Software Qual J (2007) 15:365–381
DOI 10.1007/s11219-007-9028-1

Motivating developer performance to improve project outcomes in a high maturity organization

Tracy Hall · Dorota Jagielska · Nathan Baddoo

Published online: 14 August 2007

Abstract In this paper we discuss the impact software developer performance has on project outcomes. Project performance remains unreliable in the software industry with many compromised software systems reported in the press. We investigate the impact that developer performance has on aspects of project success and explore how developer performance is motivated. We present interview, focus group and questionnaire data collected from a team of developers working in a software organization that has been assessed at CMM level 5. Our main findings are that developers value technical skills in their colleagues, but appreciate these especially when supplemented with good human skills. Software developers with a proactive, flexible, adaptable approach who are prepared to share knowledge and follow good practice are said to be the best developers. Motivators for these developers are pay and benefits, recognition and opportunities for achievement in their work. Overall, we found that technical competence, interpersonal skills and adherence to good practices are thought to have the biggest impact on software project success.

Keywords Software developers · Performance · Project success · Motivation

T. Hall (✉) · N. Baddoo
Systems & Software Research Group, School of Computer Science, University of Hertfordshire, College Lane, Hatfield AL10 9AB, UK
e-mail: t.hall@herts.ac.uk

N. Baddoo
e-mail: n.baddoo@herts.ac.uk

D. Jagielska
Department of Mental Health Sciences, University College London, 67-73 Riding House Street, London W1W 7EJ, UK
e-mail: rejudja@ucl.ac.uk

1 Introduction

Project performance remains unreliable in the software industry with many compromised software systems reported in the press. Despite the increasing sophistication of technical methods and tools, outcomes from development projects generally remain disappointing. We argue that understanding the human factors in projects can complement technical advances in developing software (Hall & Wilson, 1997; Wilson & Hall, 1998; Wilson, Hall & Baddoo, 2000). Adopting such a dual approach to improving software development is likely to be more effective than previous approaches that focus solely on technical aspects (McDermit & Bennett, 1999).

In this paper we present results from the first stage of our longitudinal case study of a software project team developing a safety critical embedded system. In this study we track a range of technical and social factors over time and analyse these in terms of project progress and project outcomes. The objectives of this study are to identify the technical and social factors that have the most impact on project outcomes.

A great deal of research and development effort goes into developing improved technical environments and tools for software engineering. Although evidence showing that these technical advances actually improve project success is scarce, for example there is scare evidence showing improved software development using OO development, there remains significant effort focused on new tool and method development. However, previous empirical evidence suggests that the areas that directly relate to improved project success are focussed on good project management and requirements understanding (Verner & Evenco, 2005). These areas are fundamentally underpinned by human factors which are widely reported to be critical to project success. We propose that a dual focus on the important technical and social factors is likely to be the best approach. Indeed that there is likely to be a relationship between some technical and social factors. For example it is probably the case that developers enjoy developing and using new tools and methods, rather than there being any strong need for such developments. This 'enjoyment' may contribute to improved developer satisfaction and motivation, which may indirectly contribute to improved project success. This relationship between technical and social factors in the context of project success is also related to the emergence of agile approaches like XP, where project improvements are very much built from meeting the social needs of developers during the agile technical development process. Such agile approaches are starting to generate a body of evidence showing that they have significant impact on project success.

Our objective at this stage of the project is to understand how individual developers in this successful environment are performing and how particular developer attributes contribute to particular features of project success. In this paper we present findings on the impact of the human factors on the project. We investigate which attributes are the most valued by developers in their colleagues and themselves. We investigate how developers are motivated to exhibit these attributes. We look into which attributes they consider to have the biggest impact on the success of a project. We are also interested in organisations motivate good performance. Overall we aim to identify factors that underpin good performance.

The research questions that this paper addresses are:

RQ1: What are the attributes of a good software developer?
RQ2: Which outcomes of a project are the most related to developers' performance?
RQ3: What motivates good performance in software developers?

Software development managers should find the answers to these questions helpful in improving the outputs of software projects. A better understanding of the attributes of good developers and how the organisation can facilitate these attributes puts managers in a better position to improve the performance of the project overall. Such improved performance may contribute to reducing the software quality problems reported across the industry.

The rest of the paper is structured as follows: In Sect. 2, we present the literature context in which we ask our research questions. In Sect. 3, we describe the project and in Sect. 4 the research methodology. Results are presented in Sect. 5 and we discuss the implications of our results in Sect. 6. We summarise and conclude in Sect. 7.

2 Background

The complex factors contributing to software project success have been extensively studied (Shah, Hall, Rainer, & Baddoo, 2003). Wohlin and Andrews (2002) identify drivers to success in software engineering environments. Dyba (2000) proposes an instrument for measuring enablers to success. Verner and Evanco (2005) investigate the particular contribution that project management practices make to project success.

Overlaid on these individual contributions a variety of frameworks have been devised to evaluate success in a project. Shenhar, Tishler, Dvir, Lipovetsky, and Lechler (2002) present a generic framework on the multi-dimensions of success in projects. While others (e.g. El Emam & Birk (2000), Goldenson & Herbsleb (1995), Wohlin & Andrews (2001)) present frameworks for the evaluation of success specifically to software projects. Analysis of these evaluation frameworks reveals that 'staff productivity', 'staff morale' and 'job satisfaction' play a major role in project success. This is particularly the case in the frameworks specifically for software engineering projects (Table 1).

Generic frameworks (e.g. Shenhar et al., (2002) shown in Table 2) focus more on the customer-facing human issues than software engineering-specific frameworks. This may be as a result of the people intensive nature of developing software and the type of intense technical skills that are required. However, this developer aspect of project success seems to be an under-researched factor in work which emphasises people factors, as most work relates to users and customers. Consequently there is no clear picture of the impact of human factors associated with the non-customer facing human issues.

Table 1 Software engineering frameworks

Goldenson & Herbsleb (1995)	El Emam & Birk (2000)
Ability to meet budget commitments	Ability to meet budget commitments
Ability to meet schedule commitments	Ability to meet schedule commitments
Product quality	Ability to achieve customer satisfaction
Customer satisfaction	Ability to meet specified requirements
Staff productivity	Staff productivity
Staff morale/job satisfaction	Staff morale/job satisfaction

Table 2 Shenar et al's (2001) project success framework

Success dimensions	Success measures
Project efficiency	Met project schedule
	Stayed on budget
	Met functional requirements
	Met technical specifications
Impacts on customer	Addressed customer needs
	Solved customer's problem
	The product is used by customer
	Customer satisfaction
Business success	Commercial success
	Creating a large market share
	Creating a new market
Preparing for the future	Creating a new product line
	Developing a new technology

In this paper we discuss the impact of human factors within the development team on project outcomes. There is no doubt that a skilled team is the basis for a project's success. Turley and Bieman (1995) previously studied the performance of the most skilled individual developers. They showed that exceptionally performing software engineers considered themselves to be more proactive with management, more willing to exhibit and articulate strong convictions, more able to maintain a "big picture", are better at mastering skills and techniques and helping others more often, than non-exceptionally performing engineers. Bohem and Papaccio (1988) also reports great differences in productivity and error rates between the most and least productive software developers. We investigate what combination of these traits is considered the most desirable by developers.

Our aim is to understand whether a management environment can be 'manufactured' to optimise good developer performance. Previous studies show that software developers have a high need for personal growth and therefore are highly motivated by factors that are intrinsic to the job that they do (Couger, 1988; Fitz-Enz, 1978; Mata Toledo & Unger 1985). Additional factors that motivate software developers also include: opportunities for advancement and growth (Couger & O'Callaghan, 1994), recognition (Warden & Nicholson, 1995), increased responsibility (Couger & Adelsberger, 1988) and senior management support (Ahuja, 1999; Diaz & Sligo, 1997; Mellis, 1998; Pitterman, 2000; Willis et al., 1998). A fuller understanding of how to motivate software developers may allow projects to be more effectively managed.

3 The leds project

In this paper we present findings from our study of a software development project in a UK defence contractor company. The project develops a large complex safety critical embedded system, referred to further as LEDS, to maintain confidentiality of our industry collaborator. LEDS is a complex and novel engineering product. The overall project is multidisciplinary involving the development and integration of a range of sophisticated hardware and software.

The company considers LEDS to be a prestigious project with a high internal and external profile. Managers have set up the project to show-case the high quality work of the company. Consequently managers have carefully selected highly skilled and experienced developers for this project. The project team is made up of 10 developers who are led by a software project manager. The software team is one of four teams in the overall project. During the project the team remains part of the software department.

Software development in this company is highly mature and in April 2004 the software department was assessed as operating at CMM level 5. The department operated at level 4 for the previous 2 years. This means that these developers are well placed to identify those aspects of developer performance that contribute to project outcomes.

The software project started in 2003 and is scheduled to complete in 2008.

4 The research methods

In this study we implement a triangulated research strategy (Denzin, 1989). This approach increases confidence that we will produce reliable results with internal validity. The importance of triangulation is well-known in HCI studies (e.g. Preece, Rogers, & Sharp, 2002) and is increasingly advocated as a high quality strategy for empirical research in software engineering (Harrison et al., 1999; Seaman 1999), We apply triangulation in this study to converge our results and explore overlapping and different aspects of the same issue. Our aim is to explore contradictions that emerge and expand the scope and breadth of our study (Greene, Caracelli, & Graham, 1989).

To answer our research questions, we used focus group discussions, one-to-one interviews and questionnaires.

4.1 Qualitative data collection

In this paper we present findings based on qualitative data we collected on the project. This part of research addresses the impact of human factors on project outcomes.

We have collected the following qualitative data:

4.1.1 Interview data

We interviewed all members of the software project team. We also interviewed representatives from the other disciplines involved in the project: the hardware team, the requirements modelling team and the project management team. The interviews were designed at various levels of granularity and consequently some people were interviewed several times as we analysed hotspot issues in increasing detail. During interviews we used laddering techniques to gain detailed insights into the opinions and experiences of developers. Laddering is used in the Repertory Grid Technique (Stewart & Stewart, 1981), and is a technique we have used in previous studies (Baddoo & Hall, 2002a, b). The objective of these interviews was to identify the social determinants of project outcomes.

We interviewed all members of the software project team and representatives from the other disciplines involved in the project: the hardware team, the requirements team and the project management team. The interviews were designed at several levels of granularity

and consequently some people were interviewed several times as we analysed some issues in increasing detail. Interviews were designed to specifically address developer performance in this project. Our interviewing strategy reflected the three elements of our research question, we asked about:

- Developer performance (what makes a good developer);
- Motivation for performance (how developers have become good);
- Project consequences (the impact of good developers).

During the interviews we used laddering techniques[1] to gain detailed insights of the opinions and experiences developers were telling us about. Appendix 1 shows a first level interview script.

4.1.2 Focus groups

We ran two focus groups with six project developer participants in each group. The advantage of focus groups is that assembling groups of peers allows participants to be braver about discussing issues than they will in, for example, individual interviews (Morgan & Krueger, 1993; Krueger & Casey, 2000). All project developers participated in focus groups.

We encouraged developers to discuss various aspects of the project and facilitated discussion on the long term implications of how the project was currently performing. Again, we used laddering techniques to drill down and develop a rich understanding of the issues. We have used focus groups extensively in our previous work (Baddoo & Hall, 2002a, b; Rainer & Hall, 2002).

We followed up some of the issues uncovered in focus groups in more detail. We designed specific drilling-down interviews for specific individuals with reference to issues that had arisen. For example, the relationship between the project management team and the developers arose as an issue. In response to this we interviewed the project managers to further investigate the relationships between the two groups.

The objective of the focus groups was to explore technical and project management decisions and identify impact on project performance.

We used the broad principles of content analysis (Krippendorf, 2004) to analyse data from both the individual interviews and focus group discussions. We used content analysis principles to develop categories to code comments made by participants.

Below is a fuller explanation of the analysis procedure.

Stage 1: A manual content analysis was performed on responses in each interview and focus group. Under each question, all the responses cited were identified. A list of all the responses was then drawn up.

Stage 2: We constructed a matrix mapping responses to questions from all the interview and focus group sessions.

Stage 3: We then weighted each response for each question by frequency. These frequencies were converted into percentages. The results we report in Sect. 4 are from the responses with the highest percentages. Consequently our findings are based on strong agreement across individual interviews and focus groups.

[1] Laddering techniques are related to elements of Repertary Grid Technique (Bannister & Fransella, 1986).

4.2 Quantitative data collection—Questionnaire data

We used questionnaires to explore how widespread the views expressed during individual interviews were to the whole project team. We have used questionnaires very successfully in previous studies (Baddoo & Hall, 2002a, b).

The objective of these questionnaires was to reflect back to the whole project team those issues expressed by individuals. The intention is to measure the significance of potential determinants of project performance. These data have been previously presented (Baddoo, Hall, & Jagielska, 2005) and will be mentioned here only as a point of reference for the qualitative data.

5 Results

5.1 Developers' performance

The interviews showed that the software team believes they are highly technically competent. During interviews the whole software team showed considerable confidence in the high quality of their work. They were very proud of attaining CMM level 5 and believed it accurately represented the high quality work they were doing. Their confidence seems to be based on their success in previous projects and the stated confidence managers have in their work. The fact that developers knew they had been selected for the project on the basis of the quality of their work boosted their confidence significantly.

An analysis of the data collected from the interviews shows the following most important perceived attributes of a high performance developer:

5.1.1 Technical competence

Technical ability was absolutely taken to underpin the performance of a developer. Technical competence was highly regarded and respect for the best technicians was very strong. Some developers were almost held in awe for their technical abilities. Technical ability seemed to be a measure of status within the team. One developer held a position of team guru whom the rest of the team used as technical trouble-shooter. This person was regarded by the team as very high performance developer. He had a wide understanding of the domain combined with strong technical knowledge and experience of the hardware and software in use.

5.1.2 Technical confidence

Developers valued technical confidence in high performing developers. They valued the ability of other developers to suggest a technical solution and then promote this solution in the face of competing alternatives. They did not value developers who were defensive of their work and who hid problems. They said that good developers worked in a transparent way, acknowledged problems early and took responsibility for fixing them. However developers identified a boundary between technical confidence and arrogance. Many developers said that over-confidence leads to arrogance. Developers emphasised the importance of respecting the ability of others—even when promoting a 'best' solution.

5.1.3 Seeing the bigger picture

Developers highly rated the ability of others to 'see the bigger picture'. Getting too caught-up in tiny details that ultimately do not matter was seen by many as a problem with some developers.

5.1.4 Flexibility

Developers valued other developers with flexible and adaptable attitudes. They wanted to work with developers who were willing to listen, learn and change. Some developers felt that an inflexible attitude to change meant that the Department did not always get the full benefit of process improvement.

5.1.5 Communicators

Developers who are good communicators and who are willing to communicate are perceived as good developers. Good communication is seen as a characteristic of good team work that helps the team to gel. Communication is also valued as a tool for providing feedback on work to the rest of the team. *Articulate*, *good conversationalist* and *listens to others* are some of the terms used to describe a good developer. Developers are valued when they can articulate technical and non-technical issues succinctly and who are willing to share knowledge.

5.1.6 Team worker

Developers valued the personal attributes of others such as integration with the team and being a 'nice' thoughtful person. It was important to be an amiable colleague who is easy to work with.

Interviews also revealed the high expectations developers have of project team members in the software team and in the other project disciplines. Developers displayed a ruthless streak when discussing poor performance in others. They were acutely aware that poor performance in any part of the project will impact on the overall performance of the project. Developers defended the software team's high performance status ruthlessly.

5.1.7 Follow-up

Combining the interview results with literature sources we constructed a list of 14 attributes of a good software developer. We asked respondents to rate how important each of these attributes is in terms of being a good developer, with aim of assessing how widespread these views are in the team. Detailed analysis of these data was presented in (Baddoo et al., 2005). The developer's attributes rated the most important by all respondents were: being *proactive, flexible and adaptable, fully documenting work* and *sharing knowledge with the team*. Attributes considered slightly less important include: *innovativeness, resolving complex problems, self-dependence*, *communicating well with stakeholders* and *high technical competence*. Even less important such attributes as: *eager*

to try new technology, adhering to the process and *being driven by the work itself*. They rated two attributes neutral to being a good developer: *strong knowledge of the problem domain* and *being prepared to work long hours*. These results are surprising as they suggest that the social skills of developers are valued more than the technical skills. Though all are important, they are not equally important. What these practitioners report is in line with much of what the literature says and so it is surprising that progress in these areas has not been more substantial.

5.2 Project outcomes and developer performance

During the focus groups the following project outcomes were raised as being related to developer performance:

5.2.1 Quality

Developers said that this was as a result of having good communicators who are prepared to defend the best technical solutions. This is also a result of having technically competent developers who would get the design of the solution right in the first place. Also, when developers are flexible and are prepared to consider other alternatives, then it results in the best solution being adopted as the best ideas are presented and defended.

5.2.2 Progress

Having self-driven developers who have high expectations of themselves means that they are prepared to get a lot of work done and make a lot of progress. This in turn drives on the people they work with.

5.2.3 Customer relationship

Many issues were raised as related to effectively modelling requirements. There was a strong feeling that there are issues related to communicating with the customer. In particular that channels of communication were not clearly identified and communication was regularly slow and opaque. Developers are also acutely aware that ineffective communication with customers is a well-reported factor in project failure.

Using questionnaires, we asked the team which one from the previously identified traits of good developers have the biggest impact on project success. Three attributes were considered to contribute strongly to all aspects of project success and these were: *fully documents their work, shares knowledge with the team* and *has high technical competence.* Almost as influential on project success were thought to be: *having strong knowledge of the problem domain, being proactive* and *resolving complex problems.* Developers also indicated that the ability to *communicate well withstakeholders* significantly improves customer satisfaction. Respondents identified a group of developer attributes that, according to them, do not contribute strongly to project performance. These were: *self-dependent, driven by the work, eager to try new technology* and *prepared to work long hours*. Detailed analysis of these results are reported in (Baddoo et al., 2005).

5.3 Factors motivating good performance

During interviews the following issues emerged as the main ways in which good performance is motivated.

5.3.1 Rewarding performance

Developers were adamant that motivating good performance is related to rewarding 'the right things'. The attributes of good developers described in 5.1 above were those attributes developers wanted to see rewarded by the company. Developers believed that it was those good attributes that determined project success and therefore should be rewarded.

Some developers thought that managers could miss the critical contributions of some high performance developers as good work was not always highly visible. They said that managers sometimes reward superficially visible performance which did not always contribute to project success. Developers displayed a certain amount of tension regarding how job performance was rewarded. Indeed some said that project progress could be undermined by personal ambition and a desire to visibly demonstrate performance. Developers believed that managers can be seduced by developers who are 'too eager to please'.

5.3.2 Appropriate task-fit

Developers said that an important way to motivate good performance is to ensure good task-fit. Matching tasks to individual developers was seen as a major performance motivator. Developers said that task-fit should account for not only the particular strengths and interests of an individual but also account for the level of challenge that individuals responded best too. Some developers were motivated by very challenging and complex technical problems while others were said to be worried and de-motivated by that level of challenge. Furthermore developers also believe that inappropriate task-fit compromises the outcome of the project.

5.3.3 Team quality

Developers reported that high individual performance is motivated by working in a high performance team. Developers were very proud to be working in a highly regarded team and developers were very aware that they were on a show-case project. They knew that the company had very high expectations of them and they were committed to exceeding those expectations. In addition there was a certain amount of competition to perform well amongst team members. No one wanted to be the weak link in this team. However everyone said they benefited from the good team dynamics that existed. The working relationship between team members was universally said to be excellent and very motivating.

5.3.4 Good development tools

Developers were highly excited by the implementation of a new toolset for this project. The new toolset had been thoughtfully selected and implemented exceptionally well. The

developers were universal in saying how much easier it was to work with this new toolset. It was clear from interviews that the new toolset was highly effective at facilitating good work but also that developers really enjoyed using it. The toolset was perceived by developers to be state-of-the-art. This fitted in with their self-image and reinforced them feeling valued by the company. Developers also hinted that working with such a state-of-the-art toolset kept them up-to-date and enhanced their CV. However developers are convinced that the new toolset is contributing to the quality of the project so far and are determined to make the toolset a success. The importance of having high quality development resources readily available to facilitate developers to do their best work is confirmed by scenario 2. It shows the impact on developers' morale of having project progress impeded by 'raw material problems'.

5.3.5 Communication

Developers believed that performance could be motivated by good project communication. They valued open and transparent communication channels throughout all discipline teams on the project. Furthermore they were clear that effective communication from project management to low level developers was important. They felt that open communication channels can effectively allow low level input into the high level co-ordination and scheduling of complex technical tasks. An open communication culture was described where the 'truth' can be voiced and accepted positively. Such an approach was seen as benefiting the whole project rather than just being a software specific issue.

This question also was followed with questionnaires, listing motivators derived from literature and on the basis of interviews with the team. This allowed us to judge how widespread the motivating power of each motivator was, at the same time the follow-up questionnaire allowed us to compare the motivational profile of our respondents with what is reported in the literature.

Respondents considered *pay and benefits*, *recognition* and *opportunity for achievement* the strongest motivators for their own performance. Respondents generally valued all of the other motivators similarly and no motivator appears to be unimportant. The next in line to motivate their performance were *job security, technically challenging work* and *senior management support.* These were followed by: *opportunities for promotion, increased responsibility, technical support/supervision, company policy, autonomy, a sense of ownership* and *work conditions* (further analysis of this data in (Baddoo et al., 2005)). The importance of Pay and Benefits does not correspond to the motivators reported in the literature. Indeed pay and benefits is thought not to particularly motivate IT professionals. This result may be a result of the self reporting nature of a questionnaire, where respondents may believe or say that this motivates them, but in reality their actual behaviour is not affected by pay and benefits. This apparently anomalous result is discussed in more detail in the next section.

5.4 Summary of our findings

Table 3 summarises our main findings relating to performance attributes, motivation and project outcomes.

We will discuss the implications of our findings in the next section.

Table 3 Main findings

Developer performance attributes	Motivation factors	Project outcomes
Technical competence	Rewarding performance	Quality
Technical confidence	Good task fit	Progress
Seeing the bigger picture	Team quality	Customer relationship
Flexibility	Communication	
Communication		
Team worker		

6 Discussion

In this section we use the results presented in the previous section to answer the research questions introduced at the beginning of this paper.

6.1 RQ1: What are the attributes of a good software developer?

During focus groups and interviews technical competence was declared the most important attribute of a good, high performing developer. Developers affirmed that technical competence must underpin any other performance factor. This is very much in agreement with the findings of Turley and Bieman (1995) in their study of exceptional developers. 'Soft' skills, such as flexibility, good communication skills or team working skills were also valued by developers, but took a less prominent place in discussions and did not seem to be as highly appreciated as technical skills.

However when we presented them their own opinions gathered in a form of a questionnaire items, the team rated other skills higher then technical competence. These were being *proactive, flexible and adaptable, fully documenting work* and *sharing knowledge with the team*. This might be related to the CMM level 5 development environment that these respondents are working in. Such a high maturity environment must have control over the technical processes of software development. Therefore it may be that the non-technical issues that are outside the scope of the CMM are now the most valued attributes in this environment as it is these that are under less direct control. However, the developers seem to need the prompt of a questionnaire to cite the soft skills of their colleagues as important, but when asked unprompted during interviews the technical skills came to their mind first.

Our findings are consistent with those of Turley and Bieman (1995), who report that 'exceptional practitioners' characterise their own performance as highly proactive and eager to help others, which relates to *sharing knowledge with the team* in our study.

Our findings also confirm that being technically skilled is not enough in today's working environment. The complexity of systems being designed, which requires many people working together on one goal, adds to the list of important success factors coming from outside the technical domain, to name just a couple: customer / user involvement, or executive support etc. (The Standish Group, 1999). This means, that high performing developers must complement his or her technical skills with human ones.

6.2 RQ2: Which outcomes of a project are the most related to developers' performance

Developers identified three project outcomes which they believed to be highly related to developer's performance. These were quality, progress and customer relationship. These outcomes are highly related to the frameworks of project success introduced in Section Two. All three of these issues appear in some form in the three frameworks previously discussed. Furthermore, unlike the data we present, the three frameworks discussed do not include any ranking of factors. This may explain the limited number of factors we present as these are only the most important factors.

The attributes which we found had most impact on project outcomes are: *high technical competence, fully documenting work* and *sharing knowledge with the team*. These skills can be divided into three groups: technical competence, following good practice and interpersonal skills. Skills from each of these groups add positively to project outcomes. This suggests that it is essential that companies support and develop as aspects of a developer's skill profile, rather than just focusing on the technical skills.

6.3 RQ3: What motivates good performance in software developers?

We found that there are a number of important management behaviours that motivate good performance in software developers. The most important of which are: rewarding the right behaviour, appropriately fitting tasks to people and good team infrastructure. In terms of the impact of these issues on project performance these issues impacted on technical work quality, progress in the project and relationships with the customer.

These results may provide more evidence of a shift in the profile of software developer motivators. Works done during the 1980s found that software developers were primarily motivated by the intrinsic aspects of the work they do (e.g. the technical challenge of the work). Our results suggest that the importance of such intrinsic motivators may be diminishing. Indeed in 1994, Couger and O'Callaghan (1994) reported that software practitioners in the US and Europe consistently ranked pay amongst their top four motivators.

Furthermore, even in open source projects extrinsic motivation in the form of the promise of monetary rewards in the future seems to play a significant role. Hertel, Niedner, and Herrmann, (2003) internet survey of contributors to the Linux kernel identified an interest in building a reputation to support a future career as an important motivator. Hars and Ou's (2001) study of the motivation of open source software developers found that the majority of them indicated improving their human capital (77%) and self-marketing (50%) were very important reasons for engaging in open source projects. These results suggest that the motivation profile of software developers may be becoming increasingly similar to the profiles of others professions.

6.4 Role of communication on the project

Although it was not an aim of this study to analyse the role of communication on a project, the results of this study concerning it seem to deserve a special mentioning. Communication is said to influence the project outcomes in two-fold way: directly as a factor shaping the relationship with customer, and also indirectly, as a factor influencing developers' motivation.

7 Summary & conclusions

This paper presents preliminary findings characterising the relationship between developer performance, performance motivation and project performance. *High technical performance* is undoubtedly the core attribute of a good developer. However, interpersonal traits, such as sharing knowledge, being proactive, flexible and adaptable, and also following good practice by fully documenting work are also highly valued by practitioners and regarded as important for developing into an exceptionally performing practitioner. Currently these are developers' perceptions and if these links could be confirmed on a larger scale, this would be invaluable information for managers of software projects to use in targeting staff development effort and their project management.

We also found out that, at least in this particular company, the motivation profile of developers is different from that described in the literature. Extrinsic motivators, such as pay and recognition, appropriately fitting tasks to people and a good team infrastructure are starting to play more important roles than the traditional intrinsic motivators relating to the work itself.

Our findings stress the importance of non-technical skills among developers. Understanding and managing these non-technical project factors may be especially important in high maturity teams where the technical issues are under process control. Hopefully further work in this area will provide insights that can allow managers to understand how to assemble a team with the right profile of developer attributes for the particular project being delivered and ensure project success.

Appendix 1. Research instrument scripts

Individual interview script

1. Thinking about the best developer on this project….
   - What is the best thing about them?
   - What else is good about them?
   - Why do they have those good qualities?
   - How do they contribute well to the project?
2. Thinking about the worst developer on this project….
   - What is the worst thing about them?
   - What else is bad about them?
   - Why do they have those bad qualities?
   - How do they contribute badly to the project?
3. What is good about you as a developer?
   - What is the best thing you do?
   - What else do you do well?
   - Why do you have those good qualities?
   - How do those good qualities contribute to the project?
4. What is bad about you as a developer?
   - What is the thing you do least well?
   - What else do you not do so well?

- Why do you have those bad qualities?
- How do those bad qualities contribute to the project?

Focus group structure

1. Thinking about the project highs, what is working really well in this project….
   - What is this best thing about this project?
   - What are the other good things about this project?
   - Why are these so positive?
   - How have these positive aspects of the project been established?
   - How does this impact on project outcomes?
   - Who are the key players in this positive aspect of the project?
2. Thinking about the project lows, what is not working so well in this project….
   - What is this worst aspect of this project?
   - What else is negative about this project?
   - Why is this so negative?
   - How have these negative aspects of the project been established?
   - How do these impact on project outcomes?
   - Who are the key players in this negative aspect of the project?

References

Ahuja, S. (1999). Process improvement in a rapidly changing business and technical environment. Fourth annual European process group conference. Amsterdam, Holland. c303.

Baddoo, N., Hall, T., & Jagielska, D. (2005). Software developer motivation in a high maturity company: A case study. Proceedings of European software process improvement and innovation conference, 9–11 November 2005, Budapest, Hungary.

Baddoo, N., & Hall, T. (2002a). Motivators of software process improvement: An analysis of practitioners' views. *Journal of Systems & Software, 62*(2), 85–96.

Baddoo, N., & Hall, T. (2002b). Practitioner roles in software process improvement: An analysis using grid technique. *Journal of Software Process Improvement and Practice, 7*(1), 17–31.

Bannister, D., & Fransella, F., (1986). *Inquiring Man: The Psychology Of Personal Constructs* (3rd ed.). London: Croom Helm

Bohem, B., & Papaccio, P. (1988). Understanding and controlling software costs. *IEEE Transactions on Software Engineering, 14*(10), 1462–1477.

Couger, J. D., & 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. (1988). Motivators and demotivators in the IS environment. *Journal of Systems Management, 39*(6), 36–41.

Couger, J. D., & O'Callaghan, R. (1994). Comparing the motivators of Spanish and finish computer personnel with those of the United States. *European Journal of Information Systems, 3*(4), 258–291.

Denzin, N. (1989). *The research act: A theoretical introduction to sociological methods*. NJ, Prentice-Hall: Englewood Cliffs.

Dyba, T., & Dyba, T. (1997). How software process improvement helped motorola. *IEEE Software Archive, 14*(5) 75–81.

Dyba, T. (2000). An instrument for measuring the key factors of success in software process improvement. *Empirical Software Engineering, 5*(4), 357–390.

El Emam, K., & Birk, A. (2000). Validating the ISO/IEC 15504 measure of software requirements analysis process capability. *IEEE Trans on Software Engineering, 26*(6), 541–566.

Fitz-Enz, J. (1978). Who is the DP professional? Datamation September:125–128.

Greene, J. C., Caracelli, V. J., & Graham, W. F. (1989). Towards a conceptual framework for mixed method evaluation designs. *Education Evaluation and Policy Analysis, 11*(3), 255–274.

Goldenson, D. R., & Herbsleb, J. D., (1995). After The Appraisal: A Systematic Survey of Process Improvement, its Benefits and Factors that Influence Success, Software Engineering Institute, CMU/SEI-95-TR-009 ADA302225. http://www.sei.cmu.edu/publications/documents/95.reports/95.tr.009.html

Hall, T., & Wilson, D. (1997). Views of software quality: A field report. IEE Process on Software Engineering, Apr, 111–118.

Harrison, R., Baddoo, N., Barry, E., Biffl, S., Parra, A., Winter, B., & Wuest, J. (1999). Directions and methodologies for empirical software engineering research. *Empirical Software Engineering, 4*(4).

Hars, A., & Ou, S. (2001). Working for free?–motivations of participating in open source projects, *Proceedings of the 34th Hawaii International Conference on System Sciences*.

Hertel, G., Niedner, S., & Herrmann, S. (2003). Motivation of software developers in open source projects: An Internet-based survey of contributors to the Linux kernel. *Research Policy, 32*(7), 1159–1177.

Krippendorf, K. (2004). *Content analysis, an introduction to its methodology*. Thousand Oaks-London-New Delhi: SAGE Publication.

Krueger, R. A., & Casey, M. A. (2000). Focus groups: A practical guide for applied research. Thousand Oaks,CA: Sage Publications.

Mata Toledo, R. A., & Unger, E. A. (1985). Another look at motivating data processing professionals. *Computer-Personnel, 10*(1), 1–7.

McDermit, J. A., & Bennett, K. H. (1999). Software engineering research: A critical appraisal. *IEEE Proceedings—Software, 146*(4), 179–186.

Mellis, W. (1998). Software quality managment in turbulent times—are there alternatives to process oriented software quality management? *Software Quality Journal 7*, 277–295.

Morgan, D. L., & Krueger, R. A. (1993). When to use focus groups and why. In D. L. Morgan (Ed.), *Successful focus groups: Advancing the state of the art*. Newbury Park, CA: Sage.

Pitterman, B. (2000). Telcordia technologies: The journey to high maturity. *IEEE Software, 17*(4), 89–96.

Preece, J., Rogers, Y., & Sharp, H. (2002). *Interaction design*, NY: John Wiley.

Rainer, A., & Hall, T. (2002). Key success factors for implementing software process improvement: A maturity-based analysis. *Journal of Systems & Software, 62*(2), 71–84.

Seaman, C. (1999). Qualitative methods in empirical stuides of software engineering. *IEEE Transactions of Software Engineering, 25*(4).

Shah, M., Hall, T., Rainer, A., & Baddoo, N. (2003). Software engineering projects: How to evaluate success? University of Hertfordshire Technical Report, Faculty of Engineering & Information Science, number 394.

Shenhar, A., Tishler, A., Dvir, D., Lipovetsky, S, & Lechler T. (2002). Refining the search for project success factors: A multivariate, typological approach. *Research & Development Management, 32*(2), 111–126.

The Standish Group International Inc. (Ed.), *Chaos: Recipe for Success (1999)*, source: http://www.standishgroup.com.

Stewart, V., Stewart, A., & Fonda, N. (1981). *Business applications of repertory grid*. London: McGraw Hill.

Turley, R. T., & Bieman, J. M. (1995). Competencies of exceptional and nonexceptional software engineers, *Journal of Systems and Software, 28*(2).

Warden, R., & Nicholson, I. (1995). IT quality initiatives at risk. *Software Quality Management New Year, 24*, 24–27.

Willis, R. R., Rova, R. M., Scott, M. D., Johnson, M. I., Ryskowski, J. F., Moon, J. A., Shumate, K. C., & Winfield, T. O. (1998). Hughes aircraft's widespread deployment of a continuously improving software process. Software Engineering Institute, Canergie Mellon University.

Wilson, D., & Hall, T. (1998). Perceptions of software quality: A pilot study. *Software Quality Journal,7*(1), 67–75.

Wilson, D., Hall, T., & Baddoo, N. (2000). The software process improvement paradox. Approaches To Quality Management, *Software Quality Management* VIII:97–101.

Verner, J., & Evanco, W. (2005). In-house software development: What project management practices lead to success?. *IEEE Software, 22*(1), 86–93.

Wohlin, C., & Andrews, A. (2002). Analysing primary and lower order project success drivers. *Proceedings of International conference in Software Engineering and Knowledge Engineering*, 393–400.

Wohlin, C., & Andrews, A. (2001). Assessing project success using subjective evaluation factors. *Software Quality Journal, 9*, 43–70.

Author Biographies

Tracy Hall is Head of the Systems & Software Research Group in the School of Computer Science at the University of Hertfordshire. Dr Hall's expertise is in Empirical Software Engineering research. Over the last 15 years she has conducted many empirical software engineering studies with a variety of industrial collaborators. She has published nearly 30 international journal papers and nearly 50 international conference papers. Dr Hall is a member of the Software Quality Journal's Editorial Board and a member of the programme committee for the The IEEE International Conference on Empirical Software Engineering.

Dorota Jagielska is a researcher with the Mental Health Sciences Department of University College London. She was previously a researcher in the Systems and Software Research group in the School of Computer Science at the University of Hertfordshire. In 2001 Dorota obtained a Masters degree in Philosophy from the University of Gdansk, Poland, and followed it by a Masters degree in Psychology from the same University in 2004. Her main research interests are human factors in software engineering, especially the role of communication within software teams.

Nathan Baddoo is a Senior Lecturer in the School of Computer Science at the University of Hertfordshire. He is a member of the Systems and Software Research group at the University of Hertfordshire. His research focuses on the relationship between developer motivation and software quality, software process improvement and software project performance. Dr. Baddoo has expertise in focus group discussions and Repertory Grid Technique interviews, and has applied novel data collection and analysis techniques such as Multi Dimensional Scaling.