Facework in a pair-programming session

BATES, Christopher <http://orcid.org/0000-0002-1183-1809>, DOHERTY, Kathy and GRAINGER, Karen <http://orcid.org/0000-0001-9379-4361>

Published version

BATES, Christopher, DOHERTY, Kathy and GRAINGER, Karen (2011). Facework in a pair-programming session. In: Psychology of Programming Interest Group Work in Progress, Sheffield Hallam University, April 18-19, 2011. (Unpublished)

Facework in a Pair-Programming Session

Chris Bates, Kathy Doherty, and Karen Grainger

CCRC/C3RI, Sheffield Hallam University
`c.d.bates@shu.ac.uk`

1 Introduction

Very little software gets developed by one person acting totally in isolation. Most larger projects are highly managed cooperative social activities within specific institutions the large corporation, the web services provider, the game development company and so forth.

Improved communication is part of the agile solution to the problems we have in developing software. [1] showed that some development practices restricted feedback within teams because *Empirical studies suggest that a significant portion of the software maintainer's time is required to understand the functionality of the software to be maintained.* [5] demonstrated that when developers work together as tightly knit pairs this changes as their productivity and *feelgood* increase. Other studies of agile methods which promote communication at daily meetings such as [6] and [2] have shown similar results.

2 Using ethnomethodology to examine interactions

The ethnomethodological approach is to examine how we create our own understandings of the world and then use them to build social situations. This makes it particularly well-suited to the study of work and workplaces [4]. Ethnomethodology tells us that as we go about our normal lives we engage in practical activities which are accountable, that is they can be observed and understood by others and that this understanding is context sensitive.

[7] argues that ethnomethodology offers firm theoretical grounding from which to explore and understand the activities and challenges that are part and parcel of software engineering. He suggests that programmers are for example, inevitably engaged in a search for *adequate indexicality* as they deal with changing requirements and strive to develop a product, software, that is essentially complex and *invisible* [3]. Indexicality refers to the way in which the meaning of text (e.g. the specification of a requirement) depends on the specific context of use or production, including mutually understood norms and expectations. Because programmers are searching for adequate indexicality they are managing *incomplete communication* and must therefore arrange interactions in order to disambiguate and clarify tasks and responsibilities to deliver acceptable products. We see this in pair programming, for example,where two developers work together and necessarily spend their time discussing the code which they are creating.

The specific example we examine is from a pair-programming situation in which some existing code is going to be reworked. Taking someone else's work and modifying or improving could be a socially difficult process. Although developers know that code has to be changed, the social processes through which this is arranged question ideas of self, of ownership of one's creations and of relationships within teams. The code in our case study had failed in some, unspecified, way some time previously, and the two developers we study are going to have another go at it. We see them talk about the history of the code and its problems as they initiate the new development. The more senior of our *pair* wrote some of the failing code, during this interaction he tries to maintain his social status, through facework, whilst working co-operatively with his junior colleague.

Face is described in [8] as *the presentation of a civilised front to another individual* which is culturally located and which provides a *claimed sense of self in an interactive situation.* Facework is the set of strategies, acts of self-presentation, verbal and non-verbal expressions and impression management which combine to present and preserve face. We shall see that facework is an important part of the interaction between developers, even when they are used to working together closely.

3 Analysis

In the interaction which we observe here, Darren and Andy are re-working some code which was started and abandoned a few months previously. The code synchronizes contact lists between the server and a user's mobile phone. We are analysing this interaction in context because we are interested in how specific working practices are expressed as a specific task is undertaken.

Our field notes include briefings from the developers about their work and about the specific task we see them undertaking here. The task is framed by the knowledge of the earlier failure and must be analysed through the previous abandonment of this work without which they would not be engaged in it in this way.

1	Andy	what's the sense around using echo? (1.7)
2	Darren	at the moment that functionality's all kind of been turned off 'cause it it got to a certain point (1.1) and it wasn't really (2.8) to [improve the user experience]
3	Andy	[what was the point of th]at?
4	Darren	the (.) the pla' when we did it to echo (.) then that was (.) the database was here the synchronisation was going and the contacts were pushed up (1.6) into this database. THERE WAS various issues (.) and from a usability point of view it wasn't (1.0)
5	Andy	Hmm mmm (1.0)
6	Darren	err >working that< well and also echo (0.9) echo was kind of err (1.7) Yeah (1.0) it had problems (0.3) it was kind of a big
7	Andy	laughs
8	Darren	laughs
9		a big test (.) but this is you know this is kinda tryin' a make it easier

Before this piece of talk they had been discussing database structures. Andy's turn, "what's the sense around using echo?", is a major challenge and a change of topic. Here, Andy is referring to a sketch we saw them make earlier in the day. The sketch started as a drawing of the existing, failed, design. It was used to identify the major components and their relationships and in a discussion about the functionality of the system. Their previous talk has avoided the analysis of problems in that code, here Andy asks for an explanation.

Darren hesitates and stumbles showing that he is thrown by the turn. He has been focussing on the future, on what needs to built yet his colleague is intent on examining the past in more detail. Darren becomes hesitant the pauses around this statement total over 5.5 seconds and once again starts to hedge. When discussing the database design he was clear and the pauses were due to his sketching.

Darren starts by deflecting attention, saying that the functionality was "kind of" turned off but notice that he hedges. We would expect the functionality to be either on or off, he doesn't commit to this but places it in a nebulous third state. There is a false start in his justification, "cause it got to a certain point", before he says that the code wasn't improving the user experience. The latter statement is a perfectly normal reason for removing or refactoring a piece of code. Darren's statement that the code is turned off suggests that he wants to move on to something else. Moments earlier he was outlining the new "little" database, here he wants to continue thinking about they are going to do.

Andy interjects during this explanation with "what was the point of that", meaning why remove the code. This interruption comes at a possible turn transition point following Darren's long delay and overrides the explanation. Andy is referring back to the old system once again, pursuing a response. His question threatens Darren's face because he has been trying to look forwards. It has the effect of throwing Darren off so that, once again, he begins to stumble over his words.

References

1. Rajiv D Banker, Gordon D Davis, and Sandra A Slaughter. Software development practices, software complexity and software maintenance performance: a field study. *Management Science*, 44(4):433–450, 1998.
2. Gabrielle Benefield. Rolling out agile in a large enterprise. In *Proceedings of 41st Hawaii International Conference on Systems Sciences*, Hawaii, USA, January 2008. IEEE.
3. Fred P Brooks. *The Mythical Man-Month anniversary ed.* Addison-Wesley Longman, 1995.
4. Harold Garfinkel. *Ethnomethodological studies of work.* Studies in ethnomethodology. Routledge and Kegan Paul, 1986.
5. Matthias M Muller and Frank Padberg. An empirical study about the feelgood factor in pair programming. In *Proceedings of the 10th International Symposium on Software Metrics*, Chicago, USA, September 2004. IEEE Computer Society.
6. Linda Rising and Norman S Janoff. The scrum software development process for small teams. *IEEE Software*, July/August 2000, August 2000.
7. Kari Ronkko. Interpretation, interaction and reality construction in software engineering: An explanatory model. *Information and Software Technology*, 49:682–693, 2007.

8. Stella Ting-Toomey, editor. *The Challenge of Facework: cross-cultural and interpersonal issues.* State University of New York Press, Albany, New York, USA, 1 edition, 1994.