Scrum Down: A Software Engineer and a Sociologist Explore the Implementation of an Agile Method

Chris Bates
C3RI/CCRC Sheffield Hallam University
Howard Street
Sheffield, United Kingdom
c.d.bates@shu.ac.uk

Professor Simeon Yates
C3RI/CCRC Sheffield Hallam University
Howard Street
Sheffield, United Kingdom
s.yates@shu.ac.uk

ABSTRACT
This paper provides an overview and position statement on the work undertaken as part of a project to explore the implementation of Scrum in the context of an interactive digital media software development company. The project is being undertaken in the Communication and Computing Research Centre at Sheffield Hallam University.

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]: Management

General Terms
Management, Human Factors.

Keywords
Scrum, sociology, ethnography, ethnomethodology, phenomenology, discourse.

1. INTRODUCTION
This paper is based on early work studying the implementation of an agile methodology, Scrum, in a software house which produces applications for use in developing interactive digital media. The company is running a trial of a small scrum project alongside its more established plan-based approach in the hope that it will lead to better communication within and more satisfaction for customers.

The clichéd view of software development, in particular programming, taken by outsiders is that is an essentially anti-social practice. Those who write code are perceived to work in isolated ways and to prefer technology to people. Whilst this is undoubtedly true of some individuals it is not true of the majority of software developers. Programming is an essentially social activity which requires both genuine teamwork and innovation if it is to be successful [5, 6].

CHASE'08, May 13, 2008, Leipzig, Germany.

Software development is often analyzed from a business perspective which focuses upon metrics such as project cost or programmer productivity. This view of the industry takes a predominantly quantitative approach to its subject with numeric values attached to success or failure. However it is not obvious that saying, for example, that 18% of projects fail [10] is really helpful to project managers and developers trying to maximize their completion rates. In [10] Jim Johnson, chairman of the Standish Group, points out that the usual situation for IT projects is "over budget, over time, and with fewer features than planned". But what lies behind that comment is a complex mix of technical and social interactions and problems.

To understand the figure for failing or challenged projects, those which fit Johnson's description, we must understand what happens during software development. Over the years effort has gone in to devising new project management approaches and new development methodologies to good effect. The number of challenged projects has increased whilst the number of outright failures has fallen yet the figure for outright success has hardly changed since the late 1990s.

It has been known for years that social and environmental factors within the workplace greatly affect the chances of a project succeeding [5]. By examining closely how software developers operate in their daily practice we should be able to understand what happens during agile development that makes it attractive to developers. Further, if software development is to be understood as a social process then it must be studied using techniques from the social sciences. In this work a team of developers will be followed using ethnographic techniques, such as observation and conversation analysis, as they move from plan-driven development to a set of agile processes [4].

The C3RI is concerned with analysing the nature of effective communication, exploring the effective use and design of technology, and investigating research issues common to these two areas of study. The centre's research activities are incorporated within the domains of Communication, Cultural and Media Studies, Library and Information Studies, and Computing. The Centre, led by Prof. Yates, is committed to carrying out research with an applied and inter-disciplinary emphasis. It has cultivated strong links with other research disciplines within the university and boasts extensive links with industrial and other external agencies. A key aspect of the work of the centre is the exploration of the linguistic, interpersonal, cultural and technical issues in relation to ICT development and use.

2. BACKGROUND

The company that we are working with is a relatively small development house with approximately twenty five developers and a smaller number of quality assurance staff. The development team and head office are in the UK but the sales arm is primarily based in the USA. The company makes authoring software which is used in the production of DVDs by a number of large clients in Hollywood. Commercial sensitivities prevent too much detail about the products or clients being revealed in this paper.

Development is currently undertaken using planned procedures and conventional project management with developers allocated to tasks according to their availability and to the specific requirements of the task. A dedicated quality assurance team handles testing with work assigned to each tester as it and they become available.

Developments are driven primarily by a sales team based in the USA. Sales are usually modifications of, or customizations to, the company's main product so that it can be used in particular environments. The sales team not only makes deals but also set delivery dates as part of that process. Although there are tensions around delivery dates, which can be set on realistic but very tight timescales, the engineers strive to meet them wherever possible. To make matters worse feature-creep can be a problem as new requests come in from Sales.

Tasks typically last from six to eight weeks using small groups of developers. Completed code is handed off to QA for testing

The company currently organizes its work around a waterfall model and uses an engineering approach to development. Managers feel that whilst this approach works, it is insufficiently flexible given the dynamic nature of their market. They want to be able to allocate work rapidly to teams, to be able to estimate delivery times and to involve the sales team in the ownership of the product. On a larger scale the management wants to move avoid the creation of a blame culture which could arise if Sales see Engineering as being unwilling or unable to deliver, and Engineering see Sales' deadlines and requirements as unreasonable. The result of all this is a trial project involving the use of Scrum.

A single project will be run using Scrum whilst alongside it other projects continue using the established approach. The scrum team will be supported by an outside mentor to ensure that they get maximum benefit from the approach. At the completion of the initial project the use of scrum will be evaluated to ascertain if it should be used more widely. Evaluation will involve empirical studies undertaken in our work which will compare the situation before and after the project - and will look at how the rest of the developers view Scrum from the outside.

2.1 Scrum

There are a number of different agile methods of which the best known are probably extreme programming, XP, and Scrum. The essence of agile processes is the ability to react to changing requirements [5]. Each agile method has its own approach to the running of a software development team but all of them place flexibility at the heart of their approach.

Proponents of flexible software development claim that it can only happen if a number of conditions are met [16]. These include placing individual developers and their skills at the heart of the process, collaborating with customers throughout development, treating working code as the major artifact of the project and reacting to change instead of blindly following a plan. Scrum is a classic agile method in all of these respects if implemented properly. Since Scrum has no hard and fast rules there is no way to mandate, for example, that individuals are placed at the heart of the process

Scrum projects are organized around small closely knit teams working on rapid iterations of a develop-release cycle. Development takes place in tightly focussed sprints which last just a few weeks. At the start of each sprint goals are set including the functionality to be included in the next release and a hard deadline on which to release. During the sprint, brief daily meetings are held to ensure that the team remains on-track [12].

In the company we are studying projects are six to eight weeks long which means they should be accomplished in two or three short sprints. Ideally each sprint releases a functional product but that will not always be the case here. Instead sprints are likely to release products which can be evaluated by the customer to ensure that they get the end product that they are expecting. This places additional focus on the sales team. Sales set the final deadline but must be prepared to give feedback on intermediate releases as they come from each sprint. That feedback may come from the end customer or from the sales team acting as their proxy.

3. METHODOLOGICAL ISSUES

Evaluation of any project is difficult. Many quantitative metrics based on productivity or error rates exist. These work well for certain classes of large, planned project. It is not clear that these metrics can be applied to this scrum pilot. Experimentation is not possible due to time, cost and organizational constraints although there are clearly interesting experiments which can be performed in the area. In [9] comparisons of student project using XP and a modified version of the Rational Unified Process showed that those using the agile method communicated more, documented less but that both processes required the same overall effort. Repeating the work with experienced software developers might yield more widely applicable results.

In our study the scrum project will run alongside a number of existing plan-driven projects. We cannot disrupt the normal flow of work to perform experiments so must relay on less invasive techniques.

There is a long history of social scientists and software developers cooperating on projects. Much of this inter-disciplinary effort has been in CSCW and HCI where qualitative approaches such as ethnography have found favour [1]. We are continuing in the same vein but, as in [15], examining the inner workings of software development teams. Our overall approach will be a broadly ethnomethodological to begin with.

Ethnomethodology starts from the assumption that social order arises from peoples' interactions as they go about their normal life, in this case at work [8]. It further contends that organizations are constituted by and through situated actions and talk in ongoing day-to-day work and action.

This starting point will provide the project with the opportunity to examine how the team in the target organization understands and implements Scrum as practical actions. The project will therefore

focus on an evaluation of Scrum as understood and implemented in the context of this project and this organization.

This contextual focus does not provide the quantitative comparative measures of metrics based or experimental studies. It does, though, allow for rich or "thick" descriptions of the implementation process. These descriptions can provide insights into the day-to-day practical actions that underpin agile methods as well as the challenges of first time implementations of such methods.

Ethnomethodology is of course only one of a number of approaches to workplace studies. As the project develops we will reflect critically on the usefulness of this approach. Alternatives, such as ones drawn from social studies of science practice including as Latour's work on “laboratory life”, may provide useful analytical and theoretical counterpoints.

Initially, shadowing and observation are being used as developers go about their normal daily routine [14]. Observations of groups within the office are made to acclimatize the developers to the presence of an outsider armed with notebook and pencil. Once the developers are used to being observed much closer studies can be made of individuals.

Because the observer is an experienced software engineer, the minutiae of a developer's tasks are well understood. Tasks such as debugging, coding, checking into or out of CVS can be noted but do need to be examined in detail. Rather than concentrate on *what* is being done we examine *why* it is happening. For example, is code being checked into a repository because the QA team needs to run integration tests or because someone else is writing code which requires this update?

Shadowing individual developers means that the observer can look at interactions within the group both in formal team meetings and in more *ad hoc* situations. Debugging provides a perfect example of the latter. Within the company there is an established practice though which people pair up to discuss particularly difficult problems. Staff in the studied company are generally willing to help a colleague but people don't ask just anyone for help. Instead certain people, often team leaders, supply more than average amounts of help.

Part of the shadowing is to ask questions about why some people do more helping than others. The reasons may be obvious: personality, experience, skill level. Only by closely observing over a period of time will the real reasons be discovered. Do team leaders use helping as a way of asserting their status within a relatively flat hierarchy? Is helping a learning strategy? Does being helpful mean that one can briefly change task and so break monotony?

Observation is only part of the process of understanding complex interactions between developers. A lot of work is done within conversations to create structure and meaning. Within technical conversations there is not only an exchange of factual information but also the building of *face* and the creation of a shared sense of the situation. This shared understanding comes from tacit reasoning and leads to actions which are *accountable* [11].

We could try to understand conversation between developers by examining the superficial technical layer of their discussion but that would not tell us everything. Social science provides us with a technique called conversation analysis, CA, which helps us analyze what is happening during these discussions. CA gives us a method of understanding both the content of an interaction and its structure. Using CA we are able to look at how relationships are built within the conversation so that we can discover how team leaders or product managers may exert power or influence during interactions.

CA also gives us the chance to consider how the context affects the form and outcome of a conversation. A debugging session may begin with a question about a specific problem but contain a long discussion of surrounding code during which the participants are feeling each other out, trying to establish what each of them knows. This process lets them establish the point at which they go more deeply into the specifics of the problem at hand. As the bug is narrowed down the conversation may become equally narrowed. Once the problem is resolved the conversation may open out into other matters not related to work which signify the end of the interaction. CA lets us show when and how the participants move through these phases.

Where disagreements may arise, for example between developers and QA, we may see different conversational forms. The disagreements may have little or nothing to do with technical or business issues but instead may be due to having not established an appropriate context before beginning.

4. EPISTEMOLOGICAL ISSUES

The goal of our research is simply to see how an agile process changes the environment and interactions within a team of experienced developers.

Ethnomethodology works from a very specific epistemological position within the social sciences. It has, over time, become strongly associated with workplace studies including the use of methods developed by Button and colleagues [2, 3, 7]. Though ethnomethodology and its sister activity of conversation analysis are often cited in sociological discussions of the *linguistic* or *relativist* turn in social theory, they in fact remain highly grounded and empirical in the sense of data collection and close analysis. Debate continues in the ethnomethodology community on the problem of including wider social and organizational issues and data as well as the role of technological artifacts and processes in the analysis [2].

Situations such as the implementation of scrum in the target organization provide excellent contexts within which to assess ethnomethodological approaches as tools for understanding software design practices and as sociological methods in themselves. Whilst the work would be worthwhile event if it were only feeding back into the academic social science community its real value will only be realized if lessons can be drawn from it for software engineers. Ethnographic studies are easy to criticize for their reluctance to engage with changes in practice within the observed community [2, 15]. Our hope is that by examining communication between software developers and within development teams we can engender a deeper understanding of their practice in the programmer community.

The project will reflect upon the epistemological assumptions underpinning three distinct areas: those underpinning ethnomethodological approaches to workplace studies, especially software engineering workplaces; those underpinning more metrics-based approaches to the evaluation of software development teams and projects; and, finally, those that underpin

the developers and the organizations understanding of method, implementation and evaluation processes.

5. CONCLUSION

This work is still in its early stages. We hope that it will begin to show the effect of project type, size and structure on a development environment. The accepted wisdom is that using well formed teams working on defined problems using agile methods increases cohesion, satisfaction and performance. Ethnographic observation of development practices will let us see if the benefits of agile processes accrue from changes to developers' work and communication patterns or if they arise from changes in management practice.

We also hope to discover common ground between software engineers and sociologists, to enjoy working outside our normal domains and comfort zones. We are interested in seeing what emerges from the interaction between our different academic approaches and the world of commerce. Finally, we hope that our work can feed back into the practice of software developers and those who manage them.

6. REFERENCES

[1] Anderson, B. 1996. Work, Ethnography and System Design. The Encyclopedia of Microcomputers. Volume 20. Ed Kent, A. and Williams, J.G. Published Marcel Dekker, New York.

[2] Button, G. and Dourish, P. 1998. On Technomethodology: Foundational Relationships between Ethnomethodology and System Design. Human Computer Interaction, Vol 13 No. 4.

[3] Button, G. and Sharrock, W. 1998. The organizational accountability of technological work, Social Studies of Science. Number 28.

[4] Cohn, M. and Ford D. 2003. Introducing an Agile Process to an Organization. IEEE Computer. Volume 36.

[5] DeMarco, T. and Lister, T. 1999. Peopleware. Dorset House Publishing.

[6] Desouza, K. and Awazu, Y. 2005. Managing radical software engineers: between order and chaos. In Proceedings of the 2005 workshop on Human and social factors of software engineering. Published ACM.

[7] Fisher, S. and Todd, A.D. eds. 1983. The Social Organization of Doctor-Patient Communication. Center for Applied Linguistics. Washington, D.C.

[8] Garfinkel, H. 1984. Studies in Ethnomethodology. Polity Press.

[9] Germain, E. and Robillard, P. 2005. Engineering-based processes and agile methodologies for software development: a comparative case study. The Journal of Systems and Software. Number 75. Published Elsevier.

[10] Hartmann, D. 2006. Interview: Jim Johnson of the Standish Group. http://www.infoq.com/articles/Interview-Johnson-Standish-CHAOS.

[11] Heritage, J., 2001. Goffman, Garfinkel and Conversation Analysis. In The Handbook of Discourse Analysis. Edited Schiffrin, D. et al. Blackwell Publishers

[12] Janoff, N.S. and Rising, L. 2000. The Scrum software development process for small teams. IEEE Software volume 17 number 4.

[13] John, M., Maurer, F. and Tessem, B. 2005. Human and Social Factors of Software Engineering - Workshop Summary. In ACM SIGSOFT Software Engineering Notes volume 30 number 4.

[14] Lethbridge, T., Sim, S. and Singer, J. 2005. Studying Software Engineers: Data Collection Techniques for Software Field Studies. Empirical Software Engineering. Volume 10. Springer Science and Business Media.

[15] Sharp, H., Woodman, M. and Robinson, H. 2000. Using Ethnography and Discourse Analysis to Study Software. In Proceedings ICSE 2000. Janice Singer et al, editors.

[16] The Agile Manifesto. 2001. http://agilemanifesto.org