

Agile Methods and Motivation: A Position Paper

Nathan Baddoo¹, Tracy Hall¹ and David Wilson²

¹School of Computer Science
University of Hertfordshire

²Faculty of Information Technology
University of Technology, Sydney

Abstract

Even though it has been widely acknowledged that human factors are critically important to effective software development processes, they continue to be neglected in the industry. One such human factor is the motivation needs of software developers. In this paper we argue that plan-driven software development methods do not satisfy the motivation needs of software developers, who have been shown to have significantly different motivation needs. This is because plan-driven development methods have been derived from the manufacturing and engineering disciplines and thus tend to be more predictive and less people oriented. We hypothesise that agile development methods may better satisfy the motivation needs of software developers because they happen to be adaptive and essentially people oriented. In this paper, we propose a programme of research to investigate the impact of agile methods on software developers' motivation needs. This programme of research should include a comparative study of how motivation needs are met in planned-driven versus agile environments. We suggest that the results of this study can provide useful insight into the factors that can improve software developers' performance.

1 Introduction

It is widely acknowledged that human factors are critically important to effective software development processes [Hall & Wilson, 1997; Krasner, 1997; Wilson & Hall, 1998; Ahuja, 1999; Hammock, 1999; Wilson, Hall & Baddoo, 2001]. Despite this, human factors continue to be neglected in the software industry [McDermid and Bennett, 1999]. Previous work [DeMarco and Lister 1987] suggests that traditional development methods do not effectively address the human factors in software development. Such traditional development methods have largely been borrowed from the manufacturing and engineering disciplines and, despite their success in these disciplines, they have performed less well in software development [Hyde, 2002; Elvin & Davies, 2002; Yetton et al, 2000].

The motivational needs of software developers have been shown to be significantly different to those of other professionals, including engineers [Couger & Zawacki,

1980; Couger, 1988; Warden & Nicholson, 1995; Khalil et al, 1997; Sharp et al, 1999]. Consequently, forcing software developers to adopt development methods that have ostensibly been developed for use in other disciplines might mean that the specific motivational needs of software developers are not being met effectively. It therefore follows that software developers may not be performing as well as they can and this might be related to the poor performance of the software industry that is regularly reported by way of software disasters.

A variety of development approaches are practised in software development. However Boehm & Turner [2003] classify all development approaches as either 'plan-driven' or 'agile'. We argue that agile methods are the first development approaches to match the motivational needs of software developers. The agile environment promises to be an influential approach towards optimising software developers' motivation and therefore improving the performance of those software developers. Improved software development performance can only have a positive impact on the success of the software industry.

2 Motivation in the Software Industry

To understand the impact of motivation on the performance of software developers it is important to first understand the underlying principles of motivation. One of the best known researchers on motivation is Frederick Herzberg who separated motivation factors into 'satisfiers' and 'dissatisfiers' and described satisfiers as the factors that

“describe man's relationship to what he does, to his job content, achievement on a task, recognition for task achievement, the nature of the task and professional advancement or growth in task capability” [Herzberg, 1964].

Herzberg found that satisfiers are intrinsic to the job. This work was further developed by Hackman & Lawler [1975] who originated the job characteristics theory of motivation and Hackman & Oldham [1980] who developed the Job Diagnostic Survey (JDS).

Couger and Zawacki [1980] applied these motivation principles to workers in the IT industry and their initial survey included 2,500 employees from 50 organisations. The study now contains information on more than 18,000 Americans and 19,500 participants from other countries [McNurlin & Sprague, 1996]. Software developers have the highest growth need strength of any job category that has been analysed using the JDS – they want to work on the latest technology, both hardware and software, and must be continually provided with new challenges to keep them motivated. However, they also have a low need to socially interact with others, but are heavily motivated by opportunities to advance themselves in their jobs [Khalil et al, 1997]. Furthermore software developers also have very high needs to self-advance with the nature of the job being more motivational than peripheral factors such as social liaisons or remuneration [Couger & Zawacki, 1980].

International studies of motivation in the software industry [Fitz-Enz, 1978; Couger & Adelsberger, 1988; Couger et al, 1989; Couger et al, 1991; Couger & O'Callaghan,

1994; Couger & Ishikawa, 1995] have covered such diverse cultures as Taiwan, Hong Kong, Singapore, Australia, Israel, Finland, South Africa, Egypt and Austria. Despite opinion that people from different cultures would be significantly different, these studies show that software developers in these countries also exhibit high growth need strength and low social need strength. The IT profession appears to attract people with similar characteristics, irrespective of their culture - what motivates software developers seems to be the same across cultural boundaries. Overall, the evidence suggests that the motivation factors that are most important to software developers are: rewarding, long term, intrinsic to the job and satisfy a personal need to develop. These factors are predominantly the "satisfiers" as proposed by Herzberg.

3 Plan-driven versus Agile Development Approaches

3.1 Plan-driven Approaches

A variety of development approaches are practised in software development. However Boehm & Turner [2003] classify all development approaches as either 'plan-driven' or 'agile'. Plan-driven approaches have typically emerged from engineering and manufacturing practices and are based on the philosophy underpinning 'sequentially defined' engineering processes [Williams & Cockburn, 2003]. Plan-driven approaches include methods based on models such as CMM [Boehm & Turner, 2003]. The effectiveness of using these approaches has increasingly been questioned. Williams & Cockburn [2003] argue against the use of plan-driven approaches. They argue that sequential plan-driven approaches cannot cope with the inherent change within software development. Some argue that the software quality and performance problems reported in the literature stem from the use of plan-driven approaches. Indeed Kent Beck [Beck & Boehm, 2003] claims that *'Efforts to force developers and customers to work to a procedure... are responsible for results that have uniformly failed'*.

Plan-driven approaches are generally oriented around the technical aspects of software development. The human, social and organisational issues related to software development are rarely explicitly accounted for within these approaches. Many studies show that human factors in software development are most problematic in software projects. Indeed the neglect of human factors in software development is now thought to be a significant problem [Sharp et al, 1999].

3.2 Agile Approaches

Agile methods emerged in response to the poor performance of plan-driven approaches. Agile approaches emphasise the non-technical aspect of developing software where software development is viewed as a highly social activity. Agile approaches are related to the 'inspect & adapt' engineering approach where cycles and feedback loops are short [Cohn & Ford, 2003]. They were formed by the coming together of three independently developed approaches: Europe's Dynamic Systems Development Method (DSDM); Australia's feature-driven development; and eXtreme programming in the US [Williams & Cockburn, 2003]. The resulting agile approach has a manifesto based on the following principles [Fowler & Highsmith, 2001]:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation

- Customer collaboration over contract negotiation
- Responding to change over following a plan

This manifesto indicates that agile approaches are about delivering good products to customers by operating in an environment that is adaptive and has a people-first orientation. They attempt to provide a compromise between no process and too much process and between no documentation and too much documentation. There are two key points that differentiate agile approaches from plan-driven approaches [Fowler, 2003]:

- *Agile approaches are adaptive rather than predictive.* Plan-driven approaches attempt to plan out the software process in great detail over a period of time. This works reasonably well until things change and, therefore, plan-driven approaches tend to resist change. In contrast, agile methods tend to welcome change. Agile processes adapt and thrive on change, even to the point of changing themselves.
- *Agile approaches are people-oriented rather than process-oriented.* Plan-driven approaches attempt to define a process that will work well for any individual allocated to that process. In contrast, agile processes recognise that no process will compensate for the skill of the development team and are designed to support the development team in their work.

Many commentators allude to the benefits of agile approaches, for example Boehm & Turner [2003] say that agile approaches promise increased user satisfaction, lower defect rates, faster delivery times and a solution to rapidly changing requirements. Cohn & Ford [2003] report advantages such as quick decision-making and increased productivity. However, there remains scant empirical evidence identifying the impact of agile methods on software development outcomes. In contrast there are many case studies and company accounts of the success of agile methods. For example, Cohn & Ford [2003] present their experience of introducing the Scrum agile approach into seven companies over the last four years. The overall result was success. Similarly, Williams [2001] reports benefits such as a 15% reduction in development time and cost, improved design quality, reduced defects and enhanced technical skills to “pair programming”, another agile practice.

4 Agile Approaches and Motivation

Although some case studies describe resistance to the introduction of agile methods from developers [Cohn & Ford, 2003], most reports claim that developers respond well to the introduction of an agile environment. The positive response of software developers to agile approaches may be related to Fowler’s [2003] suggestion that agile methods avoid viewing developers as replaceable process components in the way that plan-based approaches are said to do. Instead, agile approaches value the skill of the development team above the structure of the development process. In agile environments the development process is simply provided as a support to skilled developers. This emphasis on the people aspects of software development means that many of the motivational “satisfiers” that Herzberg proposes are provided in the agile environment. This means that agile environments may better satisfy the motivational needs of software developers.

There are many instances in software development where social need strength is a factor, for example, in meetings. Software developers show frustration at lengthy or frequent meetings. We suggest that this is because software developers will participate actively in meetings that are meaningful to them; but their high growth need also causes intolerance for group activities that are not immediately productive. On the other hand, people with a high social need use meetings as a prime device for fulfilling their social need [Khalil et al, 1997]. Traditional planned development methods require software developers to interact extensively with others, yet they probably want as little interaction with others as possible. This is one way in which we suggest that traditional planned development methods do not match software developers' motivation.

We suggest that agile methods are the first development approaches to match the motivational needs of software developers. The agile environment promises to be an influential approach towards optimising software developers' motivation and therefore improving the performance of those software developers. Improved software development performance can only have a positive impact on the success of the software industry.

5 A Research Approach

To date no work has been done to investigate the impact of different development environments on software developers' motivation. In particular, no work has investigated the relationship between the development methods used by software developers and their motivation. We believe it is important to investigate the relationship between methods and motivation especially as the motivational needs of software developers have been shown to be unique; in particular, software developers have the lowest social need strength of any of the 500 occupations that have been measured by the JDS [Hackman & Oldham, 1980].

To explore the relationship between software developers' motivational needs and the use of agile methods, it is necessary to conduct a study of different software development approaches within an industrial environment. Case studies are accepted to be especially suitable for the industrial evaluation of software development methods [Wohlin et al, 2003] such as agile approaches. The flexible and multi-purpose nature of case study research offers opportunities for gaining maximum benefit from industrial collaborators [Shaw, 1999]. We suggest that such an approach can better help understand developers' motivational needs.

Triangulated data collection increases confidence and reliability in the data collected. Harrison et al [1999] and Seaman [1999] advocate triangulation as a high quality strategy for empirical research in software development. Within the case study framework, we propose the following particular research methods to achieve such triangulation: Job Diagnostics Survey (JDS); focus groups; and scenarios. Overall, the triangulated use of the three data collection methods within the case study approach would allow for accuracy checks of the data we collect according to:

1. JDS: *What should developers be saying about their motivational needs?*
2. Focus groups: *What are developers saying about their motivational needs?*
3. Scenarios: *What are developers doing about their motivational needs?*

5.1 Job Diagnostics Survey (JDS)

This questionnaire-based instrument is a standard method of measuring motivational needs. JDS is conceptually sound and its validity and reliability have been substantiated in studies of more than 6000 subjects on more than 500 different jobs in more than 50 different organisations [Hackman & Lawler, 1975]. JDS is based on the job characteristics theory of motivation which identifies three critical psychological states associated with high levels of internal motivation, satisfaction and quality of performance. JDS measures five core job dimensions: skill variety, task identity, task significance, autonomy and feedback. The application of JDS to many different jobs enables comparison between IT professionals and professionals in other disciplines.

JDS can be used to establish a standard measure of the motivational needs of the developers within the industrial context. This can provide a baseline of motivational needs. This baseline can then allow comparison of motivational needs between different groups of developers in different software development environments. JDS would address a research objective *to confirm the motivational profile of software developers*.

5.2 Focus groups

Focus groups involve assembling small groups of peers to discuss particular topics. They have been described as *"a way to better understand how people feel and think about an issue"* [Krueger and Casey, 2000]. Also, *'the comparisons participants make among each other's experiences and opinions are a valuable source of insights into complex behaviours and motivations'* [Morgan and Krueger, 1993].

Focus groups serve as an appropriate data collection method for identifying what developers think about their motivational needs. Focus groups can be used as a tool that allows developers to talk about their needs and to report their own perceptions of their needs. Focus groups would address a research objective *to establish the extent that the motivation needs of software developers are satisfied by planned and agile approaches*.

5.3 Case Study Scenarios

Case study scenarios can be used to explore how individuals would behave within a particular context. They are commonly applied in recruitment, ethics, problem-based learning and construct elicitation. They are applied by working through fictitious scenarios and recording participants' reaction to or reflection of particular events.

Case study scenarios can be used to explore software developers' behaviours in terms of motivational needs. We can set up work through fictitious scenarios to establish how software developers would behave in specified situations. The data collected from this method would then be used to analyse the motivational needs that developers exhibit in scenario-based practice. Case study scenarios would reaffirm the research objective *to establish the extent motivation needs of software developers are satisfied by planned and agile approaches*.

6 Conclusion

We have presented a review of the literature on agile approaches to software development and have postulated that agile methods are the first development approaches to match the motivational needs of software developers. The literature suggests that agile approaches view software development as a highly social activity, are adaptive, and have a people-first orientation. We have argued that these attributes can be viewed from the perspective of the motivational literature and do in fact match the high growth need strength and low social need strength exhibited by software developers.

In summary, the agile environment promises to be an influential approach towards optimising software developers' motivation and therefore improving the performance of those software developers. Improved software development performance can only have a positive impact on the success of the software industry.

References

- Ahuja, S. (1999) Process Improvement in a Rapidly Changing Business and Technical Environment in *Proceedings of 4th Annual European Process Group Conference* Amsterdam June 6-10, C303.
- Beck, K. and Boehm, B. (2003) Agility through Discipline: A Debate *IEEE Computer* June:44-46.
- Boehm, B. and Turner, R. (2003) Using Risk to Balance Agile and Plan-Driven Methods *IEEE Computer* June:57-66.
- Cohn, M. and Ford, D. (2003) Introducing an Agile Process to an Organization *IEEE Computer* June:74-78.
- Couger, J. D. (1988) Motivators and Demotivators in the IS Environment *Journal of Systems Management* **39** (6):36-41.
- Couger, J. D. and Adelsberger, H. (1988) Comparing Motivation of Programmers and Analysts in Different Socio/Political Environments: Austria compared to the United States *Computer Personnel* **11** (4):13-16.
- Couger, J. D., Borovits, I. and Zviran, M. (1989) Comparison of Motivating Environments for Programmer/Analysts and Programmers in the US, Israel and Singapore in *Proceedings of Hawaii International Conference on Systems Science* pp316-323, IEEE Computer Society Press.
- Couger, J. D., Halttunen, V. and Lyytinen, K. (1991) Evaluating the Motivating Environment in Finland compared to the United States: A Survey *European Journal of Information Systems* **1** (2):107-112.
- Couger, J. D. and Ishikawa, A. (1995) Comparing Motivation of Japanese Computer Personnel versus those of the United States in *Proceedings of the 28th Hawaii*

International Conference on System Science pp1012-1019, IEEE Computer Society Press.

Couger, J. D. and O'Callaghan, R. (1994) Comparing the Motivations of Spanish and Finnish Computer Personnel with those of the United States *European Journal of Information Systems* 3 (4):285-291.

Couger, J. D. and Zawacki, R. A. (1980) *Motivating and Managing Computer Personnel* Wiley and Sons.

DeMarco, T. and Lister, T. (1987) *Peopeware - Productive Projects And Teams*. Dorset House.

Elvin, R. and Davies, A. (2002) Changing the Learning Failure of IT Systems Development Practice *British Computer Society Review*.
<http://www.bcs.org.uk/review02/html/p053.htm>

Fitz-enz, J. (1978) Who is the DP Professional? *Datamation* September:125-128.

Fowler, M. (2003) The New Methodology
<http://www.martinfowler.com/articles/newMethodology.html>

Fowler, M. and Highsmith, J. (2001) The Agile Manifesto *Software Development* August:28-32.
<http://www.agilemanifesto.org>

Hackman, J. R. and Lawler, E. E. (1975) Employee Reactions to Job Characteristics *Journal of Applied Psychology* 60 (2):159-170.

Hackman, J. R. and Oldham, G. R. (1980) *Work Redesign* Addison Wesley.

Hall, T. and Wilson, D. (1997) Views of Software Quality: A Field Report *IEE Proceedings on Software Engineering* April:111-118.

Hammock, N. (1999) Critical Success Factors in Software Process Improvement in *Proceedings of 4th Annual European Process Group Conference* Amsterdam June 6-10, C304a.

Harrison, R., Baddoo, N., Barry, E., Biffel, S., Parra, A., Winter, B. and Wuest, J. (1999) Directions and Methodologies for Empirical Software Engineering Research. *Empirical Software Engineering* 4(4):405-410.

Herzberg, F. (1964) The Motivation-Hygiene Concept and Problems of Manpower *Personnel Administration* January-February:179.

Hyde, A. (2002) Information Management: Failure? Who Says? *British Computer Society Review*.
<http://www.bcs.org.uk/review02/html/p164.htm>

Khalil, O. E. M., Zawacki, R. A., Zawacki, P. A. and Selim, A. (1997) What Motivates Egyptian IS Managers and Personnel: Some Preliminary Results in *SIGCPR97*, ACM.

Krasner, H. (1997) Accumulating the Body of Evidence for the Payoff of Software Process Improvement
<http://www.utexas.edu/coe/sqi/archive/krasner/spi.pdf>

Krueger, R. A. and Casey, M. A. (2000) *Focus Groups: A Practical Guide For Applied Research*. Sage Publications.

McDermid, J.A. and Bennett, K.H. (1999) Software Engineering Research: A Critical Appraisal *IEE Proceedings on Software* **146** (4) August:179 -186.

McNurlin, B. and Sprague, R. (1996) *Information Systems Management* Chapter 18 Prentice Hall.

Morgan, D. L. and Krueger, R. A. (1993) When To Use Focus Groups And Why, in *Successful Focus Groups: Advancing The State Of The Art* (Morgan DL ed) pp 3-19, Sage.

Seaman, C. B. (1999) Qualitative Methods In Empirical Studies Of Software Engineering. *IEEE Transactions on Software Engineering* **25**(4):557-572.

Sharp, H. C., Woodman, M., Hovedon, F. and Robinson, H. (1999) The Role of 'Culture' in Successful Software Process Improvement in *Proceedings of the 25th EUROMICRO Conference* pp170-176.

Shaw I. (1999) *Qualitative Evaluation*, Sage Publications.

Warden, R. and Nicholson, I. (1995) IT Quality Initiatives at Risk *Software Quality Management* New Year (24):24-27.

Williams, L. and Cockburn, A. (2003) Agile Software Development: It's about Feedback and Change *IEEE Computer* June:39-43.

Wilson, D. and Hall, T. (1998) Perceptions of Software Quality: A Pilot Study *Software Quality Journal* March 7 (1):67-75.

Wilson, D., Hall, T. and Baddoo, N. (2001) "A Framework for Evaluation and Prediction of Software Process Improvement Success" *Journal of Systems & Software* **59**(2), pp135-142

Wohlin, C., Host, M. and Henningson, K. (2003) Empirical Research Methods in Software Engineering in *Empirical Research Methods in Software Engineering* Lecture Notes in Computer Science Springer-Verlag.

Yetton, P., Martin, A., Sharma, R. and Johnston, K. (2000) A Model of Information System Development Project Performance *Information Systems Journal* **10**:263-289.



BAM St. Andrews 2004



MENU

Conference News -
updated 15 Nov. -
includes link to
Track Best Papers

Programme -
updated 27 Aug.

Registration

Accommodation

Travel to
St Andrews - incl.
Bus from Edinburgh
Airport

LTSN
Best Paper Prize

Tracks

Activities

BAM

Join BAM (pdf)

University of
St Andrews

BAM Annual Conference 30 August - 1 September 2004 St. Andrews Management Futures

Key Dates: 2004
May/early June - Authors
notified by track chairs
11 June - Early-bird
registration ends
15 July - Final registration
30 July - Final paper
submissions

30 August - BAM
Doctoral Symposium
(Click here)

Papers and Abstracts from the Conference are now available at Proceedings.
Conference registrants have been sent the required Username and Password;
if you have not received this email, please contact us on bam04@st-andrews.ac.uk.
If you were not a conference registrant, please contact us at the same email address
where access can be given upon payment of a fee of UKP50.



£1500 has been donated
to Children in Need - a
gift made possible
through the generosity
of the participants at the
conference dinner auction

To All BAM04 Delegates,
Sponsors, Track Chairs and Helpers:
A Big Scottish Thank You
for making an excellent conference.
BAM04 Team