for nodes to be multiply connected, thereby being related to many other nodes. By carefully constructing the appropriate graph, it is possible to store objects and the relationships between objects so that information about the system and the relationships between objects can be retrieved quickly and easily. Thus, a single object can be the member of many sets, a single node can be related to many other nodes through various relationships, and the properties themselves are raised to have the same status as the files that they belong to, and can be used as a way of navigating between file objects. It is these that are defined as 'name values'.

To make CamFS non-destructive to the user's existing file system and thus allow the sense of real files being used in real ways, it was decided to leave files within the user's file system, and simply to add file nodes that stand as proxies to the files within the graph. Thus, in CamStore, the file store behaves like a typical file store, but files can also be nodes of the graph. The graph part allows for relationships between nodes to be defined. To move between the file system and the graph, a name value store is used. This store provides a fast information retrieval mechanism.

To leverage these functions, certain categories or file type features had to be predefined. Without these, the graph fields in CamFS could not be utilised. Necessarily there was some arbitrariness in this selection.

First, the *thumbnail* of every file is taken and recreated in a number of sizes to be used in CamFS to display item and set information. Thumbnails are a particular type of graphical representation; others are available – standard renderings with

names, for example. Choosing thumbnails allows the GUI to be devised to echo some of the forms data stores take on the Web. Second, and relatedly, the file *type* is determined. This is achieved by testing the file extension against a known list, and ascribing a file type for each extension. For example, file extensions .jpg, .png, .tiff, thumb and .bmp all correspond to file type Image. This allows CamStore to retrieve all files of the same type, to request thumbnails if available and if not to render standard forms, and for the resulting amalgams to be displayed. The defined types are: Image, Music, Video, Document and Application. Type is added as a property node in the graph. Third, the *date* associated with the file (of whatever kind) is determined. For most documents it is taken to be the Last Write Time, although for images the Date Taken time is used. The date is then broken down into a number of properties, so that all items from the same date, month and year can be quickly displayed. Fourth, every folder in the file path is used to create a *set* that the file belongs to. For example, a file with the path /Photos/Barcelona/Gaudi07.jpg would create sets Photos and Barcelona within the graph and link the file node to both. Additional attributes are included with a view to supporting further development of CamFS - these include owner, location and device.

To use CamFS, a user is required to move files and folders to a specific directory, typically called MyCamStoreFolder, within their file system. Once this is done an application called CamStoreCreator processes the files and folders within this folder and creates the CamStore. CamStoreCreator works by creating a list of all of the files that lie under the defined location, and then processes each file in turn.

Upon opening CamFS, the user is presented with a homepage displaying three groupings of objects: a group of Favourite items, a collection of recently interacted with sets, and a collection of recently interacted with items (see Figure 1). On first use, the Favourites section is empty, and recent items are organised according to the dates in CamStore.

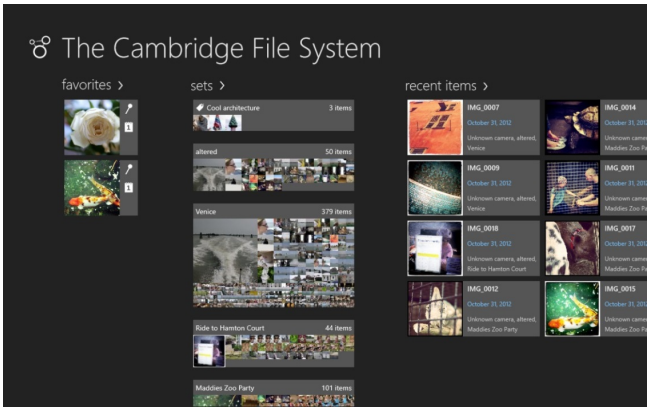

Figure 1. CamFS home page.

The expectation from the outset is that the interface becomes richer over time and through use. For, in addition to using the sets that are automatically created by the system, date, type and so on, the user can create sets, add objects to existing sets (which also adds that set to the recent sets list), and mark items as favourites (which causes them to appear on the Favourites list on the left, in the order by which they were made a favourite).

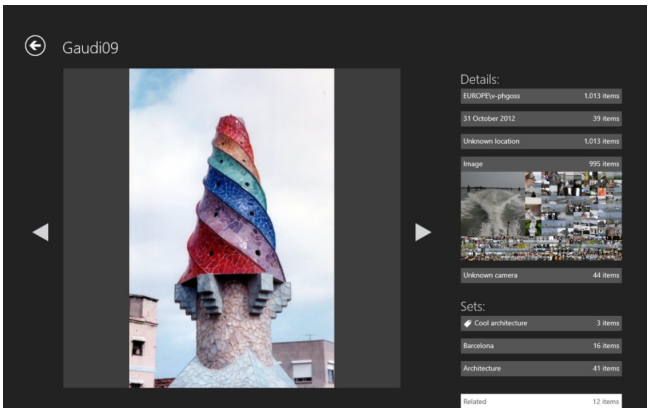

Figure 2. CamFS item page with set expanded.

The items themselves are presented in such a way as to highlight the graph-like relationships present in the system between file objects. In Figure 2, the right hand side shows a predefined set of details of the item, namely owner, date, location, type, and as it is an image, what camera took the picture. In addition to the details, a predefined subset of the sets that it is a member of is also shown. It is a member of the sets Cool architecture, Barcelona and Architecture.

In addition to the sets, a list of 'related items' is given, which is a list of items created before and after the current item. All of the above sets and lists are relationships that are constructed within the graph. If one of the lists, e.g. Image, is clicked, then that set is expanded to give some sense of what is in that set. So each individual item connects not just to the details of that item, but also to the sets of items that are related to it, and these relationships support navigation through the system.

USING CAMFS: A BREACHING EXPERIMENT

As should be clear, building CamFS took considerable effort; as it happens well over a person year in coding. These efforts produced a system that not only offers a fairly comprehensive and fully interactive interface but is robust and extremely fast. Users – us – really can engage with our digital stuff in and through the system. Our approach to using CamFS has been (and continues) to be one of engagement with it as far as possible: that is to say to treat it as a practical, real worldly tool, and as we do so, to see what it evokes, enables and prohibits: to force us thereby (through *breaching*) to provoke our 'sluggish imagination' into insight about future forms of data stores on PCs and on the Web, as well as their interaction.

Typologies, topologies, arrays

As described, various categories (and ways of grouping) were designed into CamFS at the outset. These were chosen as initial ways of uncovering potential values. Our experience of the system made us consider both these and different typologies and topologies in relation to sets and their constituents, files. It also led us to confront some of the taken for granted benefits of file hierarchies – a form of set making though with different consequences for what is displayed (and displayable) in a GUI.

Scale of Arrays

One of the most notable features of using CamFS is the contrast it highlights between how file hierarchies hide content whilst this graph-based data store makes content visible. Or, to put this another way, CamFS reveals files that are hidden in Windows Explorer, a file hierarchy interface. This was especially striking when CamFS was first used, this being shortly after building a CamStore directory (that is to say, soon after a selection of files had been copied into the MyCamStore folder), and thus when we had thought ourselves confident that we knew what would be included in the system. Yet, when confronted with how CamFS rendered content, we were surprised – if not taken completely aback – by the volume of files to be seen. The surprise here was great not because we were completely unprepared for a different rendering of their content, but because of the scale of the content. With CamFS, the enormity of the number of files that are ordinarily produced through normal working practices becomes visible.

To give an example, on one of our machines, a folder entitled 'Publications' accessed through Windows Explorer contains 13 sub-folders, 3 Word documents, 1 XML file, 1 RTF file and 1 text document. Opening the equivalent Publications set in CamFS presents an incredible 1410 items. Before saying anything about this number, it is perhaps worth remarking on what they are and how they can be associated through CamFS. Arranging these items by Date, for example, reveals that the first document was created on the 5th of January, 2004. Arranging them by Type reveals that the contents are not only Documents but also Images, Videos, and Applications (there are 10 of these), as well as two files with no extension.

Be that as it may, the bigger question is why this number is so considerable. In this case, the self-defined Publications folder acts as a kind of 'working documents' folder for this colleague. It collects what she is working on 'now', if you like. But, as it happens, all the 'worked-on-already' documents are also filed away, put in a 'done' folder that also sits in the overall Publications folder. This includes entirely finished papers and prior versions of ones that this colleague is working on at the current time.

When this colleague uses Windows, and hence File Explorer to render abstractions of these files, only the ones currently being worked on are shown at the top level. Each file sits within a folder, with one folder being parent of all others. From a File Directory design point of view, this is a virtue: nesting allows occlusion and the efficient use of screen real estate; it gives organization to the files. This in turn makes the GUI tractable to the user – in this case these folders have got every version of every paper this colleague has worked on but does not make these available all at once. This larger set, for want of a term, includes all the files this individual is working on at the current time and those they are keeping 'just in case'; it includes in addition to this 'completely done with' files too.

To be sure, our colleague is like many 'users'; like them, she produces and seeks to engage with numerous files, so many in fact that they cannot all be accommodated on a 'desktop'. As digital documents of all forms have become more prevalent in the workplace (and elsewhere) the number of files that need filing has increased. Solving this problem – or rather offering

design solutions to it – is reflected in contemporary file directory design: in File Explorer and so on. As we say, this is how the tension between real estate and the need for organisation has come to be solved.

CamFS inverts this, however; or rather, it forces one to reconsider the benefits that ensue. The set of 1410 mentioned above alludes to both the need to manage one's digital output in a way that does not result in overload (visual, cognitive, say) at point of interaction, on the desktop say. But it also points towards a different question: whether one's efforts are produce an undue surfeit. To put this provocatively, an entirely unexpected consequence of using CamFS is a kind of schizophrenic frisson: we come to ask whether we are industrious and producing a good volume of stuff or if we are deluding ourselves into producing too many things: in this view, volume is a measure of franticness, of papers for papers sake.

Confronting the number of our digital files does not make us *only* grateful for the benefits of file directory design, then. It leads us to think twice about the relationship we have with our digital affairs: it points us towards housework minimally; sometimes more searching questions. Most obviously, it suggests that filing away does not need to be a difficulty for a graph based system; rather, a graph based system might provide a handle on what needs sorting, what needs doing and what measures might be applied to oneself and one's output. It might allow the user to see at a glance their digital stuff so that they can then embark on the right subsequent course of action. From this view, they might determine what is best 'hidden', what needs safe-keeping and what should not be forgotten but should be actively dealt with: thrown away perhaps; it might even induce them to reduce their output, to be more consolidated in their affairs.

Visual Functions of Arrays

In Xerox's original design of the desktop, the decision had been made that icons should offer some differentiation in the way that they represented different file types, but all types were to be icons – with the 'hidden powers' listed above justifying use of the term. CamFS, meanwhile, relies upon thumbnailing file types. But in our use of CamFS we have come to see that the glance-ability of different thumbnails varies according to the properties internal to the file in question and this is consequential for how one engages with those files.

For example, in the case of one of our documents being prepared, CamFS takes from Word a thumbnail; this renders the first 'page' of the file in question, the Word '.doc' file. But this creates problems when comparing multiple versions of a document which often appear similar when seen as thumbnails. After all changes in a Word file are more likely to be toward the end of the document than on the first page. The first page – with a title, author, say - is likely to look similar across multiple versions.

On the other hand, thumbnails of PNGs, of pictures, are very much more useful. Indeed, using CamFS does bring home how useful it is to see picture files alongside other picture files. We have found that some of our individual pictures, often created as part of a series, seem better understood as such; that is to say as part of a series of pictures taken at various points in time or from some particular vantage point. Their unity provides value as does their dissimilarity; the distinctions part of their charm as self-created visual objects. Word files often have no similar value delivered through bundling them into an array or set of thumbnails; as we say, sometimes only muddling results.

In sum, the turn to thumbnails in the CamFS interface results in eliding some of the relevant properties of file entities while privileging the properties of a particular subset. One might add that the latter are those files whose features point towards something essential in the thing in question – a thing to be seen at a glance, as against a thing with a more complex visual geography. One is a picture, the other a text of sorts, a document say of several pages.

This feature of CamFS also draws attention to what actions different objects permit beyond the simple distinctions alluded to here (display/not display, set together, set apart). Different actions have distinct values in the context of file types. 'Create' makes sense for 'documents', but not quite for 'photos'; for Word files or .tiffs say. Similarly, to 'favourite' may mean different things for different types – PPT or mp3. Favoriting could serve as a reminder for action with regard to some presentation (as exemplified in accounts of Tagtivity) or it could signify that an mp3 file is indeed a favourite song.

Arrays and topologies as performative

CamFS use also highlights something else that was, presumably, of no concern to the Xerox researchers. As we have remarked, when first confronted with files rendered in CamFS, one can be taken aback by the totality of one's digital stuff. Leaving aside the question of the housework this provokes, something else is also brought to the fore: the possibility that display itself is a value. We are led to wonder whether hitherto our desktops have been organised to make their arrangement a matter of private engagement, while with CamFS the question of whether others might want to engage highlights whether this engagement might be of a different order.

A comparison that was brought to our mind might help here. There are various types of libraries, of which two can be noted. The first is the one most frequented by readers of this article: an academic library. The other is the domestic or personal library. Each affords a different function and this is reflected in the role of display in each. In one, the purpose is to store content so as to make it conveniently available to those who need it. Seeing the content is thus not so important as being able to find it. The

invention of the duo-decimal index reflects this very need: it allows users to identify where content is without having to search through that content itself. They do not need to look at the stacks of books and journals to find what they want. In contrast, the purpose of a personal library, the one kept in the home, is not so much to be an information repository as it is intended to display cultural acumen and reading history. The display of books acts as a symbolic representation of the owner, of their cultural capital. When we use CamFS we realise that our design merges and mixes this functionality: it certainly is beneficial to see the files we are working on, but only insofar as one can thereby get more directly and easily to the stuff in those files: to the site of work, as it were. But we also find that CamFS does not offer the kind of visual economy of the duo-decimal index of a library. Instead it points toward and enables a desire to display one's activities, not to engage with them. Indeed, we find great delight in demo-ing CamFS for this very reason. Our managers too seem to relish the opportunity to 'show it off'.

Leaving aside the foibles of research cultures (our own or anyone else's) there will be, clearly, times when this showability is just what is sought for. If on one's own desktop, there might be less value for such display, this might well become a value when the location of our digital stuff is available for public view. There the social act of display might very well be the sought for value – as indeed our managers' keenness attests to. There will be nuance here, too, when it might be that a person wants to create a public version of their own desktop and one that is as it were private. In a sense, one might be like a studio of one's stuff, arranged to celebrate one's industry, this will be public; while the other is more like a painter's hidden palette – a surface with various resources and components, but not something very edifying to look at, its centrality in the creative process not-withstanding..

The Work Before Arrays Become Possible

Making these (and potentially other) dimensions explicit points towards questions such as how a graph representing a collection that is curated might differ from a graph that encompasses multiple versions of the same essential document that has yet to reach the stage of needing curation – of being worth 'showing'. Given part of the purpose of CamFS is to highlight issues to do with the interaction between and relative affordances of Web-based file types and those on the PC, this particular issue seems salient indeed – it points to the problem of just when files can start moving from the PC to the Web and just what sits in-between.

This issue pertains to files that have reached a threshold, one where they can begin to have some kind of social life – in display, for example. But the opposite end of the life cycle of digital entities became increasingly prominent as we leveraged CamFS in our daily affairs. We came to see there is a cusp between set making and what comes before; a moment when one is engaging with entities in ways that cannot be effectively articulated with reference to graph relations as represented in CamFS. Our digital entities have, as it were, a pre-graph status, a condition before they can be 'set' so to speak and before questions of show-ability make sense. Certainly CamFS has a home page, a screen that presents favourites and so on, but essentially CamFS is a tidy place, a display of organised files – or those files that are in a state where 'setting' is a sensible thing to do. But CamFS does not let us identify the 'just whatness' of different entities. And this 'work', if work it is, is something we came to realise we do on the desktop.

There will be, doubtlessly, various ways that desktops are used – and indeed the HCI literature is replete with such studies going right the way back to Malone's thoughts on 'piles' [9; see also 14]. Without wanting to review that literature too carefully for reasons of space, what one can note with our experience is that ordinarily a desktop is the site of a diverse practices that reflect a combination of 'things done' and 'things to do' as well as a place in which things exist 'that are still undefined'.

We discovered, also, that these sorts of practices are not only on the virtual desktop, though at least one of used that space for this 'work'. One other used a combination of applications in ways that is analogous: as a place where those entities that are still undefined sit before they are dealt with. In their case the applications were email and OneNote.

In particular, files already created by some Other person join this person's desktop via email.. The particular moment of their arrival, their form and volume, is, of course, contingent – dependent upon another and hence to some degree unpredictable. Our colleague finds some of the things sent via email easy to 'put away' on receipt (into one set or other, into a folder or a list of to do items), while other things are harder to sort out. Some remain in the in-tray of the email tool; some are saved into the folder structure. As it happens this manner of work with email is well documented [17].

Our colleague also makes notes and jots down ideas in OneNote. OneNote is used to make raw text notes, collect images, Web content and so on. Again, and as with email associated material, there is a contingent aspect to this, it being uncertain beforehand how much stuff gets produced through and kept in OneNote; only some of it is eventually transferred to a Word document or some other file, thus finding itself part of the folder hierarchy.

The important point is that this colleague has various objects at hand, objects that will end up in different forms and places but the specificities of which might be unclear at any moment in time. They are, if you like liminal. The tidiness of CamFS then highlights the possibility that there needs to be a special place for this work; a place, properly, of the liminal; of a state of disorder that one attends to so as to create order.

This has implications for the relationship between this place, graph-based data stores and resulting GUIs, and the Web more generally. One possibility is to treat the 'desktop' (or its equivalent) as analogous to the bowl by the front door; that is to say as a domain where things come to and get dealt with in various ways but which at any moment consists of a heterogeneous set of unclassified stuff. Or, rather, as a place where the applicability of classification as implied by graph-based systems like CamFS might be misleading. Those classifications are subordinate to or subsequent to other work.

But this other work could nevertheless be supported by graph based stores – if not in the way CamFS does – tidily, for display. After all, this work entails classification: users are distinguishing, firstly, what a thing is – not its computational nature, but what it represents in relation to their doings. They identify it as, say, something that can be thrown away (deleted), or something that points to work in the future, or a confirmation of something that has been done, that is now in the past. Some of these relations might be helpfully rendered in terms of creation and production: in the work of the palette to use a metaphor from above.

Once this work has been done, then a new place might be found in which the graph-based data store instantiated by CamFS might be ideal. This could afford some of the benefits of juxtaposition between graph stores and file hierarchies mentioned above – most obviously the contrast betweenmaking visible on the one hand and hiding on the other.

It also points towards what might go on Web-based stores – ones beyond CamFS if you like. For these are places of yet another sort: ones where the display-ability of sets is perhaps the main function, when the things in themselves are not to be represented as icons with hidden powers but as merely symbols of the creator, the one who organised them as such. In this respect, the Web is then not a place for back-up and storage, but a site of performance through what is chosen to store and display. Already users of Flickr recognise this, treating their accounts not as repositories of their pictures but as devices that show their élan at picture selection [8]. Our research on CamFS is suggestive that this might be a more widespread phenomenon, albeit not well supported at the moment.

DISCUSSION

As should be clear, investigating a new data-store like CamFS does not allow the kind of enquiries typical of CHI papers – a design with user testing in the lab or 'in the wild'. It does require, to be sure, engineering and giving the result of that engineering to users. But here those users are ourselves and this makes the testing peculiar, or so we have argued. Our engagement with the technology is not, as it were, intended to let us figure out how to appropriate the technology; our research has entailed engaging with it so as to fathom our own imaginations made real through use of the technology.

Key to this has been unsettling some of our taken for granted practices with files and their abstractions as well as pointing towards hitherto un-thought of possibilities. As we seen, we have found ourselves revisiting questions that we thought long dead in HCI while at other times have come to see current issues and topics in HCI in new light.

Graph Stores and File Abstractions

It seems entirely sensible to keep a history of changes to a file (a document say) during the process of its production. Indeed, on the Xerox Star system, the associated word-processing system supported the ability to annotate text and to keep records of changes and thus the history of a file. In the case of Xerox, this evidence of history were, however, embedded in the file. Consequently, in terms of file abstractions, there was only ever one. Versions of the file were visible *only* through engagement with the file icon. Since that time, however, our practices have evolved as has our treatment of file abstractions. The long and short of these changes has been that the singularity of the file abstraction in Xerox's design has been replaced by the production of multiple files to record this history. Our own process of producing this paper attests to this – as will be recalled one of our folders had 1410 items in it and this included *versions* of files; multiple abstractions and hence, multiple icons muddle evolution (history) with category (what a file is about).

It seems to us that rendering all files and the history of work with them through icons seems misleading, or at least begs the question of what a thing is - a topic or a history. What seems implied through our use of CamFS is that version history might be better when not given the same status as a file. The singularity of the file abstraction that was central to Xerox's design could be preserved (or reinstalled) in current designs, and the display of 'versions' only be possible through the selection of disclosure once 'inside' a file.

If this were the case, then the operations on the file abstraction operand are as follows. **History** becomes a **property** of the **Icon,** along with **Move, Copy, Open,** etc. Though the operand might remain singular, this does not mean that types of history might not be plural; that the operations on the operand, the file, might be varied. Authorship of changes might be displayable, for example, or the degree of change across versions (e.g. 500 new words have been written), or the sections where these changes have been made (e.g. I'm writing mostly in the 'using CamFS' section).

Second, if it is the case that some files have a history, that they are as it were 'works in progress', then our experience with CamFS suggests that there is an important distinction between these and those files that have reached a state of 'completeness'.

At this juncture, the history of a file no longer has the valence it once did. Moreover, just as history is no longer pertinent, other operations too, like editing, lose their relevance once this state has been reached.

As it happens the importance of this distinction is we think elided by a common practice that we ourselves abide by. Typically, we manifest this distinction by converting the file in question from one application to another; in the case of word processing files, for example, we change the file from Microsoft Word to Adobe PDF format. This transformation is used by us as it is by many- as a way of announcing a shift in what can be done with a file. Of course, the assumption here is that none of us have a full Adobe license and therefore the closure on the editing of a file occurs when the move in application is made. The point is that this shift in status might be consequential in terms of operations on the operand. In effect, what was once an icon when rendered as a Word file becomes a symbolic representation when presented as an Adobe Acrobat file. One alludes to interaction with its content, the other represents a sealed nature as regards the file in question.

Be that as it may, while both these entities might be kept for the record – the file with-a-history and the file-as-finished-object – the former is likely to be filed away, while the latter may be stored and shown, displayed say, shared and even hosted online. One should not think that actions on files cease when they file's contents become stable, then; that operations on operands end. It is, rather, whereas clicking on the icon representing unfinished files opens up the insides of those files for editing, changing and reviewing, so in this new context, editing has to do with the relationship between one file and others. It is, as it were, actions external to the file that become possible at this point.

We think that this shift should be manifest in the design of how the entity is rendered and in terms of the operations possible in ways that is more robust than the ad hoc move between Word and Acrobat. If icons represent powers with regard to the internal features of a file, then another form of icon needs developing that indicates a file has shifted its status from 'in preparation' to 'done', and which privileges interaction related to **external features** as against **internal ones**. Of course, it is currently applications that normally provide these external functions – as in CamFS. What we are learning through using CamFS, however, is that it might be better if the abstraction articulated this for the reasons just described.

Since the actions in question are related to the connections between the file and other entities, we propose that this rendering be called a **gossamer**, alluding to a spider's Web and yet a shroud at the same time. Operations on the gossamer operand might well include the already defined Move, Copy, and Open, but might also include **Set.** This is already an operation possible in CamFS. Set might be better articulated as **Link**. In any case, the purpose of this action is to subject the file in question to a move, one that puts it in a nest. This nest can simply be a collection of similar items or a context in which tags, comments and other metadata can be associated.

Third, our experience with CamFS has suggested that there might be a relationship between these different functionalities and the place (albeit virtual) where this interaction occurs. As we noted, the virtual desktop affords 'place' while combinations of applications can aswell but the important point is that however constituted, this is a place where patterns of engagement with file entities combines the creative process along with what we called pre-work. This kind of work is, in a sense, essentially private; the output of that work, the defining of files into categories, into things that can be set, equates in our mind with a move from back stage to front stage, from messy and inchoate into tidy and ornamental, even visible. However, this is not then strictly a move from private to public, but a move in performative value.

At the same time, the way that CamFS provides a way of looking also draws attention to the unique status of the desktop as a domain of a particular form of engagement – here the value of graph based rendering is of another order. Our research also points to the possibility that graph relations might also be of value in working with files that are still being produced. Graph relations can highlight histories of changing files, they might highlight connections between different sources and entities – Tagtivity comes to mind.

If this is so, if there are then at least two places in our digital lives created by the introduction of a graph store, then it seems to us that a fourth consequence from our inquires is the idea that the location on which a file is displayed, namely the desktop or some version of CamFS, should also implicate the functional unit, the abstraction. Operations possible on the icon representing that abstraction should reflect this context. As we have alluded, the difference here might be related to the distinction between 'internal' and 'external' functions.

It is also clear from our experience that the location of one or other of the two fields is not fixed in cement; sometimes files that are finished come back to life and need to be re-edited. Sometimes one wants to show things that are not quite 'done'. To facilitate this we think that a combination of desktop and a graph store like CamFS should allow easy movement between their respective domains, as in a drop and drag process for an icon. Once an icon is dragged into CamFS it becomes a finished object; when dragged back onto a desktop it becomes open to editing again.

CONCLUSION

We have covered a great deal of ground in this paper and have elided many important topics. Nevertheless, we think a number of insights have been produced thus far, which we have elaborated. In summary,

- The first has to do with relationships between file abstractions, the history of file content and the finished state of a file;
- The second, the distinction between engaging with the internal contents of a file and external operations that affect the placing of that file alongside others;
- third, the functionalities of File Directories in putting away or hiding content and the reverse role of graph directories like CamFS in making work outputs visible;
- Fourth, the relationship between domains of interaction that might be engineered in the future – one being creative, for pre-work, another for display;
- And last, the role of graph-based stores on the space for this creative work, for the management of the liminal, the un-sorted, the incomplete, the stuff of the mind if not for the gallery.

REFERENCES

1. Appadurai, A. (Ed). *The Social Life of Things: Commodities in Cultural Perspective*. Cambridge University Press, Cambridge (1986).
2. Boehner, K., Sengers, P. and Warner, S. Interfaces with the ineffable: meeting aesthetic experience on its own terms. *ACM Trans. Comput.-Hum. Interact. 15*, 3 (2008), Article 12.
3. Button, G. and Dourish, P. Technomethodology: paradoxes and possibilities. In *Proc. CHI 1996*, ACM Press (1996), 19-26.
4. Crabtree, A. Design in the absence of practice: breaching experiments. In *Proc. DIS 2004*, ACM Press (2004), 59-68.
5. Johnson, J., Roberts, T., Verplank, W., Smith, D., Irby, C., Beard, M. and Mackay, K. The Xerox Star: a retrospective, *IEEE Computer 22*, 9 (1989), 11-26.
6. Garfinkel, H. 'A Conception of, and Experiments With, 'Trust' as a Condition of Stable Concerted Actions', in O.J. Harvey (ed): Motivation and Social Interaction. New York: The Ronald Press, (1963) pp187-238.
7. .
8. Lindley, S.E., Marshall, C.C., Banks, R., Sellen, A. and Regan, T. Rethinking the web as a personal archive. In *Proc. WWW 2013*, (2013), 749-760.
9. Malone, T. How do people organize their desks?: Implications for the design of office information systems, ACM Transactions on Information Systems (TOIS) TOIS Homepage archive, Volume 1 Issue 1, Jan. (1983), pp99 – 112.
10. Neustaedter, C. and Sengers, P. Autobiographical design in HCI research: designing and learning through use-it-yourself. In *Proc. DIS 2012*, ACM Press (2012), 514-523.
11. Odom, W., Harper R., Sellen. A. and Thereska. E. Lost in translation: understanding the possession of digital things in the cloud. In *Proc. CHI 2012*, ACM Press (2012), 781-790.
12. Oleksik, G., Wilson, M., Tashman, C. Rodrigues, E.M., Kazai,G. Smyth, G. Milic-Frayling, N. and Jones, J. Lightweight tagging expands information and activity management practices. In *Proc CHI 2009*, ACM Press (2009), 279-288.
13. Poole, E.S. Interacting with infrastructure: a case for breaching experiments in home computing research. In *Proc. CSCW 2012*, ACM Press (2012), 759-768.
14. Sellen, A. and Harper R. *The Myth of the Paperless Office*, MIT Press, Boston, (2003).
15. Tolmie, P. and Crabtree, A. Deploying research technology in the home. In *Proc. CSCW 2008*, ACM Press (2008), 639-648.
16. Whittaker, S. and Sidner, C. Email Overload: Exploring Personal Information Management of Email. *Proc. CHI 96* 276-283