Text at Work: Mundane Practices of Reading in Workplaces

John Rooksby
University of St Andrews, UK
jr@cs.st-andrews.ac.uk

Introduction

Books, documents, letters, notes, marginalia, emails, status updates, text messages, spreadsheets, filenames, keyboards, posters, labels, notices, manuals, instructions, dials, gauges, clock faces, receipts, tickets, road signs, name badges, business cards, computer code, lists, maps, … this list could go on for a very long time. Written language and diagrammatic forms are pervasive in our everyday and working lives. Text is rarely far from our sight, and is associated with more or less everything we do. Not only is such a list testament to the amount of text in everyday life but to the great variety of textual forms, and the great diversity of practices associated with them. Text is often related to its surrounds and is used as a part of navigating some activity. Consider how the buttons, dials, nozzles, and sockets on things like telephones, ovens, coffee machines, petrol pumps, medical infusion devices and so on are usually marked out and described by text. Consider how electronic displays, dials, gauges, clock faces and so on predominantly use textual indicators in order to inform us of something. Consider how signs, labels, menus, and maps all routinely employ text and diagrams to help us navigate, select and act on objects and places. Sometimes we are 'just' reading, for example reading a document, or reading a book. But consider how text is not only central to most books, magazines, printed documents, newspapers, and so on, but how these also have things like page numbers and section headings to help us navigate them. Consider how text is routinely displayed on computers, not just in the documents, emails or web pages that we might be reading, but also on the addresses, menus and toolbars we use to access, navigate and edit such text. Thinking more broadly about our activities, consider how text often supplements images on TV screens, for example TV shows have titles that let us know what we are watching and can be listed in guides to help us plan our watching. Consider how going on holiday might involve reading a guidebook and sending postcards. Consider how cooking could involve reading a recipe book step by step as you proceed, and doing things like setting a timer. The examples given here emphasise the constituency of text in mundane, everyday activities. They emphasise, as the following quote above from Watson states, how text helps us orient ourselves to that activity, occasion or setting and make sense of it.

"…virtually every recognisable activity in our society has its textual aspects, involving and incorporating people's monitoring of written or other textual 'signs' – text that, in a wide variety of ways helps us to orient ourselves to that activity, occasion or setting and to make sense of it." [1 p7]

The chapter will concentrate on text and reading in the workplace. Go into any workplace and you will see people reading and writing, and more often than not, this reading and writing will either constitute or be of consequence to the effectiveness, safety and outputs of that work. And yet, this grossly observable fact, that people spend a great deal of their working lives reading and writing, is rarely the subject of mainstream investigation (at least it has become something of a niche for ethnomethodology). This chapter will consider the work of reading text, the means and methods by which people read text and the ways in which text is written, designed, positioned, and so on with respect to expectations about how it will be read. The examples used in this chapter relate to doorways, to health records, and to computer programming. The examples are all loosely related to technology design, which is something that often throws up questions, problems and frustrations to do with mundane practices, especially when technologies are designed badly. In one sense then, technology design gives us a footing for investigating mundane work practices. In another, it reveals the importance of investigating these. Designers ought to be familiar with the practices they are designing for, if that design is to be done well [2].

Signs on Doorways

As an example of mundane reading, consider how text is often placed on or beside doors [3], particularly on or beside doors in public or organisational settings. This text might be an identifying name or number; information about access (such as a list of opening times, a sign saying "staff only" or "emergency exit", or a note saying "back in five minutes"); it might be a poster or information sheet to catch the eyes of passers by; or it may be an instruction about how to operate the door. Such operating instructions are usually short, for example "push" or "fire door, keep closed". On some more complicated doors, these instructions are longer, for example emergency exits on commercial airliners have lengthy instructions. The main door at my place of work also has a lengthy set of instructions:

> NEW AUTOMATIC DOORS
> To exit push these doors and then swipe card reader on left hand side of main door. Outside door will open automatically. All doors will then automatically shut.
> Everyone <u>must swipe out</u> even if the door is open. You will have a problem getting back in if you don't!
> DO NOT USE PUSH BAR ON RIGHT TO OPEN. THIS IS FOR EMERGENCIES ONLY

Unlike the emergency exits on airliners, this particular door is in regular use. For a door in such constant use, these instructions are comparatively long. They are written on a laminated sheet of A4 paper, total sixty-six words, use three colours, with some parts emphasised by capital letters, and other parts underlined. The door itself is not like most doors, and because of this it is not intuitive to use. It leaves many people stuck. Often people are baffled about how to get in, particularly about why their swipe card stops working (they forgot to swipe out). Similarly, people are often baffled about how to exit. We regularly find that people have used the push-bar (intended only for emergencies) to exit. If this is done late at night, the door can be open until the next morning. The door was designed to increase security, but has not been wholly successful. The instructions have been put up in an attempt to repair the shortcomings of the way this doorway has been designed, to make clear how it functions. However these instructions have not been wholly successful. People do not read them, people do not generally stop to read instructions on doorways, at least not until they can find no way of getting through that door. Everybody knows how to use a door, you push or you pull it and maybe operate a handle. Only when immediate actions do not work do people usually stop to find out how doors work. This might be to read instructions, or if you are not alone, to stand back and see what someone else does. Lengthy instructions are not an ideal way to get across a message about how to use a door. In fact, the person who wrote these instructions knew full well that they would rarely be read. But with other options being too expensive (say changing the door, or employing someone to operate it) the sign is the best that could be done. The way that sections of the text have been emphasised using different colours, capital letters and underlining is not for artistic reasons but to show that there are multiple important items in this text, to capture peoples attention, to show that the reader needs to pay attention to several things. It is an effort to grab attention where often there is little.

The shortcomings, limitations or complexities of technology often have to be repaired through added explanation. This can be verbal, for example narrating or explaining incongruous outputs [4][5]. Or it can be done with text, for example warnings about the correct usage of computer equipment [3][6], or indeed lengthy instructions on how to operate a door. Such repair is not the only reason for placing text by a technology (for example Grint and Woolgar [6] discuss how labels on computing equipment can serve to define, enable and constrain legitimate, serviceable use). The example above is intended to highlight that there is a relationship between technologies and the texts positioned around them, that there is a relation between what texts say and how technologies operate, and that the explanation is oriented to what the author knows or assumes about how people will read when operating or intending to operate that equipment.

The Structure of Neuroradiology Reports

Documents feature extensively in organisational life, yet are regularly overlooked in analyses of organisations [7][8]. There seems to be a common assumption that documents are just containers of information, containers of text that state some facts or opinions about the world [1][9][10][11]. This is a dismissive perspective, it ignores how documents are tied with practice, how they enable and support the uses they are put to. The problems of this perspective have become acute in information systems development projects that have sought to computerise tasks previously done on paper. When paper is seen as a storage device, the appropriate ways of computerising it can be conceptualised as a "technical problem", a problem of digitising information and of designing the right user interface to access it [9][12]. In areas such as healthcare this has led to enormous problems. The computerisation of medical records was initially thought to be a relatively simple task, but in reality has proved enormously complex and expensive. It was only after early efforts to computerise healthcare ended in failure that the health informatics community realised that not only does design need to account for what forms of information are contained in the record, but also the ways in which it relates to work practices [13].

Below is a neuroradiology report, from a UK hospital (originally discussed in [11]). These reports are produced by neuroradiologists. Neuroradiologists investigate and diagnose suspected or known physical problems to do with the brain, head, neck, spine and nervous system. A common problem they look for is the development of aneurysms, which are balloon-like inflations of blood vessels. These often occur in a circle of arteries, called the Circle of Willis, which supply blood to the brain. Aneurysms can cause severe problems or death if they grow too large and press on the brain, or if they rupture. If a patient has such an aneurysm or is thought to be at risk, images of the relevant section of their brain can be made using Magnetic Resonance Imaging (MRI) or Angiography (MRA). These images are made by specialist radiographers. The neuroradiologists examine these images and dictate a report onto a voice recorder. The report is later transcribed by a specialist secretary.

MRI BRAIN/MRA CIRCLE OF WILLIS

> TECHNIQUE: Axial GRASE brain, 3-D TOF volume images circle of Willis.
>
> FINDINGS: There are aneurysms arising from both intracavernous internal carotid arteries. On the left, the immediate pre-cavernous and intracavernous portions of the internal carotid artery are dysplastic with a fusiform aneurysm. This has a maximum dimension of approximately 1cm. On the right, there is a larger more saccular aneurysm with a maximum dimension of 2cm. This also arises from a dysplastic intracavernous internal carotid artery. The source data images from the angiography and axial GRASE images demonstrate these aneurysms nicely lying within the cavernous sinuses. In addition, there are changes of small vessel cerebrovascular disease in the brain with small lacunar infarcts involving the right gangliocapsular region.
>
> COMMENT: Bilateral intracavernous internal carotid aneurysms. This is fusiform on the left measuring approximately 1cm. On the right, the aneurysm is larger and more saccular in nature measuring 2cm in diameter. Small vessel cerebrovascular disease.
>
> FGH/JKL

The neuroradiology report has a great deal of technical language, but is not simply a 'container of facts'. For a start, consider the part of the findings section that states "the angiography and axial GRASE images demonstrate these aneurysms nicely lying within the cavernous sinuses". Why would an aneurysm ever be described as "nicely lying" anywhere in someone's body? The answer, of course, is that this report is oriented to the medical procedures that are to follow. The report also has a broadly common-sense structure, it begins by describing what was scanned and how, it then discusses the findings, which are then distilled into a more a matter of fact conclusion. This is a common way of ordering text, it is a format can be seen in many contexts beyond healthcare; it is somewhat akin to storytelling [14]. This structure does not reflect precisely the processes of producing the images or making sense of them and diagnosing, but narrates what was done in terms of the purposes for which this text will be read.

Diagnostic work, in healthcare and elsewhere, is very commonly oriented not just to the problem itself but simultaneously to the potential remedial actions [15].

Reports such as these, in order to present coherent and actionable information about a patient, do not fully elucidate the actual, lived work [16][32] involved in their production. This report provides a docile contrast to the activities that produced it, one that both exhibits and conceals what was done during its production. Radiographers scan patient after patient, and the images that are produced are put into folders that pile up ready to be reported by radiologists who report image set after image set. But the report is narrated in a patient centric way: what was done and what was found. The radiographers' efforts of selecting and confirming appropriate imaging techniques, of positioning patients and keeping them still are crucial to making suitable images but are not reportable, and the help given to the radiologist, the letters and other records read, and the journals and books consulted are rarely mentioned. What we have is a matter of fact, economical, narrated statement of what is relevant to the doctors and surgeons who may read this report.

This report, and many others like them, was examined as part of a project looking towards the digitization of neuro-radiology. At first, the ways in which reports were written and used was barely a consideration in this project. The focus was on the digitization of images (the use of computers to store and process images, to ensure they were available where and when they were needed and to automate analysis for diagnostic purposes and population monitoring). Early on, there was a suggestion that in the digitisation of radiology, the report would become redundant. Why not mark up images with data rather than have separate images and reports? Could the information that is in reports be generated automatically and provided as annotations to images? However, having spent several weeks observing work practices in neuroradiology, and working through example images and reports, it became clear to me that the reports were not containers of 'facts' mirroring what was observable in the images. As can be discovered by reading through the report, the computerisation of reports in this way would remove several mechanisms that the producers and users of these reports have for making sense of images with relation to past and future medical procedures on that patient. This is not to deny that integration of image and report could be beneficial, but to point out that it is more than a technical problem (of data integration), but one that must also address corresponding changes to work practices.

Text, Walls and Workflow

Text is often displayed on the walls (as well as the doors, tables, and sometimes the floors and ceilings) of organisations. Signs, posters, whiteboards, lists, charts, instructions and so on are often displayed on walls. This was true of the neuroradiology department, which had various signs to direct patients around, posters for them to read while they waited, and instructions and look-up tables for the radiographers to use when operating the scanners. Slightly more esoteric were special racks, somewhat like magazine racks, fixed to the walls near the MRI scanners and in view of the radiographers as they worked. Receptionists would place in these the folders containing the medical records of patients currently in the waiting room. When a patient arrived, their folder would be retrieved and moved to the rack, thus allowing the radiographers to see at a glance that a patient was ready and waiting. When the patient was brought through to be scanned, their folder would be placed by the scanner so that their notes could be viewed. Once scanning was done the folder was placed by the printer, and once the images were printed (which often took time) the folder was placed in a stack ready for a radiologist to collect. The radiographers would already have a schedule for the day, printed out and placed by the scanner, but patients would not always arrive on time or at all, some would taken less or more time than anticipated, and emergency cases would have to be fitted in. So the schedule would give the radiographers a sense of what 'kind of day' it would be and help them plan when they would take their breaks, but the actual workflow, the moving of patients in and out, the selection of protocols on the scanner, the printing of images and the handing over of the images to the radiologists, was choreographed physically, with text moved around the room in a visible, accountable and recognisable way. Analogous practices can be seen in many workplaces, including print shops [17], airports [18], warehouses [19], and, as I will explain, in software development organisations.

The examples that follow are from a small software development organisation that employed (at the time of the study) five programmers plus three others in managerial and administrative roles. The organisation develops applications to run on mobile phones and other handheld devices. At the time of the study I undertook there, the product in development was an IDE (integrated development environment) for use by other programmers to write code to run on mobile phones. The premise behind this IDE was that it could be used to write code that could be compiled for any mobile phone or mobile device. This would save its users from having to write different code for different phones and devices. Interestingly therefore, their users in this case would be other programmers. The programmers worked in a shared office. Of their four walls, two were used extensively for displaying text (the other two walls had windows). One of the walls had three white-boards; the other had a large notice board (or pin-board) and a large wall-planner. As with the racks in the radiology department, the whiteboards and notice board in this software company formed a component of the way workflow was kept accountable and organised.

*** INSERT FIGURE 1 HERE ***

Figure 1: A task card

The software organisation follows a popular software development process called XP (see [20][21]). As a part of this process, colour coded cards are used (figures 1 & 2). Yellow "story-cards" give a story of some functionality to be implemented by the programmers. Green "task-cards" (figure 1) represent a programming task necessary to implement the functionality described in the story. Several tasks are usually necessary to complete a story. Each card is given a title and usually contains a brief description of what is required. The cards also have some extra information including who wrote it and when, and how long it is estimated it will take to implement. The cards are pinned to a notice board (figure 2). The programmers can take the task-cards one by one, completing one task at a time. This way the board is an ordering device for their work. The tasks can be done one at a time, and can be seen to be progressing. This does not always work smoothly, for example the card slightly out of position in figure 2 has been placed as such because of a problem; in this case the programmer who took it decided it was best implemented by another particular person. The programmers agreed this and the card was put back, deliberately out of position. When a task is finished the card is collected in an enveloped marked "for testing". The cards embody a process [21], they are the process and at the same time are visibly in process. The use of these cards, their reading, is enabled not just by the text but the physical artefacts by which this text can be arranged. Order is created on the board by colour coding, and by the arrangement of cards. This order does not mirror exactly what the programmers are doing, and certainly does not mirror their code, but rather gives a workable representation of where they are and where they are going, including cues for the programmers about the next thing to do, and an indication of how long it should take.

*** INSERT FIGURE 2 HERE ***

Figure 2: The Notice Board

The programmers also use whiteboards. These were used for sketching and discussing issues to do with the software during development, but played a particularly key role at the end of each development cycle when the software would be tested. The task cards could be used for some of the testing, with the tests defined and undertaken on a card-by-card basis, but testing also had to examine the integration of the components of the software and look at it more as a whole. To handle this, a list was written on one or more of the whiteboards. The list worked in a similar way to the task cards in that it is a shared resource between the programmers. The programmers would select something from this list by putting their initials beside it. This enabled the programmers to see not only which tests had been done or were in progress but also who was doing them. Testing in this organisation was admittedly basic. The kinds of improvements we looked at for this organisation (this being one of the main reasons I and a colleague were undertaking a study there) were things like expanding what the testing covered, how tests could be prioritised according to their customers' needs and priorities, and how the testing process could be made more accountable. The possible improvements were mainly procedural, and to a great degree involved consideration of how paper and electronic records could be used and managed. Sometimes this was to add new textual representations, sometimes to change existing ones. So, for example, one question was how could a lasting record of the tests undertaken be generated? In contrast to the task cards, the use of the whiteboard left no lasting record of what was done, the whiteboard would be erased when the tests where completed, and its contents largely forgotten about. Taking a photograph was one candidate solution but this lacked the elegance of the task cards with their ability to both document and order workflow. Use of further task cards and a new notice board was another possible solution. An electronically generated record was another. There is a growing body of literature on testing and XP that has plenty to say about these issues (eg [22]), the essential point to be made here is that improvements to the software development process involve consideration of how texts can be designed and used. The role of texts in software engineering, how accurately they represent the problem and how they are structured and used during implementation has been core to the debates in Software Engineering since the field was born in the late 1960s. Understanding the relation between text and practice in this context, or indeed in neuroradiology departments and so on, is not simply an intellectual curiosity but essential to the debates about how to evaluate and improve practice.

Reading and Writing Computer Programs

In this next example, two of the programmers from the software development organisation have just run tests on a section of code they have written. One of the tests fails, giving an error message highlighting a particular line of code as the point of failure. In the verbal exchange below, one programmer reads the first part of this line out aloud and the other answers with what happens after that line.

Transcript 1

1.	Dan	If node has children
2.	Pat	I just cast to string stringL

Debugging code involves, amongst other things, reading through it. Some of the reading in this example is done quietly, and some out loud. The test fails and an error message is produced. This error message points to a line of code, which Dan (who has control of the mouse and keyboard) goes to. Both programmers have read the output from the failed test and without remark about having read it or what they should do next, they proceed to the line of code where the failure is said to occur. This all involves reading but involves no efforts of coordination as such. Dan does not need to ask Pat's permission to move to the code in question and Pat does not need to ask Dan's reason for doing so. They both know what to do and have a good idea of what the other is doing. The line of code itself is something that Pat has written, Dan's reading aloud of a part of this line of code "If node has children" is taken by Pat as a cue for an explanation. The explanation "I just cast to string stringL" follows the logic of the code, the thing that happens after the code read aloud by Dan. However, while the question involves a reading aloud of almost exactly what the code says, the answer is a more abstract description and contains an evaluation. By qualifying the answer with "I just", Dan appears to be downplaying that section of the code as a possible source of the problem. Even though error messages from tests give the point at which the test failed, the failure can often be the result of something that happened earlier. Therefore, debugging here involves not only understanding sections of code but working out where the relevant section to debug actually is. Programmers working together can often be seen debating, narrating, qualifying and working out where exactly they need to focus their efforts. Reading code is not a mechanical action on the part of the programmers, but is a thoroughly coordinated, thoroughly human activity.

Below is another example conversation between the same two programmers. In this example, they are arguing quite heatedly about how to fulfil a particular task; an older section of code needs to be re-written and they are trying to work out how. Dan and Pat have differing opinions on both how and where in the code, new code is to be written to solve a problem. The whole debate, which went on for some time, involved a variety of activities including various formulations in talk, the use of a diagram sketched on paper, and the reading of various sections of code. The example is made up of a transcript (transcript 2) and a copy of part of the code discussed (figure 3). The programmers have been using a sketch on a piece of scrap paper to aide their debate, but at line 2 Pat tries to draw their focus from this sketch and to the code itself which is displayed on the monitor in front of them.

Transcript 2

1.	Dan	Its not just a matter of changing garbage collect's connection. There are all sorts of thread groups such as=
2.	Pat	=Yeah well there are but we change this ((P turns to the screen)) so when this ((P highlights a line of code)) is working in threads as well.
3.		[3.0]
4.	Dan	Yeah [0.5] Well yeah, but I agree but its not just a case of just checking that.
5.	Pat	No, no no.

In transcript 2[1], line 2, Pat makes two uses of the word "this", each time accompanying it with a gesture. The first "this" orients them at the screen, Pat positions his body and gaze away from Dan and towards the computer. On the second "this", Pat uses the mouse to highlight a part of the code that says getConnection (The code on-screen at this point, including the highlighting, is reproduced in figure 3). This second gestured use of the word "this" orients reading to a specific place on-screen. Pat wants to turn their efforts to focus on a particular section of code. In this collaborative reading (and ultimately writing) of code, what Tolmie has termed premising work [23] can be observed. This is work to bring parts of text to one another's attention for particular purposes.

[1] In transcript 2, the "=" sign on lines 1 and 2 is used to show where Pat cut off something that Dan was saying. The double brackets "(( ))" on line 2 contain a comment about an action taking place. The square brackets "[ ]" on lines 3 and 4 give the duration in seconds of pauses in the talk,

Premising work is directed for the other person to recognise why the text being shared; why the shared text is significant and why it is relevant for solving the task in hand. That it is undertaken without specifically 'spelling everything out' indicates that the programmers proceed (at least initially) assuming that one another will understand why something is being pointed at in this way. As a feature of such premising work, highlighting fine-tunes what is intended, relevant, and significant and effectively pre-motivates the reading, making it apparent, for example, that the intended object, the text, is not so much everything on screen as just a certain entry on it. But can also be seen how co-presence of the programmers is central, since part of the premising resides in the way the sharer physically points at the text shared, whether with a casual waving at the screen, or pouncing on a line and stabbing at it. The sharer can manipulate the ways in which the object becomes accountable. A key feature of premising work is that it is not only directed at sharing an understanding but it is also directed at witnessing that shared understanding. The sharer can see the recipient providing recognition, and the recipient can see the sharer witnessing his recognition, that, in some sense, they are reading this the same way. So highlighting alone does not constitute premising. But premising may rely heavily upon the way in which highlighting is achieved.

*** INSERT FIGURE 3 HERE ***

Figure 3: The Code

Looking closely at the code itself, just as with the neuroradiology report earlier, it can be seen how it is written to be read. For code to compile and run, it must be written using particular syntax and ordered in a logical, computable way. But there is actually a great deal of flexibility in how sections of code can be ordered, combined and separated, named and so on. In computer programming, many aspects of the way code is written rely on convention rather than a strict requirement from the compiler itself. The code in figure 3 follows conventions concerning, for example, the ways in which indentation is done, where the brackets are placed, how sections are grouped together, how variables and methods are named, and so on. All of this is done in order for the code to be readable; the code would still compile if it was all written on one line, or if the methods and variables were called “x” or “y” rather than the more descriptive, more readable, more conventional “m_connection” or “getDataBaseConnection”. Also, the ways in which sections of this code is ordered is down to convention. The way in which JDBCpassword (line 13) is defined after JDBCusername (line 12) is also entirely down to the choice of the programmer, password follows username. Computer programs are certainly more formal than the texts discussed so far, but similar socially organised practices can be found in their reading and writing [24][16][25][26]. Code is written to be read as well as to compile, and reading through code often involves such mundane things as pointing, disagreeing, and explaining.

Designing for Reading

In the next example, again from the software development organisation, Pat, Dan and three of their colleagues argue about the usability of their product. This example is not of how the programmers themselves read, but about what they say about how their users read. As was mentioned earlier, their users are other programmers. Pat, Dan and three others discuss the design of a drop-down menu for the selection of mobile phone emulators. An emulator is a software version of a phone that can be run on a computer. It is quicker to test code using an emulator than to download and run code on actual devices, and hence most users of the IDE first run code on an emulator and on the actual device only when the code is complete. There are a finite number of emulators relevant to the IDE and these are listed in a menu. The problem is that no emulator is installed automatically with the product but each must be downloaded from various websites and installed separately by the user. When using the IDE being developed by the programmers, selecting menu items pointing to emulators that are not installed gives an error message informing the user of this. The developers discuss this:

Transcript 3

1. Sam Users don’t bother reading the error message … they will see it as an error with the IDE itself.

Sam suggests that emulators not installed should be “greyed out” in the menu. Tom disagrees and suggests writing a better error message (which at the moment is fairly cryptic). Dan is visibly annoyed and agrees with Tom.

2. Sam Look, I know its stupid but there you go, the most important thing is that it runs out the box, it is a business decision, I would want it working for the initial play. The first 10 minutes, it should work out the box.
3. Jim Is it hard to make it work out the box?
4. Sam No it’s trivial. We want it to work out the box.
5. Dan All you’re doing is postponing the issues.
6. Sam Good. Until after they’ve bought it.

The conversation goes on about how the users will know how to install additional emulators if the menu item is greyed out. The conversation includes the sequence:

7. Pat If you’re going to install an emulator, you have to read the instructions.
8. Dan Well people don’t read the manual.
9. Sam Well people don’t read error messages either.

Sam goes on to refer to an email from a customer who couldn’t get the demo to work:

10. Sam He wasn’t happy about the trouble to just get hello world running … it’s a matter of impression.

Other possibilities are explored and eventually they decide on having a tick and a cross beside each item in the menu, and an error message that appears when a crossed item is selected. This seems like a good idea because surely

> an error is expected when you click something with a red cross beside it, and therefore people will read this message.

The programmers start with an error message they think no one reads, and end with an error message they believe people who download and try the software will be happy to read. The error message is reworded, but it is not through the quality of the text itself that the programmers believe their users will become potential buyers. It is the sequence that leads to that text, the framing of this text within a sequence of actions. In coming to the decision about using a tick and a cross, a number of quite varied references to users are made. On line 2, Sam refers to "the initial play", talking here with a high degree of generality about what users do. On line 7, Pat talks about using the software in terms of 'you': "if *you're* going to … *you* have to read…". Here again there is a massive degree of generality about what users need to do, where "you" can refer to anyone. On line 8, there is suddenly a reduction in the level of generality: "…*people* don't read…". The word "you" used in line 7 can refer to anyone, including the people in the room. The word "people" in used line 8 more or less refers to anyone, except the programmers in the room. Similar switches in the ways users were referred to were often noticeable when the programmers began to talk about 'bad' or 'undesirable' practices (such as not reading manuals or error messages). This points to a morality surrounding reading practices, there are some practices that the programmers would want to distance themselves from. By line 10, Sam has started referring to a specific person. He is saying something very similar to what he said before at line 2, but this time using a specific example rather than a generality. During the course of an argument, claims have been made at varying levels of generalisation about what it is users do, including claims about how users read. These claims are integral to how a design about a user interface is made and justified. As Sharrock and Anderson [27] have observed, the user appears as a series of typifications at different forms of generalisation, in a way that is scenic to many of the design decisions made. A practical sociology of what users do, including how they read, is constituent to design decision making.

Discussion

Reading and writing is something that is so mundane, so commonplace and routine that we rarely give it much thought. Often, we barely notice exactly what it is we do when we read or write (say, just glancing at the clock or checking an item off a list). Doctors who read medical records do not worry greatly about how to read them but what the record says. Academics do not worry greatly about how to read research papers, but about what they mean for their research. Computer programmers do not worry greatly about how to read code, but about what it does. And so on. This is not to say that they never worry about these things, but that the work of reading is made routine, a background issue, something that does not need to be worried about on a normal basis. On occasion of course there will be worries. A new doctor, a PhD student or a novice programmer may well worry about how to read well. But even they will not notice some of the methods they employ in reading, ones that they have been developing since childhood: how to position yourself and the text, how to read from top-to-bottom and left-to-right, how to scroll a text on a screen without loosing your place, etc. These mundane, practical features of working with text are crucial elements to how work is done. These are the things in which anyone, in more or less any workplace, has to become competent. The things they need to know so well that they can be done without noticeable effort or remark.

Reading, as Livingston [16] puts it, is in "the reader's implementation of society's ways of reading" (p16). The practices of reading, the means and methods by which people read and what they achieve by reading is a thoroughly social achievement. This is not to say that reading is always done socially, but to say that the methods, the competencies and expectations about how reading will be done are resultant from our relationships with those around us. We have a great many expectations about how people will read, something that is true when we write something, or, as with the programmers, when we situate text within a design, but also something that is true when we work with people reading from a shared text. Our expectations are not necessarily always correct. but when they are, the practicalities of reading become a background issue, something of little trouble to those doing it. Texts are written for reading, medical reports present what the reader needs to know in a consistent and logical format. Code is structured, written and ordered in ways that make it more comprehensible to humans. Indeed the professionalism of code or of medical reports, what is good code or a trustworthy report, is contingent upon them not just being technically correct but upon them being conventional. Text is also not just arranged on the page or on whatever it is written, but it is arranged around workspaces. Physical documents, be they medical records or programming task-cards are moved around workplaces in orderly, representative and accountable ways. Text is something that is best viewed in action, as a pair with reading [1][16].

The examples in this chapter have all related to design, the design of technology or of work practices. In different ways, consideration of these has involved consideration of mundane practices of reading. A badly designed door is supplemented with a colourful, textual explanation. Questions about the computerisation of neuroradiology reports lead to questions about how exactly these texts are used. Questions about improving software testing lead to consideration of how documents are used and moved around an office. These are the kinds of issues ethnomethdological investigations can cast light on, but other disciplines downplay. Cognitive approaches, for example, generally downplay practice. Consider the following line from a paper about reading and writing computer programs.

"[I will assume] that programs are written by translating cognitive structures into code [and] that part of the process of comprehending a program is parsing it back into the original cognitive components" [28, p25].

This is not a finding from a paper, but an assumption on which the findings rest. The great majority of studies of programming assume that the problems of programming lie in the programmer's head. It is as if there is a program in the programmer's mind that is like the program on the screen, and the more competent the programmer or the more understandable the program, the more similar the two will be. From this perspective the programmer's ways of programming, their ways of reading and writing, of gesturing, of moving objects around, of talking to their colleagues, of shouting and laughing are just not relevant. In the quote above, writing is re-specified as "translation… of structures" and reading becomes "parsing it back". These are mechanistic metaphors that are of no benefit to an analysis other than to dismiss human practice. This is not to claim that cognitive science does not offer insight into design (The work by Green mentioned above, demonstrably has been useful to the design of programming languages), and this is certainly not to enter the grander debates and battles between ethnomethodlogy and cognitive science. Ethnomethodology is simply shown in this chapter to offer an alternative perspective, one that is also demonstrably useful [2][29]. The problem of ethnomethodology as a line of research is that it can often be combative and arcane. There is a growing body of ethnomethodlogical studies of practices related to reading (for example, how reports are written and read by the police [30] or by medical workers [31], how flight strips are used by air traffic controllers [33], how mammograms are read by radiologists [4] how contracts feature in information systems development [34], how job tickets are used in print-rooms [17], how whiteboards [18] or blackboards [19] feature in workflow, how documents are edited [35] and read [36], how timesheets are kept [37] or how computer code [24] and mathematical proofs [38] are worked through) all of which are extremely helpful, but the major texts tend to be quite arcane. The thrust of most ethnomethodological analyses of text and reading has been to criticise other sociologists. Ethnomethodologists such as Watson [1], Garfinkel [39], Livingson [16], Lynch [40] and McHoul [41] explain how other sociologists, particularly those doing conversation analysis, rely heavily upon the practices of text and reading to achieve some study and yet systematically ignore these.

Acknowledgements

The examples from the software development organisation arise from collaborative research I undertook with David Martin and Mark Rouncefield while at Lancaster University. The examples from the neuroradiology department come from my PhD research, which was done under the supervision of Stephen Kay at Manchester University.

References

[1] Watson R (2009) Analysing Practical and Professional Texts: A Naturalistic Approach. Farnham, UK: Ashgate.

[2] Button G, & Sharrock W (2010) Studies of Work and the Workplace in HCI: Concepts and Techniques. San Rafael, CA: Morgan & Claypool.

[3] Norman D (1988) The Psychology of Everyday Things. Basic Books, New York.

[4] Hartswood M, Proctor R, Rouncefiled M, Slack R, Soutter J, & Voss A (2003) "Repairing" the Machine: A Case Study of the Evaluation of Computer-Aided Detection Tools in Breast Screening. Proceedings of the Eighth European Conference on Computer Supported Cooperative Work, 14-18 September 2003, Helsinki Finland. 375-394.

[5] Arminen I, Poikus P (2009) Diagnostic Reasoning in the Use of a Travel Management System. Computer Supported Cooperative Work, 18 (2-3). 251-276.

[6] Grint K, & Woolgar S (1997) The Machine at Work. Technology, Work and Organisation. Cambridge, UK: Polity Press.

[7] Slack R (2010) this book

[8] Harper R (1997) Inside the IMF: An Ethnography of Documents, Technology and Organisational Action. Orlando Fl, Academic Press.

[9] Berg M, & Bowker G (1997) The Multiple Bodies of the Medical Record: Towards a Sociology of an Artefact. The Sociological Quarterly, 38(3), 511-535.

[10] Prince K (1996) Boring records? Communication, Speech and Writing in Social Work. London: Jessica Kingsley Publishers.

[11] Rooksby J, Kay S (2003) Patient Reports as Stories of Clinical Work: Narrative and Work in Neuro-Radiology. Methods of Information in Medicine, 42(4), 445-450.

[12] Heath C, & Luff P (1996) Documents and Professional Practice: "Bad" Organizational Reasons for "Good" Clinical Records. Proceedings of Computer Supported Cooperative Work (CSCW 1996), 354–363.

[13] Berg M (1997) Rationalising Medical Work: Decision-Support Techniques and Medical Practices. Cambridge, MA: MIT Press.

[14] Kay S & Purves I (1998) Medical Records and the "Story Stuff": A Narratavistic Model. In Greenhalgh T, Hurwitz B (Eds.) Narrative Based

Medicine: Dialogue and Discourse in Clinical Practice. London: BMJ Books. 185-201.

[15] Büscher M, O'Neill J, & Rooksby J (2009). Designing for Diagnosing: introduction to the special issue on diagnostic work. Computer Supported Cooperative Work, 18(2–3), 109–128.

[16] Livingston E (1995) An Anthropology of Reading. Indiana University Press, Bloomington.

[17] Bowers J, Button G, & Sharrock W (1995) Workflow from Within and Without: Technology and Cooperative Work on the Print Industry Shopfloor. Proceedings of the fourth European Conference on Computer Supported Cooperative Work (ECSCW), Stockholm, Sweeden. 51-66.

[18] Suchman L (1997) Centres of Coordination: A Case Study and Some Themes. In Resnick LB, Säljö R, Pontecorvo C, & Burge B (Eds.) Discourse, Tools, and Reasoning: Essays on Situated Cognition. Berlin: Springer-Verlag. 41-62.

[19] Kawatoko Y (1999) Space, Time and Documents in a Refrigerated Warehouse. Human Studies, 22(2-4), 315-337.

[20] Beck K (2000) Extreme Programming Explained, Embrace Change. Boston, Addison Wesley.

[21] Mackenzie A, Monk S (2004) From Cards to Code: How Extreme Programming Re-Embodies Programming as a Collective Practice Computer Supported Cooperative Work, 13(1), 91-117.

[22] Crispin L, & House T (2003) Testing Extreme Programming. New Jersey, Addison-Wesley.

[23] Tolmie P, Hughes JA, O'Brien J, & Rouncefield M (1999) The Affordances of Paper and Co-Presence: Some Interactional Phenomena of Organizational Life, Unpublished paper, Department of Sociology, Lancaster University.

[24] Button G, & Sharrock W (1995) The Mundane Work of Writing and Reading Computer Programs. In Have, P ten, & Psathas G. (eds.) Situated Order. Studies in the Social Organization of Talk and Embodied Activities. Boston, University Press of America.

[25] Brown B (2006) The Next Line: Understanding Programmers' Work. TeamEthno-online 2, 25-33.

[26] Reeves S (2006) The Code Document's Structure and Analysis. TeamEthno-online 2, 34-51.

[27] Sharrock W, & Anderson B (1994) The User as a Scenic Feature of Design Space. Design Studies, 15(1), 5-18.

[28] Green TRG, (2000) Instructions and Descriptions: Some Cognitive Aspects of Programming and Similar Activities. Proceedings of AVI2000, ACM Press. 21-28.

[29] Sommerville I, Rodden T, Sawyer P, Bentley R (1992) Sociologists can be Surprisingly Useful in Interactive Systems Design. People and Computers VIII: Proceedings of HCI'92. 341-353.

[30] Ackroyd S, Harper R, Hughes JA, Shapiro D, & Soothill, K (1992), New Technology and Practical Police Work. Milton Keynes, UK: Open University Press.

[31] Cicourel A (1981) Language and Medicine. In C Fergusson, S Heath (eds) Language in the USA. Cambridge: Cambridge University Press: 320-334.

[32] Berg M (1996) Practices of Reading and Writing. The Constitutive Role of the Patient Record in Medical Work. Sociology of Health and Illness, 18, 499-524.

[33] Hughes JA, Randall D, Shapiro D (1992) Faltering from Ethnography to Design. Proceedings of Computer Supported Cooperative Work, Toronto, Canada: 115-122.

[34] Martin D, Procter R, Mariani J, Rouncefield M (2007) Working the Contract. Conference of the Computer-Human Interaction Special Interest Group (CHISIG) of Australia on Computer-Human Interaction: Design: Activities, Artifacts and Environments, Adelaide, December, 2007.

[35] Heap J (1992) Normative Order in Collaborative Computer Editing. In Watson G, & Seiler R (Eds.) Text in Context: Contributions to Ethnomethodology. London, Sage. 123-137.

[36] Laurier E (2004) Doing Office Work on the Motorway. Theory, Culture & Society, 21(4-5), 261-277.

[37] Brown B (2001) Unpacking a Timesheet: Formalisation and Representation. Computer Supported Cooperative Work, 10 (3-4), 293-315.

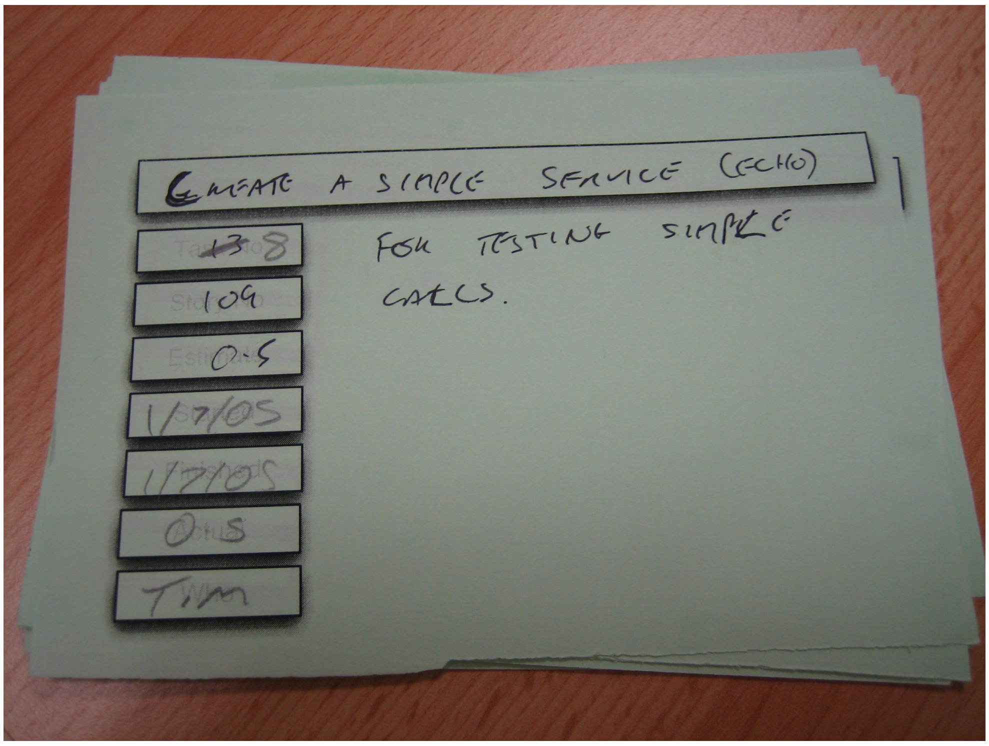
[38] Livingston, E. The Ethnomethodological Foundations of Mathematics. London: Routledge, 1986.

[39] Garfinkel H. (1967) Studies in Ethnomethodology. Englewood Cliffs, Prentice-Hall.

[40] Lynch M (1993) Scientific Practice and Ordinary Action. Ethnomethodology and Social Studies of Science. Cambridge, Cambridge University Press.

[41] Mchoul A (1982) Telling How Texts Talk: Essays on Reading and Ethnomethodology . London: Routledge & Kegan Paul.

Figure 1:



CREATE A SIMPLE SERVICE (ECHO)

Task No	~~13~~ 8
Story No	109
Estimate	0.5
Started	1/7/05
Finished	1/7/05
Actual	0.5
Who	TIM

FOR TESTING SIMPLE CALLS.

Figure 2:

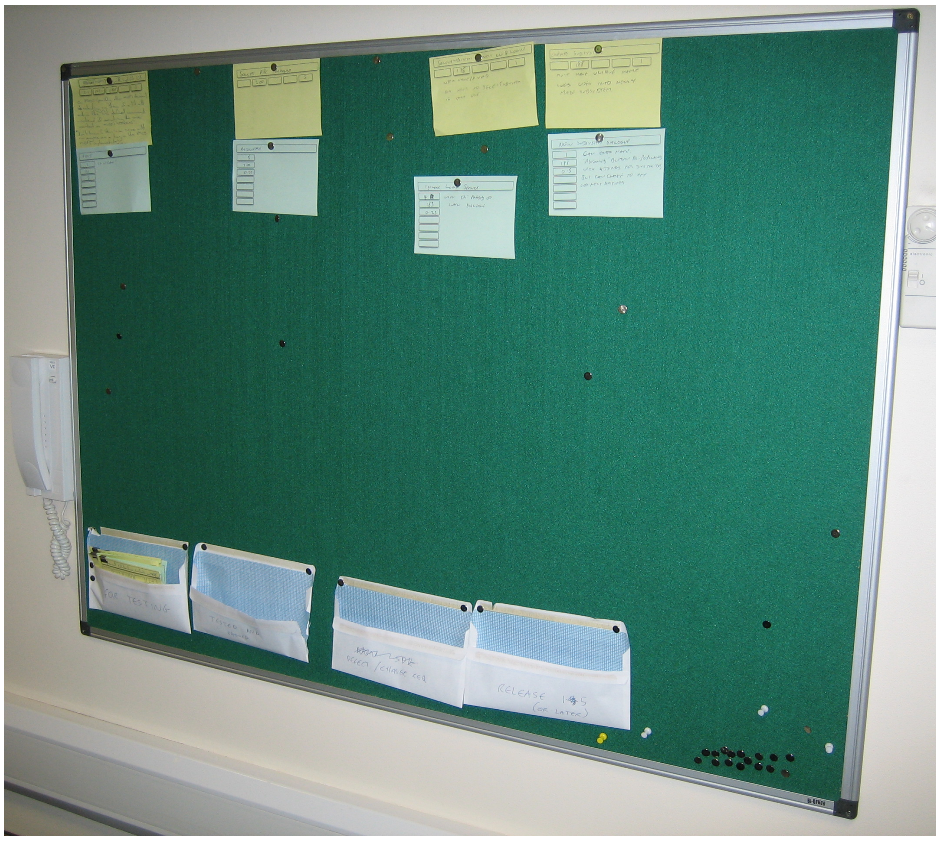
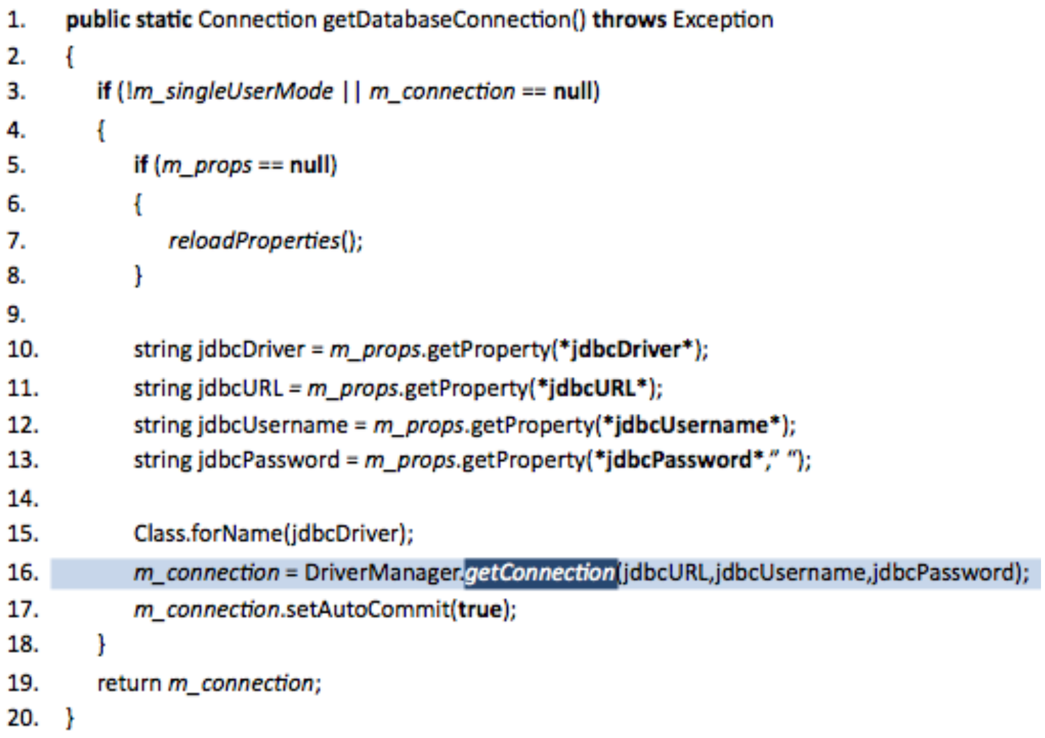


Figure 3:



```
public static Connection getDatabaseConnection() throws Exception
{
    if (!m_singleUserMode || m_connection == null)
    {
        if (m_props == null)
        {
            reloadProperties();
        }

        string jdbcDriver = m_props.getProperty(*jdbcDriver*);
        string jdbcURL = m_props.getProperty(*jdbcURL*);
        string jdbcUsername = m_props.getProperty(*jdbcUsername*);
        string jdbcPassword = m_props.getProperty(*jdbcPassword*,” “);

        Class.forName(jdbcDriver);
        m_connection = DriverManager.getConnection(jdbcURL,jdbcUsername,jdbcPassword);
        m_connection.setAutoCommit(true);
    }
    return m_connection;
}
```