

Ethnographies of Code: Coordination Between Pairs in Agile Pair Programming.

Chris Hinds, Ondrej Mates, Marina Jirotko.

5th Feb 2006

Ethnographies of code seem to raise the possibility of making three broad kinds of contribution. They may inform and further our sociological understanding of the organisation of work; they might provide reflections that the participants of a study might find valuable; and they may help to inform the approaches that those participants use. It is on the last of these that this paper focuses. Cockburn's book on Agile software development is widely respected. In this he motivates the use of Agile methods by presenting particular views of communication and collaboration. These understandings are important as they help to both ground and define the Agile agenda. In light of our own ethnographic studies of Agile methods in action, we sought to re-examine Cockburn's views.

Cockburn has been a critical figure in the Agile community, he was one of the 17 original signatories of the "Agile Manifesto" (February 2001), and his book on Agile methods [1] has been extremely influential in popularising the approach. In this text Cockburn describes various inherent challenges for software development, often connected with the way that people collaborate and communicate. These priorities are further reflected in the wider community; the Agile Manifesto itself, for example, places emphasis on: "individuals and interactions over processes and tools." These reflections are thus critical to the Agile project, they help to outline the ways in which it considers itself distinct from more mainstream methods of software development (which might by contrast, place a greater emphasis on following an iterative development process, than on facilitating the interactions between its developers). The recommendations of the Agile methods are thus dependent on the challenges it views as essential, and in particular on the way it views communication and collaboration. It is in this respect that Cockburn's views on the subject are foundational to Agile methods.

Cockburn tends to outline his views on collaboration and communication by analogy and metaphor. Interestingly, some of the ideas he presents through them may seem vaguely familiar to the ethnomethodologist. For example, he highlights the importance of gesture, pointing, and timing during software development discussions (p92). One of his critical notions is of "osmotic communication," an ability to monitor background conversations in the workplace and respond at relevant junctures. As Cockburn puts it: "this taking in of information without directly paying attention to it is like the process of osmosis, in which one substance seeps from one system, through a separator, into another" (p81). There would seem to be natural parallels between this description of everyday coding, and the work of London City traders [2]. Traders, for example, showed a similar sensitivity, being able to pick out relevant calls from what appeared initially to be merely background noise.

In a further metaphor Cockburn describes "information passing" as a process of gas dispersion (p79). By this he means to indicate the ease with which people can communicate when they are co-present. With each barrier that is placed between co-workers, whether that is simply facing in opposite directions, having an office divider placed between them, or worse, working in separate offices, the efficiency with which they may communicate decreases. The "gas" metaphor is thus a means to suggest that physical barriers slow down the speed of "dispersion."

These descriptions are used to develop a particular view of that which is important to the development of code. It is view which clearly values communication. Moreover, this view is used to motivate some specific proposals for the organisation of software development. The gas dispersion metaphor is used to introduce the idea of a "caves and common" (p89, but originally a proposal from XP) arrangement of the workplace. In this, a workplace is divided in two, half for project working, and half as personal space where developers can check email and make phone calls. The common half of the space is arranged to maximise the developers abilities to oversee and overhear each other, to maximise gaseous exchange, or the rate of osmosis.

In the above, effective communication is taken as critical to the challenge of developing software. Something for which barriers must be removed. It is a commitment which places Agile methods in tension with more traditional forms of software development, which tend to take programming as an activity requiring developers to engage in long solitary periods of focussed concentration. However, this aspect of coding is not entirely ignored by the Agile movement. Cockburn talks about the popular perception of programmers as

“non-communicative,” as being a professional sub-culture. He talks about the practical challenge of sustaining levels of concentration and emphasises the importance of maintaining “flow.” He describes programming as “tying together complex threads of thought,” where interruption can mean that these mental structures collapse and thus must be rebuilt (p108). The so-called anti-social behaviour is thus rendered as a protective professional skill.

For Agile methods there is consequently a tension between encouraging communication, for example through the pair programming and caves and commons of XP, while recognising the need for maintaining a fragile focus and concentration on the task at hand. On balancing the public and private needs of a programming team, Cockburn admits he has found no consensus (p60). Interestingly, and although the initial focus had been elsewhere, it was on this tension that our ethnographic study seemed to shed most light.

In the summer of 2005 a short study, our interest both in these issues and others, motivated a short ethnographic study of Agile software development. Our study, and subsequent analysis has begun to explore pair programming, and how one pair of programmers might relate to others within the workplace. We have developed an interest in what might be described as the etiquettes of pair programming. For example, by looking at the skills that programmers use to judge whether it is appropriate to “interrupt” the working of another pair of programmers. Although in its early stages, our ongoing interest is motivated by a hope that such “ethnographies of code” will help to provide a more accurate view of the real work of software development.

- 1 Cockburn A (2003) Agile Software Development. Addison-Wesley, Boston
- 2 Heath C, Jirotko M, Luff P and Hindmarsh J (1993) Unpacking Collaboration: The Interactional Organisation of Trading in a City Dealing Room. In Proceedings of the Third European Conference on Computer-Supported Cooperative Work, pp. 155-170.