The Benefits of a Long Engagement: From Contextual Design to The Co-realisation of Work Affording Artefacts

Mark Hartswood, Rob Procter,
Roger Slack, James Soutter, Alex Voß
Division of Informatics
University of Edinburgh
2 Buccleuch Place, Edinburgh
mjh|rnp|rslack|jsoutter|av@cogsci.ed.ac.uk

Mark Rouncefield
Department of Computing
University of Lancaster
Lancaster, LA1 4YR
m.rouncefield@lancs.ac.uk

ABSTRACT
This paper critically examines the contextual design methodology advanced by Holtzblatt and Beyer. We argue that contextual design provides 'thin description' compared with the 'thick description' of ethnomethodologically informed ethnographies and that this impoverishes its claims to perspicuous description. As a way of addressing the limitations of contextual design, we propose co-realisation, a methodology that requires a long engagement: i.e. a longitudinal commitment from designers to building a shared practice with users. The paper concludes with two case studies of doing co-realisation.

Keywords
Contextual design, co-realisation, ethnography

INTRODUCTION
The question of how to incorporate ethnomethodologically-informed analyses of work practice into IT system design and implementation processes has been the subject of considerable debate. One approach that has been widely advocated is contextual design [1]. We argue, however, that contextual design stops short of applying the full implications of ethnomethodology for IT systems design. In keeping with Button's suggestion "... that ethnography can be trailed into the world of design in a harder fashion than our enthusiasm currently permits." ([3] p. 330), we have been exploring the potential of 'co-realisation', a synthesis of participatory design and ethnomethodology.

In this paper, we critically examine contextual design's methodological claims and contrast them with those of co-realisation. We conclude with brief illustrations of co-realisation in practice taken from two case studies.

CONTEXTUAL DESIGN
Before moving on to a critique of contextual design, it is important to explicate its central contours. Holtzblatt defines contextual design as: "A set of techniques to be used in a customer centred design process with design teams. It is also a set of practices that help people engage in creative and productive design thinking with customer data and it helps them co-operate and design together." (Holtzblatt quoted in [6], p. 313). She lists the steps of contextual design as follows: contextual inquiry – talking to people as they do their work; interpretation and modelling with cross-functional teams; consolidation of information gained through previous steps; visioning about work practices and the development of storyboards; user environment design – using storyboards to develop 'a software floor plan (that drives) the user interface design'.

From Holtzblatt's comments, it would appear that contextual design developed out of a concern with usability and the work of participatory designers such as Kyng and Ehn. It would also appear that one of the motivations was dissatisfaction with "all this qualitative stuff" ([6]) in terms of how it came to be sidelined. It would seem that the method attempts to blend qualitative approaches (fieldwork) with the process vocabularies of more traditional software engineering methodologies.

Contextual design does do some things correctly: most important of these is the stress on understanding context in systems design. Thus, contextual design is an improvement over traditional 'over the wall', context ignorant methodologies and we would applaud the need to take users into account in the building of any IT system. Keeping the eyes of designers focused on the context is important, but in one sense, contextual design is at the mercy of its own process models in that they reify the world in terms of data to be used *for design*. Such models seem to us a means of terminating an already all too brief 'engagement with users in favour of some trans-situational 'ontology'. This is not the only drawback, however, of contextual design's lack of interest in a 'long engagement' with users. A new system may change the work and so raise new requirements. In such circumstances, IT systems and the work practices they are intended to support need to co-evolve, but contextual design never gets to grips with the 'lived' reality of being a user of the new system. Yet, we argue, it is precisely this, as IT systems and artefacts penetrate more deeply into organisations and work settings, that IT design and development methodologies must strive to achieve. This is what co-realisation sets out to do.

CO-REALISATION

Co-realisation calls for creating a shared practice between users and IT professionals that is grounded in the lived experience of users and a commitment to 'stick around and see what happens' once a new IT system or artefact is deployed [4,8]. Co-realisation's goal is to achieve a situation where users and IT professionals can spontaneously shift the focus of their attention between the different phases of the system/artefact lifecycle, even to the extent that these cease to exist as separable activities. It seeks to bring about a context for IT design and implementation work where, as Büscher, Mogensen and Shapiro [2] have put it, "… effort shifts fairly smoothly between implementing or adjusting previously decided possibilities, picking up on the host of small problems that arise during work, coping with the unanticipated consequences of previous actions, talking to individuals …"

Co-realisation aims to enable users to grow into technology: it is minimally invasive, preserving the advantages of technology for work life while refraining from engaging in gratuitous technological interventions or dubiously predicated work redesign efforts. Crucial to this is membership which can only be afforded to IT professionals by their 'being there' in the workplace along with users. The term 'membership' here means having the competencies to know 'how things go on around here' and how work gets done. It does not call for IT professionals to 'learn the trade' so as to be able to do it.

Through creating a shared practice, co-realisation seeks to capitalise on user-led processes of 'design-in-use'. "This requires crossing boundaries both within technology production and between technology production and use ... If technologies are to be made useful, practitioners of other forms of work must take up the work of design ... that is appropriating the technology so as to incorporate it into an existing material environment and set of practices." [7] Given the right choice of technologies, co-realisation can assume the characteristics of 'bricolage' – i.e., the rapid assembly and configuration of 'bits and pieces' of software and hardware [2] – led by users acting within their own workplaces.

Co-realisation emphasises tightly coupled, 'lightweight' design, development and evaluation techniques that can be easily and rapidly customised to create new systems and artefacts for evaluation in use. Co-realisation therefore has synergies and interesting parallels with agile software development methods -- such as 'just-in-time' requirements analysis (see, e.g, [5]) -- that have recently emerged as radical alternatives to conventional software engineering processes. For example, both co-realisation and agile methods stress that the functionality delivered should only be what is needed, and that functionality should accrete and track work practices.

The essential practical step for co-realisation is how to organise 'being there' through taking the technical work of design and development into the user's workplace. We have been exploring the realities of doing co-realisation in case studies set in two quite different work settings [4,8].

THE TOXICOLOGY WARD

We have been pursuing co-realisation in the busy toxicology ward of a large hospital [4]. The toxicology ward provides a specialised inpatient service that allows for joint medical and psychiatric assessment of patients following a suspected self-harm incident, for example, a drug overdose. One of the toxicology ward's functions is to determine the need for further psychiatric and social care, referring patients on as appropriate to other services. This work is carried out by the psychiatric assessment team.

The project began with a six month period of familiarisation with work practices through fieldwork. The aim was to build relationships and understanding, an essential predicate for taking on the role of 'IT facilitator'[1]. In the manner of more conventional participatory design projects, design work then began with a series of group meetings, supplemented when their schedules allowed, by meetings with individuals. These centred on the discussion of a series of potential IT applications, including a resource database of information about services and contact details of other professionals involved in patients' care, the use of speech recognition, and a minimal electronic patient record system that could be used to recall basic information about previous attendance.

Use of speech recognition was seen as a candidate solution to a particular sort of problem, namely that of 'improving communication' with general practitioners and other professionals who subsequently take charge of the patient's care. At face value, the chosen speech recognition system appeared to offer a good solution to this problem, suggesting in its specifications that it is "faster than typing" and that "dictating at 140 – 160 words per minute, a person produces a three page document in less than six minutes". The idea of a speech recognition system implicitly promises the possibility of speech transformed effortlessly into text on the screen (which can be far from the case) and the possibility of improved service provision as a consequence. Accuracy and direct input are seen as the main features of the system together with the notion that it would take pressure off overworked secretaries and improve the accuracy of the letters sent out by the clinic. Thus there are sound reasons why the system might be introduced in the clinic, which originate with the psychiatric assessment team, and the role of the IT facilitator is to make the system work.

Despite an initial conception of a computer system as a congenial means of producing letters, the speech

[1] We have adopted this term for the situated IT designer/developer since it stresses the many and varied roles implied by 'being there'.

recognition system is actually prone to the sorts of errors ascribed to human transcribers. So, while the promise of a technology may be clear, its use in practice may occasion risks and hazards for those who will make use of it: time may be lost when trying to produce a letter using speech recognition, work may be lost if there are problems with the system itself, the delivery of patient letters may be made error prone in specific sorts of ways. There are also various sorts of commitments that have to be made: to actually sit down and grapple with the system during what are often busy working days for the assessment team, to undergo the periods of training required to become familiar with the system and to enable the system to recognise their voice. These commitments are set against the risk that the 'experiment' itself may come to nothing; using the system may in the end not turn out to be a viable means of producing letters in the fashion hoped for.

Development was characterised by the IT facilitator doing development work in the workplace during times when the assessment team would be undertaking assessments and using the technologies that were under development. Thus the IT facilitator was in a position to see the various IT 'solutions' in use in and as part of team member's routine activity. In this way the IT facilitator was at hand to advise about use of the system, to troubleshoot problems, to discuss possible enhancements and to have an eye for how components of the system are actually used in practice as and when such opportunities arise. Furthermore, by making 'being there' a priority in this way the facilitator was able to demonstrate a commitment to producing technologies that are in 'working order', that is, technologies that afford the work of the psychiatric assessment team.

The team encounter a number of problems in the use of the system, the main one being the inaccuracy of the recognition, this leads to a disjuncture between the anticipated benefits of the system and the system in use, so much so that it becomes a point to celebrate when one member gets a whole sentence correct: "that worked – unbelievable – I managed a whole sentence". The inordinate amount of time spent using the system means that it is a technology that is found to be difficult to use, and often without any immediate benefits – indeed users can often be seen to 'struggle' or 'grapple' with the software. With use more serious faults emerged: the system has a propensity to crash and a number of work arounds are introduced in order that the work of dictating letters might continue. The presence of the facilitator is also important, as the facilitator acts as a guide and trouble-shooter, as well as a repository of collective memory for the 'proper' procedure and for workarounds.

Over time, routine use of the speech recognition system dropped off. Members cite the cumbersome nature of the system and the inaccuracies as central to its abandonment in favour of typing letters. Our method, however, treats this as an opportunity as opposed to a risk and the agility of co-realisation means that such switches can occur. The switch to typing is not so much a risk to be managed but a practical, situated, members' choice of a work-affording artefact.

ENGINECO

Work in the control room of EngineCo's manufacturing plant (producing diesel engines) involves various tasks like monitoring the production process, adjusting parameters, translating between the production process and the work of various other professionals (e.g., quality control), and being involved in continuous re-organisation and optimisation activities that are required to constantly match the plant's working to outside requirements [8]. Because of this mix of tasks, some of which require constant attention, there are few opportunities for control room workers to participate in systems development activities that are shaped along the more traditional lines of project work. Although the social relations in this setting are actually quite favourable in that IT professionals are located on-site and communicate with control room workers on a regular basis, most of the development activities take place outside the control room and workers do not play a role in them. The traditional break-off point between requirements analysis and design with all its attendant problems (e.g., a lack of responsiveness) is maintained.

The IT facilitator's activities in this setting aim at making the development activities visible to and accessible for the workers and involving them in the development activities as much as is feasible. The facilitator involved in this project maintains a sustained presence in the control room (currently about four days a week) and works on a number of systems that are used in control room work, most prominently an electronic shift book application. Work on IT systems in this project is occasioned by the everyday activities in the control room. In one instance, the facilitator observed a worker's use of the Internet Explorer to browse XML-formatted log files that are generated by a particular system. Since there was no mechanism in place for formatting the file, the display was quite difficult to read. The IT facilitator became interested in this and offered to try to come up with a solution that would display the same data in the form of a table. Using off-the-shelf components, a solution was created within a single day. The solution was far from perfect but it allowed workers to look at the data in a much easier to read format. Far more important, however, is that this quick-n-dirty solution occasioned a discussion (involving control room workers, the facilitator, and other IT professionals) about the general usefulness of such an application, possible extensions of it, of how this would mesh with working practices, and what the effort/benefits tradeoffs might look like.

The log files are routinely used to trace the trajectory of individual engines or to trace occurrences of a particular error or problem situation. An extension was created over the course of the next days that allows workers to search for

occurrences of error codes and messages, or to find all engines that have been worked on, on a particular day. The development work took place within the control room, occasioning many discussions about what the system should look like and how it would be worked with. Importantly, possible tradeoffs and shortcuts were discussed and negotiated in context and they were immediately put to the test by applying the system in the actual work setting. An example is that the search function does not allow workers to formulate queries of arbitrary logical form but it is restricted to a simple conjunction of instances. It was determined through in-situ discussion and "tinkering with the system" that generic logical operators were not immediately needed (although they were seen to be generally useful) and thus a temporary trade-off was made between development costs and immediate benefits. Another discussion of effort/benefit evolved around the question of how often people would have to deal with those log-files and one of the IT workers said that she thought that it would be needed in the future since the system writing the logs was under constant development. This points to the fact that co-realisation is part of a wider context of systems development and use in the organisation, and that it is ideally suited to cover the issues that are left unaddressed by more formalised processes of IT design.

CONCLUSIONS

The problematic nature of contextual design requires that those seeking to design, develop and evolve IT systems cooperatively look elsewhere. As a candidate solution, we have described co-realisation, a method of design and implementation that treats user involvement as a *sine qua non*. This is not simply because of some 'political' preference for user involvement but because we believe that the development of work affording systems requires close cooperation between all parties and not simply a process redesign method that seeks to understand in order to replace.

Co-realisation capitalises on membership, examining the working division of labour and use of artefacts therein. It enables members working together to assess the potential of off-the-shelf technologies and to develop work affording systems shielded from the imperatives of premature closure found in 'traditional' systems development methodologies, and to abandon what evidently become false starts. It gives non-IT professionals first-hand access to the development process, taking seriously the ideas of participatory design. Most importantly, co-realisation enables accountability of design activity.

Co-realisation attends to the design of work *and* work affording artefacts as a pair – the way that the system is designed is reflexively related to the configuration of work and it is possible for members to suggest changes in both system and practice. That is to say, the system and the work are potentially co-realised in a reflexive relationship: one can change the system and change work practice – something that we find problematic in the work of process driven work such as that offered in contextual design. The point is that contextual design takes a superficial look at work practice and aims to develop new systems based on this. We cannot see that the rather brief engagement allows contextual designers to become members, let alone to have the sense whereby they can do the kinds of work we have described above. Again, this speaks to our feeling that the ethnographic dimension of contextual design is a 'ticket' as opposed to a thoroughgoing ethnographic engagement of the kind that ethnographers (even those who espouse quick and dirty ethnographies) would recognise.

Finally, it is possible to say that work reconfiguration and the implementation of IT systems go hand in hand in contextual design. This may be what is demanded in some As we have demonstrated in our case studies, the need in many instances is surely to afford work as opposed to reconfiguring it. With this principle firmly in mind, co-realisation recognises the inevitability of change and regards it as an integral component of creating work affording systems.

ACKNOWLEDGEMENTS

We would like to thank staff from North British Hospital and EngineCo for their participation. This work is funded by the UK EPSRC and the Dependability Inter-disciplinary Research Collaboration (DIRC).

REFERENCES

1. Beyer, H. and Holtzblatt, K. *Contextual Design: Defining Customer-Centred Systems*. Morgan Kaufmann Publishers, 1998.
2. Büscher, M., Mogensen, P. and Shapiro, D. Bricolage as a Software Culture. *Proceedings of the COSTA4 Workshop on Software Cultures* (Vienna, Dec. 1996), Technical University of Vienna.
3. Button, G. The ethnographic tradition and design. Design Studies, vol. 21(4), July, 2000
4. Hartswood, M., Procter, R., Rouchy, P., Rouncefield, M. and Sharpe, M. Being There and Doing IT in the Workplace: A Case Study of a Co-Development Approach in Healthcare. *Proceedings of Participatory Design Conference*, (NY, Dec. 2000), p. 96-105.
5. Martin, R. eXtreme Programming Development Through Dialog. IEEE Software, 17 (4), 2000.
6. Preece, J., Rogers, Y. and Sharp, H. *Interaction Design: beyond human-computer interaction*. John Wiley & Sons, 2002.
7. Suchman, L. Located Accountabilities in Technology Production. In D. MacKenzie and J. Wajcman (eds.) *The Social Shaping of Technology* (2nd Ed.), Open University Press, 258-265, 1999.
8. Voß, A., Procter, R., Williams, R. Innovation in Use: Interleaving day-to-day operation and systems development. *Proceedings of Participatory Design Conference*, (NY, Dec. 2000), p. 192-201.