Does the XP environment meet the motivational needs of the software developer? An empirical study

Sarah Beecham[1], Helen Sharp[2], Nathan Baddoo[1], Tracy Hall[1] and Hugh Robinson[2]

[1]*School of Computer Science, University of Hertfordshire, College Lane, Hatfield, Herts AL10 9AB, UK*
[2]*Dept of Computing, Faculty of Mathematics and Computing, The Open University, Walton Hall, MK7 6AA, UK*
({s.beecham; n.baddoo; t.hall}@herts.ac.uk) ({h.c.sharp; h.m.robinson}@open.ac.uk)

Abstract

This paper examines how XP practice meets the motivational needs of software developers. Interactions with peers have been identified by others as one potential area of (de)motivation but little detail is known. The nature of this motivator, as expressed by software developers themselves, was explored through semi-structured interviews with a high maturity high performing team working on safety critical software applications in a traditional environment. From these interviews, we have identified seven themes which are characteristic indicators of peer motivation. We interrogate observational data from five mature XP teams to consider whether and how these characteristic indicators are present in an XP environment. We find that XP teams in our study had processes in place that supported many of the motivational needs voiced by developers coming from a traditional, heavyweight software development environment. However, the XP environment is at odds with other motivational needs.

1. Introduction

Motivation is a complex phenomenon with many inter-related, context-dependent factors [22]. Motivation refers to the initiation, direction, intensity and persistence of behaviour and features in one of the Agile manifesto principles, "build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done" http://www.agilemanifesto.org. Motivation is acknowledged to have a major impact on the quality and productivity of the software product [1-3], yet as it is a soft factor, and difficult to quantify, it often takes a backseat [1].

In this empirical study we explore peer interactions, understood to have a major impact on software engineers' motivation [3-7], and the conditions around these interactions that encourage or discourage motivational behaviour. Our analysis compares the XP environment as represented by observational studies of five mature XP teams, with the preferred conditions for high motivation as stated by practitioners in a traditional development environment.

Several papers have been published that endorse the XP environment as being a preferred environment e.g. [8-10], that gives greater job satisfaction [11, 12], reduces the software development cost [13] and increases code quality [14]. Yet no study has placed the known and accepted XP values and characteristics directly in line with developer preferences coming from a traditional background, to show where needs are met and where values are challenged or in some cases are missing altogether.

In order to consider how the XP environment compares to preferred conditions for high motivation, we look at data from interviews with software engineers (SEs) involved in a high-profile, safety critical project. The SEs are all highly motivated [15] having been selected specifically for their previous high quality performance in this CMM level 5 organisation. Our questions were aimed at uncovering what characterises a motivating team member and what characterises a de-motivating team member. We use this motivating/de-motivating team member data to interrogate observational data collected in five mature XP teams. In this way we empirically compare similarities and differences between perceptions of traditional software engineers and observed practice of XP teams in terms of peer interaction.

This paper is organised as follows. In section two we give a background to motivation in software engineering as identified in the literature, including studies which have looked specifically at XP and Agile environments. In section three we explain our methodology that involves case studies in traditional and XP environments. Section four presents our results relating to software engineer needs in a traditional environment and how this compares to the XP

AGILE 2007
0-7695-2872-4/07 $25.00

environment. Section five discusses the strengths and weaknesses of the XP environment in providing the conditions conducive to motivating software engineers as stated by traditional developers. Limitations to this work are discussed in Section six, and in Section seven, we give our conclusions and suggestions for future work.

2. Background

Here, we discuss literature regarding motivation in software engineering and Agile or XP teams.

2. 1 Motivation in Software Engineering

Motivation in Software Engineering is reported to have the single largest impact on practitioner productivity [2] and software quality management [16], and continues to be 'undermined' and problematic to manage [3]. Motivation is increasingly cited as a particularly pernicious people problem in Software Engineering. In DeMarco and Lister's [17] survey, motivation was found to be one of the most frequently cited causes of software development project failure. The Standish report [18] amplifies this finding by reporting that having access to competent, hard working and focused staff is one of ten success criteria for software projects.

Some studies in this area suggest that conventional approaches to motivation within the industry might be outdated. They have concentrated on rewards and recognition, e.g. [19], whereas some experts have identified Software Engineers as having a distinctive personality profile [20] that are instead motivated by the nature of the job, e.g. technical success, challenging technical problems [5, 21] and peer interaction [3-7]. In our systematic review of the literature on motivation in software engineering 1980 – 2006 [22] we found an increasing awareness of the importance of motivating software engineers.

Several papers in the literature cite factors relevant to the relationships between team members as being motivation factors for software engineering. For example, the importance of a supervisor being a good team builder; the developers preference for collaborative work, the need to belong and fit in with a group [23] which is physically close [24]. The importance of supportive relationships, team identification and dynamics is also a recurring theme [3-5, 7, 25].

Career path, defined as the opportunity for advancement, promotion prospects, and career planning within an organisation, is also an important motivator. Important motivators in this category include, promotion prospect/opportunity for professional advancement [23, 26-29], 30, 31] for the software developer is often characterised as having high growth needs [32-36].

Some motivation factors are reported as being motivating by some studies and de-motivating by others, for example, change and the maintenance task. It may be possible to account for this by considering the different context in which the software is being developed. However, little work on motivation has focused on the specific nature of software engineering itself, or of the impact of the changing environment in which software engineering is conducted.

2.2 Motivation and XP

Published studies that relate motivation or job satisfaction to Agile and XP emphasise the positive aspects of these approaches. For example, a longitudinal study by Syed-Abdullah et al. [9] revealed that XP methodology has a positive impact on an individual's disposition to be happy, across time and situations. This finding is supported in a comparative study, where job satisfaction was found to be higher in developers using XP practices as opposed to developers not using XP practices [10]. Practitioners showed a strong preference for working in an XP environment, using XP practices.

Continuing the positive theme, Melnik and Maurer [12] empirically compared job satisfaction in Agile and Non-Agile Software development teams finding that the greater the experience of working in an Agile environment the greater the job satisfaction. Melnik and Maurer study is particularly helpful in linking job satisfaction to motivation, ".. the benefits of higher job satisfaction mentioned have been: … increased individual team morale, motivation, performance productivity and retention." The importance of motivation and teamwork is further explored in Asproni [11] who explains how Agile development methods contain the necessary ingredients to motivate developers to make effective teamwork possible.

Coram and Bohner [37], take a slightly different view, indicating that perhaps the largest impact of Agile Methods is on the developers, stating that Agile Methods depend on strong developers that must be amicable, talented, skilled and able to communicate well (in [38]). They must be willing to work as a team, able to handle constant change, and resourceful enough to solve problems. Agile Methods are very lightweight methods, not affording strict guidelines and processes for developers to follow. Hence, they do not accommodate weaker developers well. In this theoretical study, Coram and Bohner [37] touch on the

difficulty that some developers might have coming from a traditional background, "Given the need for a high level of expertise, Agile Methods may be difficult to employ in a traditionally staffed organization." They add that it might be difficult to build a long term human capital strategy where skilled staff are always in demand.

Law and Charron [8] also reflected on some of the benefits and disadvantages of XP. For example they found pair programming motivating in two separate projects because it addressed the need for learning, autonomy and social activity. However, Law and Charron (2005) do accept that pairing can have a negative impact if pairs have personality conflicts and that the project might suffer from autonomy where less interesting work might not be tackled immediately.

3. Methodology

To compare similarities and differences in developer perceptions of what constitutes a motivating group member and team environment we draw on two empirical data sets. The first set comprises transcripts of a series of nine interviews with software developers working in a traditional development environment. The second is a set of observational data from five mature XP teams. Our overall approach is to analyse the interview responses to gain further insight into the kind of supportive relationships the developers find motivating, and then to use these findings to interrogate the observational XP data. All data was collected and analysed by the authors.

3.1 Empirical data sets

3.1.1 Semi-structured interview data from a traditional software development team. The findings presented are from nine semi-structured interviews conducted in July 2004. The interviewees were developers working in a large complex embedded software development project (LEDS) based in a large UK engineering company. The company considers LEDS to be a prestigious, high profile project. LEDS has been set up to showcase the high quality work of the company and is composed of several disciplines including a hardware team, a requirements modelling team, and a software development team. The project is managed by a dedicated project management team.

The software development team was a member of the company's Software Department. The Department was assessed at CMM level 5 in 2004. The processes used by the Department were therefore of high maturity. The LEDS software team had been hand-picked by the Department's managers to ensure a high quality outcome [15].

We interviewed all members of the software project team. One week was spent on site where one hour, one-to-one semi-structured interviews were conducted. The interview questions reflected three elements:

- Developer performance (what are the attributes of a good and bad developer)
- Motivation for performance (what encourages developers to be good or bad)
- Consequences (how does a good and bad developer impact the project/team).

Developers were not asked directly about motivation, but analysing their answers to questions relating to good experiences and bad experiences of working with colleagues on this project reveals issues that affect their relationships with fellow team members, and hence may affect their motivation, as suggested by literature reported in section 2.1 above. For example, issues around progress were mentioned the most when discussing both the "best" developer and the "worst" developer. This indicates that progress is a significant concern of the developers when considering peer interaction.

3.1.2 Observational data from mature XP teams. Our observational data arises from ethnographically-informed studies [39] of five mature XP teams. These teams are 'mature' in that they have been practising XP for at least a year (collectively or individually) before we have studied them. Each study lasted at least a week. Our observations focused on the interactions between team members; the data collected consisted of contemporaneous notes, photographs and some audio recordings. For the purposes of anonymity, we refer to these teams as Teams B, C, K, S, and W.

Team B (composed of 12 developers and a business analyst) produced software applications in Java to support the management of operational risk within a large bank. The application was being developed in an environment of uncertainty as policy pertaining to operational risk had not yet been issued. XP within the company was championed by a senior manager.

Team C (composed of eight developers, one graphic designer and one infrastructure specialist) developed web-based intelligent advertisements using Java. Although they had several strong clients, the company was working hard to satisfy its investors. There was no evidence to suggest that the team members would feel that their jobs were threatened.

Team K (composed of 23 developers, one project manager and two business development staff)

developed and maintained travel information webpages and travel alerts for a variety of customers in the UK. The company was doing well and expanding its workforce at the time of the study.

Team S (composed of six developers, one project manager, three business analysts, and one domain expert) worked in a large international bank, and programmed in Java. Their project concerned the migration of database information from several smaller databases to one large database. All of the developers and business analysts were on short-term contracts. This was common within this bank, and the individuals were used to this kind of arrangement.

Team W (composed of 16 developers, three programme managers, two testers, one technical author, and one development team coach) were part of a medium-sized company producing software products in C++ to support the use of documents in multi-authored work environments. The company had moved to XP because it was failing to sell its products. Business performance had since improved and the threat of redundancies had faded.

3.2 Analysis

3.2.1 Analysis of traditional development team data. Digital audio tapes of the interviews were listened to and detailed notes taken at the time of the interviews were examined. We recorded developer responses to what constitutes the 'best' and 'worst' developer on the project in terms of performance. We also recorded how good and bad characteristics were encouraged, and how these developers impacted the project. We identified recurring themes in participant answers, and using a content analysis classification scheme [50] created a reduced list of categories as shown in tables 1 and 2. Content analysis is defined as "objective, systematic, and quantitative description of the manifest content of communication" [51]. Our classification scheme involved two researchers. One researcher identified an initial set of developer attributes, which a second researcher looked at along with the associated definitions. The second researcher validated the identified themes by querying any ambiguities. The final list of attributes given in tables 1 and 2 is a result of refining and combining the initial list to the satisfaction of these two independent researchers.

3.2.2 Analysis of XP observational data. Tables 1 and 2 were analysed to identify characteristic indicators of motivational peer behaviour, and conditions which affected the presence of these characteristics. We applied a thematic approach to produce cross-cutting themes that can be viewed as both negative and positive influences on individual motivation. These themes (see Table 3) were used as prompts with which to explore the XP observational data.

Analysis of the observational data followed a rigorous approach based on ethnographic principles. In this approach, each theme from Table 3 was used to interrogate the observational data to see if any recurring patterns emerged regarding the issue being addressed. Throughout this interrogation, we looked for disconfirming as well as confirming evidence to support our conclusions. For example, one of the findings from the interview data suggested that a 'poor developer' does not admit to having a problem in a piece of software. We took this finding and looked through the observational data to see if there were any occasions when we observed any developers hiding problems. We could not find any examples of developers hiding problems across all XP teams, on the contrary, we found several examples of developers communicating problems. Our conclusions were then based on all the evidence related to this issue.

4. Results

4.1 Traditional development team results

Tables 1 and 2 reflect the characteristics of an environment that traditional developers believe will encourage developers to be 'good' (best) or to be 'bad' (worst). In this context, 'good' means 'people they want to work with'. For example, an environment with a very open culture leads to quality solutions and better flow of information among the team. This in turn encourages developers to develop as good communicators. The tables give a definition of the attribute (taken directly from the interview data), and how this attribute is encouraged (in 'helped by' column); and the impact of this attribute (in 'outcome' column).

Table 3 is the result of identifying cross-cutting themes from the interview data where the following themes emerged: access to information, fear/ insecurity/confidence, openness/communication, progress, responsibility/ autonomy/ ownership, software quality, team morale.

TABLE 1: Attributes of best developer on the project, how attribute is encouraged and outcome			
Attribute (desired)	**Defined as (from data)**	**Helped by**	**outcome**
Technically competent /knowledgeable/ experienced	Good knowledge of programming language, software paradigm and tools. Wide knowledge of software as well as hardware. Looks at the bigger picture.	Tools, information, Individual interest Training. Motivation.	Supportive role. Project success Lots done and progress made.
Good communicator/ sociable/ open /sharing /respectful	Articulate –teaches team tool use and techniques; says what is going right what is going wrong	Team dynamics. Open culture. Truth can be voiced	Quality of solution improved. Better flow of information
Confident	Faith in own abilities. Self dependent	Encouragement. Opportunity to show what they can do	Improves quality
Motivated	Driven/motivated. Doesn't need any pushing	Innate, enjoys hard work. interesting/ challenging jobs	Lots done and progress made. Drives colleagues to work harder.
Resourceful /desire to learn	Will get/learn skills required, or find someone who knows.	Individual interest. Lack of pressure. Ownership, autonomy, responsibility.	Improved quality
Scientific/engineering/o rderly	Works preventatively sorting things out before you get too far into it. Methodical. Thoughtful about every stage.	Interest/ experience in specialist tasks. Ownership of code. orderly person	Morale boosting. Adds adaptability- cheaper to modify. aids customer perception – reputation.
Committed to team/ enjoys challenge / high standards/ professional	Doesn't get stressed. High expectations of themselves. Invests time on a difficult task	Accountability: High visibility project so problems will be noticed	Good drive to milestones
Responsible	Accepts responsibility (that you have to report bad news). Ownership.	Structure. Ownership. Autonomy. Punitive. Feature of young age and personality.	Gets rid of blame culture
Flexible	Learns and adapts quickly. Flexible/adaptable – able to change priorities.	Receptive to new ideas and technology - a natural thing	Project flows better. Things get done quicker and right. Minimises Rework effort

TABLE 2: Attributes of worst developer on the project, how attribute is encouraged and outcome			
Attribute (undesired)	**Defined as (from data)**	**Helped by**	**Outcome**
Technically incompetent	Hacks code. Doesn't take design seriously. Writes code that does not work. Caught up in tiny detail do not see the larger picture. No practical experience. Not logical about development	Not recognising poor requirements. Lack of time. Fear of re-training. Punitive management practices for missed deadlines. Personalise the punishment for missing milestones.	Quality slips. Problems with integration, dependencies, system reliability, Time delays, updating and maintaining artifact, complexity, completion. Shifts problem.
Insecure / closed / distrustful	Hides having a problem in a piece of software. Poor communication. Lack of faith to do anything if not trained. Too much watching his back. Does not share knowledge.	Strong need for self promotion- encouraged by managers Expectation that people can pick things up. Wanting to maintain consistent level of attainment.	Makes others paranoid; encourages them to behave defensively. If problems [that are hidden] are progress related affects time and money.
Inflexible	Don't like change - stuck in their ways. Arrogant.	Only want to do things they want to do in the way they want to do it. Trains, children, out late.	Negative effect on team - less cohesive/friction. Change and lessons learnt difficult to implement. Poor coordination = delays & imperfect solutions.
Over-confident	Too much confidence and faith in themselves. Arrogant. Self-deluded – people thinking they're better than they are	Brought up to think they are good. Too judgmental. Attempt to get recognised for promotion.	Other people get annoyed. Difficult to work with. Chooses work with high profile for him-self rather than good of project.
Disorganised	Take on too much at the same time .No foresight in needing more time etc	Under pressure and trying to save time. Looking good more important than doing good work	Affects schedule. Quality and design will be poor. Affects how often people think about their work. Affects the mindset of others. Distracts others.
Uncommitted to team / demotivated	Comes to work for works sake. Will do what you say and needs lots of direction.	If project managers not good at motivating person.	Affects colleagues and overall morale of team. More defects and latent faults. Lack of coordination - introduce delays & lead to imperfect solutions.
Unprofessional	People who panic under pressure. Very reactive to problems	Personal circumstances. Job insecurity.	Can affect everyone around them and affect overall morale.

Table 3: Cross-cutting themes emerging from the interview data (both negative & positive)	
Cross-cutting themes	**Definition**
progress	Relates to movement towards and achievement of goals, where goals might be milestones, meeting requirements, quality, timescales
access to information	Relates both to the flow of information regarding development matters, and the desire of individuals to acquire more information and to learn
openness/ communication	Relates to the flow of information, but it focuses more on the culture of the team and of individuals' willingness to help others, to say what they believe and to encourage others to do the same
responsibility/ autonomy/ ownership	Relates to Software Engineers defending their best ideas, voicing problems, working exceptionally hard, a culture where people don't blame others. Not waiting to be told what to do. Ownership of a process and piece of code.
software quality	Covers software reliability, integration, dependencies, complexity, meeting requirements, design, defects, latent faults, maintainability, solutions
team morale	Includes (de)motivation, team dynamics, drive others in team, morale boosting, blame culture, punitive management practices, negative effect on team, friction, team cohesion, annoying others, difficulty working with people.
fear/insecurity/ confidence	Includes both positive and negative characteristics. Behaviour is influenced by encouragement (or lack of it); threats from other team members performing too well; management personalising punishment for not meeting targets; treating others with distain. Job security. Behaviour is defensive and paranoid. Can be over-confident, which is seen as negative.

4.2 XP team results

We used the themes identified in Table 3 to interrogate observational data from the mature XP teams. We were looking to see what the data could tell us about how XP teams regard the themes. Of course, these themes overlap to some degree – for example, openness and communication are likely to affect access to information – but we focused on each one separately.

4.2.1 Progress. Progress was mentioned several times in the interviews: making progress was viewed as positive while barriers to progress were rated negatively. XP teams are also concerned about progress. This is reflected in a constant awareness of progress and talk about progress.

In all our teams, progress was carefully recorded and tracked with the story card being a key artefact. For example, in team C different coloured stars were used to denote the story's status: a red star indicated an unfinished card, yellow a card that has been finished by developers and is ready for acceptance test, green indicated that the change has been accepted by the customer. In addition, a blue star indicated that the card was a task card, i.e. an element of a story card. Team K used different coloured annotations to capture estimates, actual work done, and whether or not the story was 'finished'.

Another key artefact related to progress used in all our teams was a large visible display - the information radiator (or Wall) used to exhibit the cards. Cards were placed on the Wall in particular locations to indicate whether or not they were completed (see fig 1). For example in Team K cards at the top of the Wall were being worked on while cards at the bottom were finished. Rate of progress was therefore clear from one glance at the Wall. In Team S, there were two sections of wall – one for the analysts and one for the developers, to show the progress of work from analysis through to development.

Figure 1 All our teams had some form of Wall for displaying story cards and tracking progress

The use and focal nature of these artefacts also promoted awareness about progress. Stand-up meetings, which took place in all our teams every day, were held while standing around the Wall, the planning game where stories for the next iteration are chosen, inherently focuses on which stories have been completed and which are remaining.

In team S we witnessed several examples of the frustration caused when progress was held up because of others' mistakes (from outside the team), or lack of communication. For example, the team needed to access a database that was administered in

France, and the administrators there controlled password access, but didn't appear to communicate these passwords consistently to developers in the UK.

4.2.2 Access to information. The flow of information and active dissemination regarding implementation problems and successes, company matters and personal issues, is facilitated by stand-ups, regular meetings with customers (or customer proxies), and pairing, and by the visible nature of the Wall. The detail of how these were implemented in each team varied.

The information flows around Team K are simple yet rich [41]. In this team, every member of the development team was aware about all aspects of the company. This was also true in Team C, but it wasn't true in all our teams. For example in Team W the developers were divided into smaller teams to concentrate on individual products. When we enquired of one developer about progress on another team, they had no idea. While the information may not have been withheld, it was not actively disseminated. Indeed, the coach in Team W commented that part of his role was to protect the team from things they didn't need to know about.

Access to information to support individuals' desire to learn is facilitated in both the availability of information and the culture of the team in having courage. When pairing, developers have access to several information sources including developer websites, internally-documented wikis, books, each other and customers. The desire to learn was not strong in all our teams, but was strong in many of them. For example, Team C readily committed to learn Python when it became clear that this was the most appropriate language to use. One member of Team B told us that he could earn more money if he moved to a different job, but he was happy where he was because he was learning new things. Team K decided not to employ an outside consultant to work on a database issue because they wanted to keep the knowledge 'in house'.

In all our teams, the use of documentation was limited, in line with the XP philosophy of keeping documentation minimal. However when estimates are made in the planning game, this sometimes assumes a certain design choice, and if this choice is not communicated and remembered by developers, then it can lead to repeated work, or implementation based on a different design. This in turn can have an effect on actual work required to complete a story.

4.2.3 Openness/communication. In all our XP teams, the stand-ups and pairing appeared to be conducted in an open fashion. Team K would have a company stand-up every three weeks, and new members of the company were encouraged to chair these meetings if they wanted to. During our visit we observed an example of this. In some teams, however, it was clear that some individuals felt uncomfortable with this kind of approach. In team W one developer openly confessed that he didn't like pairing because he preferred to get on with things on his own (however the fact that he could say this without fear of recrimination speaks to the open nature of the team itself).

In Team S, the project manager exerted more control on the team and their work than we observed elsewhere. He called a team meeting during our observation, and one team member became very frustrated because he didn't understand the purpose of the meeting and he felt that it was wasting time when he could be making progress. This was an example of lack of communication between the project manager and the team.

4.2.4 Responsibility, autonomy, ownership. Responsibility is one of the characteristics of an XP team [52], as well as an underlying value of XP itself, and shared ownership is a theme we've seen in all our teams. The informative environment, openness of communication and regular exchange of ideas all contribute to this characteristic. In the interview tables above, there is a negative view of this theme, i.e. that a 'worst' developer wants to do things the way they want to and not listen to anyone else.

In our XP teams, decisions have tended to be made by consensus across the team or across a pair, bolstering team ownership and team responsibility. We observed successful outcomes of XP teams running retrospectives to modify process issues indicating ownership of a process. XP teams are self-organising and hence individuals take responsibility too. However, in one of our teams, we were told after the observation visit that one team member had significantly more influence over the team and its operation than others. It seems that the team as a whole was tolerant of this – an example of team strength.

One downside of everyone feeling responsible about the project and making team decisions is that it becomes difficult to identify distinct contributions of individuals. This can cause problems in terms of promotion and career progression. One member of Team B, for example, left the XP team specifically in order to gain experience in other areas of the bank because he was aiming for promotion. In Team W, none of the team members was given a particular

title, e.g. architect, senior (or junior) engineer, etc. This became an issue for individuals wanting to move to a different company, and so there was an informal agreement that for the purposes of job applications, individuals could decide on their own titles.

4.2.5 Software Quality. A clear theme that emerged from Team C's data is that coding and quality of code matters within an XP team [40]. This has been echoed throughout our teams, but the value of simplicity which leads developers to develop code that solves the current problem may result in code that is difficult to read. Hence refactoring (improving code structure while keeping the same functionality) is needed. However problems arise when the team is not given the time or opportunity to refactor. For example, one developer in team C became upset when he recognised the need for refactoring a piece of code that linked to software he was developing, but hadn't the time available in the current story estimate to work on it. Part of XP's philosophy is that every team member has the responsibility to modify code if they see that it needs improving, but in this case the desire for disciplined progress overcame the desire to improve code quality. Instead, the developer wrote a story card for the refactoring so that it could be handled in a stand-up like every other story, but we have not seen this kind of card in other teams.

4.2.6 Team Morale. Team morale is an outcome that may be influenced by many things, including the other themes identified in this section. The presence of this theme confirms that team morale is a key concern of developer teams. This is true whether the development approach is traditional or agile.

It would be inaccurate to suggest that our XP teams did not suffer from team morale issues. For example, in Team C, the regular rhythm of development led to significant boredom within the team. Team B also reported that they gave up trying to pair all day every day because it became too intense and difficult to sustain.

4.2.7 Fear/insecurity/confidence. This theme did not arise often in our interview data, and is more common when talking about 'worst' developers. There are many causes of insecurity and fear, which then lead to lack of confidence. In our XP teams, some developers were more technically competent and were more confident than others, but pairing exposes everyone to equal scrutiny. One side-effect of pairing that has been reported to us by several developers is that continuous pairing can lead to a lack of confidence in your ability to solve a problem when left on your own. Teams C, B, and W were under pressure to perform well due to organisational situations at the time of our visits, but fear or insecurity is not a strong theme in our data.

5. Discussion

Several of the issues raised by the traditional development team, while recognised as significant in XP, are not causing XP developers a problem, i.e. XP teams have found ways of tackling them or avoiding them. How? Is it because XP team members are 'just like that' or is it the structure of the practices that avoids common problems in software development teams?

There are clear differences in what is considered good and bad practice when looking at the two different development approaches. For example, commenting code is considered good practice in traditional development, whereas XP uses an alternative practice of clear naming – avoiding the need to spend time on comments. Differences in technical approach are also dependent on the type of software being developed. For example our traditional team were frustrated by working with developers who couldn't see the bigger picture, something of great importance when working on safety-critical embedded system development.

To explore the complex question of how motivational peer relationships are supported in XP we draw on the Agile literature and general literature on motivation. We incorporate findings from our data to consider where the practices and structure of XP teams support positive peer relationships, and where they support or conflict with views from traditional developers.

5.1 XP practices that support the traditional view

The need for developers to feel they are making progress is strong, which is to be expected in a group of professionals with extremely high growth need strength, i.e. individuals with a high need for personal growth and development [42, 43]. In terms of tracking progress, XP methods appear to meet these needs as shown by the display of story cards, regular stand-up meetings and the planning game which featured in all our XP teams. Should the developer want information the XP environment provides a good, accessible flow of information, as seen in regular meetings with customers, pairing and visible records on the wall. However, even in the XP

environment there are no controls for external influences holding up progress, such as mistakes and withholding of information by people outside the team.

The need to learn is strong in many developers, e.g. [3, 44], who according to our interview data, will use available resources to learn new techniques and develop new skills to solve problems. Indeed, one developer in our XP Team B preferred to stay in his current job because he had the opportunity to learn new things rather than move to a better paid job.

The XP teams all limited the use of documentation. Controlling the flow of information to ensure that only useful information is circulated to the team, protects individuals from information overload. (Regulating levels of information is likely to be a 'hygiene factor[1]' where too much information could be considered de-motivating.)

Developers are motivated by working in a team that is open with members who can communicate well (poor communication was found to be de-motivating in [24]). The informative environment, openness of communication and regular exchange of ideas helps to avoid the problem of developers hiding problems. There was no evidence in our XP data of developers hiding problems (although we cannot be certain), and we observed one developer speaking openly about his reservations with pairing, which suggests that he felt comfortable speaking freely about problems.

XP teams are self-organising and hence individuals take responsibility. Yet decisions tend to be made by consensus across the team or across a pair, bolstering team ownership and team responsibility (e.g. retrospectives). This practice discourages developers from working in an insular myopic fashion that de-motivates other members of the team.

5.2 XP practices that conflict with the traditional view

In an XP team, it is difficult to identify individual contributions as the work is so collaborative. This may cause those who want promotion or career progression to be de-motivated. According to the literature, career progression is an important motivator [46]. Furthermore, passive programmers can lose their motivation in pairing activities, as the dominant programmer tends to dictate the path for development [8].

The importance of fair treatment of individuals is encapsulated in equity theory [47]. People are likely to be de-motivated if their inputs such as experience, education, skills and seniority, are not matched by outputs gained from the organization such as salary, recognition, opportunity for achievement and so on. Practitioners will compare the balance of their inputs and outputs with others [47]. In a recent study of software engineers, Agarwal and Ferratt found unequal treatment of individuals de-motivating where key items were: compensation, assigned work, and recognition [48]. Equity is relative to others. Therefore there is a potential problem with the XP environment where individual inputs into the project might be subsumed by the whole, again impacting promotion opportunities.

Developers are known to have high growth needs, where task variety leads to job satisfaction [49]. These needs are not always met in XP. We observed instances where team morale was reduced by individuals becoming bored with the regular rhythm of development, where continuous pairing became too intense when practised day after day, and where developers became dependent on working in pairs, and lost their confidence to work alone.

6. Limitations

In this work we have considered two different ends of a spectrum – mature XP teams, versus a CMM level 5 team developing software traditionally. There are teams who take a balanced position between these extremes, integrating aspects from both, and we have not considered this balance in our discussion or in our data.

We did not ask developers directly what motivates them to work productively and to produce high quality software. Our results are drawn from questions relating to how developers view good and bad developers in their team. Our findings in this paper are therefore based on the understanding that developers like to work with – and are motivated by – good developers and don't like working with, and are de-motivated by working with bad developers [3-5, 7, 23, 25]. Factors unrelated to team interactions are not included, such as financial compensation and job security. The findings we present here are all based on how an individual's behaviour within a team might motivate or de-motivate other developers.

We cannot be sure that all motivators and desired attributes noted by our developers in a traditional

[1] Hygiene-motivator theory asserts that removing the de-motivator (or hygiene factor) will not necessarily translate to motivating employees. It will simply maintain practitioners in their job and avoid dissatisfaction [45. Herzberg, F., B. Mausner and B.B. Snyderman, *The Motivation to Work, 2nd Ed.* 1959, London: Chapman & Hall..

0-7695-2872-4/07 $25.00

environment would be motivating in an XP environment. There is likely to be a shift in attitude when moving from a heavyweight development methodology to a lightweight methodology. Further work in this area would be to replicate the study made in the traditional environment and ask XP developers what they consider to be the best and worst developers they have worked with in an XP environment. In this way we could see more clearly where the differences and similarities lie between each group.

7. Conclusion

In this paper we have presented an assessment of how XP practices meet the motivational needs of software developers, in terms of relationships with fellow team members. We have analysed data from traditional developers talking about how they view fellow team members, and extracted themes which reflect the issues uppermost in their minds. Using these themes we have interrogated data from mature XP teams, and have found that similar themes exist in this data too.

The five XP teams in our study had processes in place that supported many of the motivational needs voiced by developers coming from a traditional, heavyweight software development environment. Our observations revealed that XP developers were able to clearly monitor project progress, share knowledge, support those with less experience, adapt quickly to changes in requirements, learn on the job, work independently, and communicate good and bad news without worrying about punitive repercussions.

However, the XP environment doesn't support the need for individual recognition, for clear career progression, and variety of work, and may weaken the developer's ability and confidence to work alone, should the developer need to move out of an XP environment.

Acknowledgements: We are sincerely grateful to the companies and practitioners in this study who, for reasons of confidentiality, must remain anonymous. This research was supported by the UK's Engineering and Physical Science Research Council, under grant number EPSRC EP/D057272/1.

References

1. McConnell, S., Problem programmers. *IEEE Software*, 1998. **15**(2), 128, 127, 126.
2. Boehm, B.W., *Software Engineering Economics*. Advances in computing science and technology series, 1981, Englewood Cliffs: Prentice-Hall, Inc.
3. Procaccino, J.D., J.M. Verner, K.M. Shelfer and D. Gefen, What do software practitioners really think about project success: An exploratory study. *Journal of Systems and Software*, 2005. **78**(2), 194-203.
4. Andersen, E.S., "Never the twain shall meet": exploring the differences between Japanese and Norwegian IS professionals. In *Proceedings of the 2002 ACM SIGCPR Conference on Computer Personnel Research*, Kristiansand, Norway, May 14-16, 2002, pp 65-71.
5. Tanner, F.R., On motivating engineers, *Engineering Management Conference, 2003. IEMC '03*. Managing Technologically Driven Organizations: The Human Side of Innovation and Change, pp 214-218.
6. Klenke, K. and K.-A. Kievit, Predictors of leadership style, organizational commitment and turnover of information systems professionals. In *Proceedings of the 1992 ACM SIGCPR Conference on Computer Personnel Research*, A. L. Lederer, (ed) SIGCPR '92, pp 171-183.
7. Linberg, K.R., Software developer perceptions about software project failure: a case study. *Journal of Systems and Software*, 1999. **49**(2-3), 177-92.
8. Law, A. and R. Charron, Effects of agile practices on social factors. In *Proceedings of the 2005 Workshop on Human and Social Factors of Software Engineering (St. Louis, Missouri, May 16, 2005). HSSE '05*, pp 1-5.
9. Syed-Abdullah, S.L., J. Karn, M. Holcombe, T. Cowling and M. Gheorge, The positive affect of the XP methodology. In *Proceedings of XP 2005*, *LNCS 3556*. Springer-Verlag. 2005, pp 218-21.
10. Mannaro, K., M. Melis and M. Marchesi, Empirical Analysis on the Satisfaction of IT Employees Comparing XP Practices with Other Software Development Methodologies. In *Proceedings of XP 2004*, *LNCS 3092*. Springer-Verlag 2004, pp 166-174.
11. Asproni, G., Motivation, Teamwork, and Agile-Development. *Agile Times*, 2004. **IV**(1), 8-15.
12. Melnik, G. and F. Maurer, Comparative Analysis of Job Satisfaction in Agile and Non-Agile Software Development Teams. In *Proceedings of XP2006*, LNCS 4044, 2006, pp32-42.
13. Kuppuswami, S., K. Vivekanandan, P. Ramaswamy and P. Rodrigues, The effects of individual XP practices on software

development effort. *SIGSOFT Software Engineering Notes,* 2003. **28**(6), 1-6.
14. Müller, M.M. and F. Padberg, On the economic evaluation of XP projects. In *Proceedings of the 9th European Software Engineering Conference*, 2003, pp 168-177.
15. Baddoo, N., T. Hall and D. Jagielska, Software developer motivation in a high maturity company: a case study. *Software Process: Improvement and Practice*, 2006 **11**(3), 219-228.
16. McConnell, S., Avoiding classic mistakes [software engineering]. *IEEE Software*, 1996. **13**(5), 111-112.
17. DeMarco, T. and T. Lister, *Peopleware - Productive Projects And Teams.* 1999.
18. Standish Report (1995) *Standish Group Chaos Report*. Available from: URL http://www.scs.carleton.ca/~beau/PM/Standish-Report.html.
19. ProjectLink (2006) *Motivation House*. Available from: http://www.projectlink.co.uk/whoweworkfor.htm, accessed 12.5.2006.
20. Capretz, L., Personality Types in Software Engineering. *International Journal of Human-Computer Studies*, 2003. **58**(2), 207-214.
21. Ramachandran, S. and S.V. Rao, An effort towards identifying occupational culture among information systems professionals. In *Proceedings of the 2006 ACM SIGMIS CPR conference on computer personnel research*: Claremont, California, USA, 2006: p. 198-204.
22. Beecham, S., N. Baddoo, T. Hall, H. Robinson and H. Sharp, Motivation in Software Engineering: A Systematic Literature Review (in review). *ACM Computing Surveys*, 2007.
23. Jordan, E. and A.M. Whiteley, HRM practices in information technology management In *Proceedings of computer personnel research conference (SIGCPR)*, Alexandria, Virginia, 1994, pp 57 - 64.
24. Frangos, S.A., Motivated humans for reliable software products. *Microprocessors and Microsystems*, 1997. **21**(10), 605-610.
25. Klenke, K. and K.-A. Kievit, Predictors of leadership style, organizational commitment and turnover of information systems professionals. In *Proceedings of the 1992 ACM SIGCPR Conference on Computer Personnel Research*, 1992, pp 171-183.
26. Burn, J.M., E.M. Ng Tye, L.C. Ma and R.S. Poon, Job expectations of IS professionals in Hong Kong. In *Proceedings of the 1994 ACM SIGCPR Computer Personnel Research Conference*, 1994, pp 231-241.
27. Garden, A., Behavioural and organisational factors involved in the turnover of high tech professionals. *SIGCPR Comput. Pers.*, 1988. **11**(4), 6-9.
28. Igbaria, M., G. Meredith and D.C. Smith, Career orientations of information systems employees in South Africa. *The Journal of Strategic Information Systems,* 1995. **4**(4). 319-340.
29. Mak, B.L. and H. Sockel, A confirmatory factor analysis of IS employee motivation and retention. *Information & Management*, 2001. **38**(5), 265-276.
30. Niederman, F. and M.R. Sumner, Job turnover among MIS professionals: an exploratory study of employee turnover. *Proceedings of the 2001 ACM SIGCPR Conference on Computer Personnel Research* (San Diego, California, United States), 2001, pp 11-20.
31. Niederman , F. and M. Sumner, Decision paths affecting turnover among information technology professionals. In *Proceedings of the 2003 SIGMIS conference on Freedom in Philadelphia: leveraging differences and diversity in the IT workforce,* April 10-12, 2003, Philadelphia, Pennsylvania, pp 133-142.
32. Couger, J.D. and R.A. Zawacki, What motivates DP professionals? *Datamation*, 1978. **24**(9), 116.
33. Dittrich, J.E., J. Daniel Couger and R.A. Zawacki, Perceptions of equity, job satisfaction, and intention to quit among data processing personnel. *Information & Management*, 1985. **9**(2), 67-75.
34. Couger, J.D. and A. Ishikawa, Comparing motivation of Japanese computer personnel versus these of the United States. In *Proceedings of the Twenty-Eighth Hawaii International Conference on System Sciences*, 1995. **IV**, pp 1012-1019.
35. Couger, J.D., Comparison of motivation norms for programmer /analysts in the Pacific Rim and the U.S *International Journal of Information Systems*, 1992. **1**(3) 16-30.
36. Burn, J.M., J.D. Couger and L. Ma, Motivating IT professionals. The Hong Kong challenge *Information & Management*, 1992. **22**(5) 269-280.
37. Coram, M. and S. Bohner, The impact of agile methods on software project management. In *12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems. ECBS '05*, 2005, pp 363-370.

38. Highsmith, J. and A. Cockburn, Agile Software Development, The Business of Innovation. *Computer*, 2001. **34**(9) 120-122.
39. Robinson, H., J. Segal and H. Sharp, Ethnographically-informed Empirical Studies of Software Practice. *Information and Software Technology*, 2007, **49**(6) 540-551.
40. Sharp, H. and H. Robinson, An ethnographic study of XP practices. *Empirical Software Engineering*, 2004. **9**(4) 353-375.
41. Sharp, H., H. Robinson, J. Segal and D. Furniss, The Role of Story Cards and the Wall in XP teams: a distributed cognition perspective. In *Proceedings of Agile 2006*, 2006, pp 65-75.
42. Couger, J.D., Motivators vs. demotivators in the IS environment. *Journal of Systems Management*, 1988. **39**(6) 36-41.
43. Couger, J.D., Comparison of motivating environments for programmer/analysts and programmers in the US, Israel and Singapore., 1989. In *Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences*, 1989. **IV**, pp 316-323.
44. Enns, H.G., T.W. Ferratt and J. Prasad, Beyond Stereotypes of IT Professionals: Implications for IT HR Practices. *Communications of the ACM*, 2006. **49**(4) 106-109.
45. Herzberg, F., B. Mausner and B.B. Snyderman, *The Motivation to Work, 2nd Ed.* 1959, London: Chapman & Hall.
46. Smits, S.J., E.R. McLean and J.R. Tanner, A longitudinal study of I/S careers: synthesis, conclusion, and recommendations. In *Proceedings of the 1997 ACM SIGCPR Conference on Computer Personnel Research* (San Francisco, California, United States), 1997, pp 36-48.
47. Adams, J.S., Inequity in social exchange, in *Advances in experimental social psychology*. 1965, L. Berkowitz (ed.), New York: Academic Press: New York. pp 267-299.
48. Agarwal, R. and T.W. Ferratt, Crafting an HR strategy to meet the need for IT workers. *Communications of the ACM*, 2001. **44**(7) 58-64.
49. Reid, M.F., M.W. Allen, C.K. Riemenschneider and D.J. Armstrong, Affective commitment in the public sector: the case of IT employees. In *Proceedings of the 2006 ACM SIGMIS CPR conference on computer personnel research*. Claremont, California, USA 2006, pp 321-332.
50. Krippendorf (2004) *Content analysis: an introduction to its methodology*, London: SAGE
51. Berelson, Bernard (1952) *Content Analysis in Communication Research.* Free Press, New York.
52. Robinson, H., and Sharp, H. (2004) The characteristics of XP teams, in *Proceedings of XP2004* Germany, June, pp139-147