Working paper submitted for 'Ethnographies of diagnostic work' 17-18 April, Lancaster University, UK.

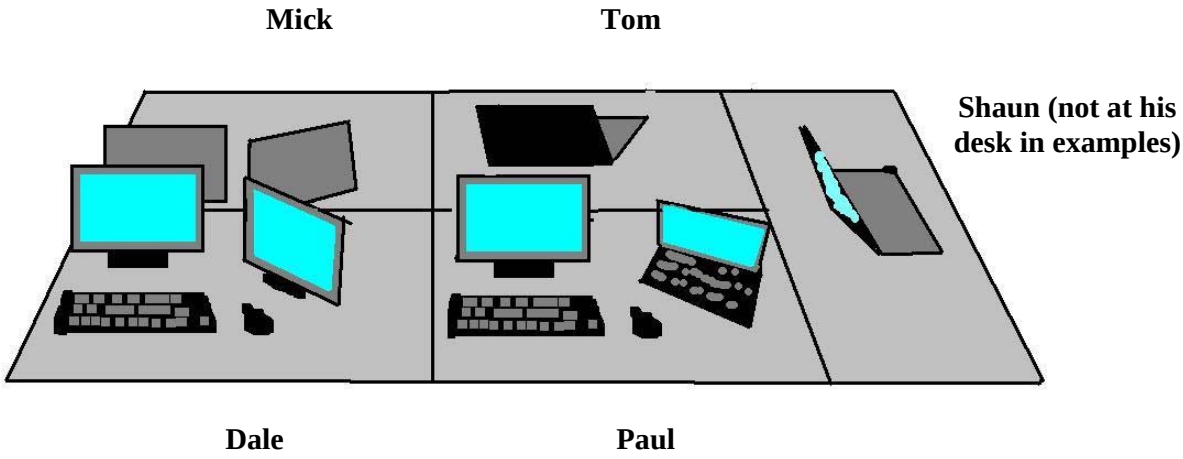
Draft : Please do not circulate beyond "Ethnographies of Diagnostic Work"

Debugging Software

John Rooksby

This talk will present data from an ethnography of software developers in a small software company developing a software product for business customers. The study employed observational methods and in-situ interviews to view, capture and understand work as it happened via note taking, video, photographic and audio recordings. A total of 30 days fieldwork was undertaken in a period between July 2005 and April 2006.

The company 'IDEco' (a pseudonym) produces an IDE (integrated development environment) for end-users to develop applications to run on various mobile phones and other mobile devices. During the study, the company had seven full-time employees, four of whom were programmers (Paul, Tom, Dale and Mick – all names have been changed). There was also a technical director and trainer (Shaun), a customer relationship manager - CRM (Gordon), and a financial administrator (Brian). The programmers work sitting around a large shared table (as shown below).



There will be (time permitting) two examples given in my talk. The first will consist of four video clips, the second is from hand-written fieldnotes.

Example 1

P has written software to allow a mobile phone to communicate with (send a request to, and get a response from) a webservice. The software does not work, and P has been trying to find out for some time why it does not work. This involves finding out what, if anything, the software is actually doing.

Clip one

Paul is trying to write code that will monitor the communication between his software running on a phone and a remote webservice run by 'SalesForce'. Mick has been instructing Paul on where to find some code in their code base that does a similar thing. Mick has been explaining what Paul will need to do to this code. Dale dissents about Mick's descriptions of what Paul needs to do

Dale *inaudiable*
Mick I don't think that matters [
Dale [*inaudible*
Mick [because you're only hitting the server ( ) once, and then you you're just putting something in between it aren't you ( ) that this [
Dale [*inaudible*
Paul *huff* Right, so what you mean how do I get the response?
Dale Unn
Paul If I open a socket
Dale Yeah
Paul [why
Tom [You don't need to
Paul what
Tom It would be the same socket
Paul It just comes back to you?
Tom Yeah
(3.20)
Tom You're goin to have to keep, gonna have to keep the socket open from the phone
Paul Yeah, think I've got it
Tom And, one to salesforce 7
Paul Yeah
Mick That will come back down that one and then you can throw it back down there
Paul Yeah
Mick And that would have kept it open to the phone I would have thought but
Dale Well you [
Mick [Its worth experimenting
Dale [I know I know but I thought there was a bit more to HTTP then that (0.5) is there not?
Paul HTTP is just a protocol, HTTP * just defines the stings and [
Mick [As far as I know its just a stream of bytes in the end innit, it goes one way and the other and I don't think it (0.5) I don't think theres uh you know theres no two way, exchange while that's happening, if one goes one way and another that responds comes …

At the end of this clip P is pasting the code into his IDE and is beginning work on it.

Clip two (around 7 minutes after clip one)

Paul Oh this is going to be such a pain in the fucking arse

Clip three (around 15 minutes after clip two)

Paul is discussing his code with Dale. Paul has discovered that there is a blank line in the request sent by the phone and surmises that this is causing the software to fail. They fix on a particular bit of code.

Paul It might be worth doing a diff on this I might have just copy and pasted that in as part of me
Dale inaudible
Paul yeah
Inaudible
Paul As part of my thing to figure out whats different
Dale Right so you can just * midP from *** there
Paul Yeah
Dale This can be a problem with people trying to figure out whats going on can introduce
Paul Yeah
Dale and that become a real nightmare donnit
Paul Yeah
Dale You don't even think to

Runs the 'diff'

Dale Right
Paul So I've put that here as part of my effort to figure out whats going on and in so doing I've err wasted about a week
Dale No no well you haven't you've found …
Paul Yeah

Dale MidP 1 was it
Paul This is probably why I get different things occurring in different things all the time
Dale Unn *inaudible*

Dale because it might be consistent behaviour
Paul Yeah it might at least be consistent

Clip 4 (Around 2 minutes after clip three)

Paul Its doing the second request
Paul Oh my God!
Dale It working?
Paul Yeah
Fin? Horray
Paul That's MidP 2, I haven't changed anything for MidP 1.
Dale Right
Paul So that's not going to help
Dale No unless
Paul But I suspect that might be, coughs, like Tim says, HTTPS is optionally implemented in MidP 1and it succeeds on the ** which is HTTP and then the second URL you get cause the URL for the second request is part of the response to the first
Dale Right
Paul And that's an HTTPS
Dale Right, yes
Paul And it may well be that its because of that

Example 2

In this example we describe the discovery of a problem with the software. This problem is eventually categorized as a 'known problem' rather than solved.

Dale is showing Paul 'virtual machine' (VM) software running on his computer. Paul is going to install this software on his computer. Dale's computer stops responding to his use of the mouse.

Paul "Your machine's just died!"
Mick *Entering conversation from across the table* "What you done?"
Dale "I've dragged a control …" …
Paul "You know what it is, you got a massive drag threshold."
Dale "No, I got it set to one."
Mick *Goes round the table to stand beside D.* "It's like the delay that's set in the threshold"

There is a short period of silence.

Mick "I mean its slow all round and I don't get that. It's on a machine that's faster than mine. The only time, is when you've got many selections and you do a validate on drag.

Paul is at his machine continuing to install a VM.

Mick "Those items seem sluggish as well"

silence

Mick "Weird"

silence

Dale "That's weird"
Mick "That is executing exactly the same bit of code
Paul "Its weird (p) it is on your native machine?
Dale "Yeah
…
Dale "It might be something to do with the VM. The only thing I can think of is the way the VM passes invalid events to the valid system.
…
Dale "no but I do think its something to do with the JVM in this VM.
Mick "a lot of redraw events of something. Which it doesn't need to (Pause). But that would explain as well why that cursor key (Pause) that's pooh though cos there's bound to be people out there using VMs.
Paul "nuh!" (A running joke this day has surrounded the fact Paul has never used a VM)
Dale "They do, but mind you not so much a client. Its on servers.
…

Mick "It might not necessarily be Java though."

Dale "Yeah I think its Java."

Mick "The JVMs running inside the VM?"

Dale "Yeah"

Mick "I suppose its something we can document as a known problem."

Paul "Yeah"

Mick "If you're running your JVM on a VM, or we could say…"

silence

Mick "It could be where [unnecessary events keep firing] but normally in Windows it would lose them so it doesn't do anything. But in the VM its firing all the time. We could change that code. But I don't know how. Its not like we can intercept the events in Java. It won't let you."

Dale "I'll Google it, see if anyone else has noticed."

A short while later Paul finishes installing a VM on his machine. He confirms that the same problem occurs. Mick speculates that it is an issue with Swing (the Java graphical user interface utilities package). This would probably imply that the issue is out of their hands.

Phil dragging stuff around on his screen – probably trying to do what D was earlier

Phil "I reckon its something to do with the events

…

Dale "Its something to do with events

Phil "cos if you click on it and hold it… but if you click on it and do it immediately it…

…

Paul "Just meaning that it takes ages to re-draw (resizing GUI window) Bizarre, its bizarre. That seems to redraw really quick. But if you move that. Uhh. Strange

silence

Dale "Yes I don't know. It might be a issue with SWING.

Later they further evaluate the issue. They talk about "Joes off the street" and whether such people would use VMs. The consensus is that ordinary people wouldn't have a VM. They then talk about what they themselves do with a VM:

Paul "But you would never develop in a VM. Our stuff doesn't work well in a VM, but you wouldn't develop in a VM. And our guys are developers."

Dale "Well supposedly."

Paul (laughing) "Not really from what we've seen…"

Dale "Too harsh!" …

Mick "Its good that you're using that and that we've found it. If we got a call coming in we could say "Are you using it on a VM?" and they would say "oh yeah!". It would be interesting to see how many we got of that nature."

They decide to categorise this as a ‘known problem’, and it will be listed in the user manual as such.

Discussion

In example one, we see how long standing problems and corresponding solutions develop. In particular we see how prior efforts to solve the problem by writing lines of code have been forgotten and become problems themselves. We see that a ‘solution’ does not wrap up the problem but sets directions for further work on the software, and we see the significance and meaning of this solution discussed.

Example one begins with a clip of the programmers engaging in discussion of how a particular technology relevant to solving a problem works. There is dissent as to how exactly this technology works, but as Mick exclaims “its worth experimenting”: it is worth getting on with this bit of code and doing some programming; what the outcome may be is not exactly known, but the outcome ought to be helpful. The strategy does in-fact turn out to be helpful and Phil discovers there is a blank line in the request. From this discovery he is able to single out some ‘suspect’ code.

Phil discovers the problem was caused by code written as part of an effort to find a cause to earlier manifestations of that same problem. This is discussed as an issue in the practices they engage in when debugging.

In example two we see how an error can arise, and in both we see the strategies the developers have in dealing with them.

In example two a problem is ‘discovered’ and, when attempts to solve it get complex the relevancy of solving it gets discussed. In this instance the initial ‘incorrect’ proffered diagnoses and investigations are not simply a product of impatience and inexperience. These ‘recipes’ (Schutz 1964) are functional and often lead to quick results. They begin with appeals to organisational knowledge (“has this happened before?” “What does this remind you of?”) and on this failing they switch to first principles, particularly a comparison of two machines running side by side. They never identify the cause of problem but are satisfied it is out of their hands and go on to consider whether this problem is a problem for them to worry about. By categorizing it as a ‘known problem’, the problem fits into the bureaucratic system and can be put off or dealt with within orderly and normal work (i.e. it *could* become a requirement for solution in a later iteration).

Example two demonstrates some of the ways in which a concern is settled, particularly through realisation of ‘the user’.

As Boden argues, problem solving is located in fine-grained, sequential organisational activities. Of particular relevance is the notion of 'local logics':

> "As they sift through locally relevant possibilities ... social actors use their own agendas and understandings to produce ‘answers’ that are then fitted to ‘questions’." (Boden 1994)

In the examples we document the contingencies of product development, the 'normal, natural' troubles whose 'usual' solution is, of a sort, ‘readily available’. Usual solutions invoke horizons of tractability, containing candidate answers (seen before) and solutions (used-before-and-seen-to-work). Problems demand quick solutions, taking into consideration the present situation, the resources available, as well as any consequences:

> "Caught in the pressing necessity of choice, organisational actors move through a fluid mix of problem identification, goal negotiation, solution seeking and decision-making." (Boden 1994)

Such situated problem-solving results in fixes that may eventually become part of the repertoire of candidate solutions. And, as the extracts suggest, the boundaries between the types of problem are permeable and resolvable. Similarly, and unsurprisingly, different members view problems differently and this may lead to the resolution of the problem in, and through, the ability to improvise or recognise similarities with previous problems.