2006

Looking for a place to hide: a study of social loafing in agile teams

John McAvoy
University College Cork, j.mcavoy@ucc.ie

Tom Butler
University College Cork, tbutler@afis.ucc.ie

LOOKING FOR A PLACE TO HIDE: A STUDY OF SOCIAL LOAFING IN AGILE TEAMS

McAvoy, John, Business Information Systems, University College Cork, Ireland, j.mcavoy@ucc.ie

Butler, Tom, Business Information Systems, University College Cork, Ireland, tbutler@afis.ucc.ie

Abstract

Social loafing refers to the behaviour of individual members of a team who have tendency not to work as hard as they could or should, because social groups provide a degree of anonymity such that individual team members feel that their poor performance will be hidden by the overall output of the team. Agile Software Development philosophy espouses the importance of cohesive project teams, the empowerment of these teams, and the collective ownership of the code produced by the team — social values similar to those of communities of practice. This paper posits that one of the unintended consequences of Agile Software Development is that it may give rise to social loafing, under certain conditions. In order to test this proposition, research was carried out on two software development teams over an eight month period to determine if the values inherent in Agile Software Development could give rise to social loafing The theoretical assumption adopted by the authors was that the project team which fully adopted the agile approach would exhibit a greater tendency for social loafing, in comparison to the other team. The findings of the study indicate, however, that the opposite was the case; accordingly, the study's findings are interpreted to offer an explanation for this apparent paradox.

Keywords: Groups, Teams, Agile Software Development, Sociological factors, Social Loafing

1 INTRODUCTION

Software development relies heavily on teams of individual developers, yet the major focus of research in the area focuses on the methods and tools used in software development projects. Accordingly, there is a need to refocus on the teams and team dynamics, rather than the processes and tools they employ. Martin (2003, p.4) puts it best by pointing out that "a good process will not save the project from failure if the team doesn't have strong players." Nevertheless, strong players will, of and by themselves, not guarantee success. As with other 'communities-of-practice', software development teams manifest a range of problems in how team members work with, and relate to, each other. One particular concern with teams is social loafing by team members. Social loafing occurs when an individual team member deliberately does not work has hard as other members of the team. The phrase 'slacking off' is often used to describe this phenomenon. Here some software developers will purposively reduce their level of effort or productivity, because they perceive that other team members will take on the extra load, or they may believe that the group gives them sufficient anonymity so that their lack of effort will go unnoticed if the group is being evaluated as a unit, rather than at the level of the individual.

A review of extant literature on social loafing highlights a potential problem for software development projects that employ an Agile approach. Agile methodologies involve empowering cohesive groups to take collective ownership of their code: however, these map directly onto factors that are described as likely to cause social loafing.. This study presents the findings of two contrasting case studies of software development teams in order to investigate the conditions that give rise to social loafing. The two cases involved software development projects that used Agile approaches; however, one project team's application of the Agile methods was diluted by the need to adhere to company-wide standards and processes around software development. Drawing on extant theory, it was predicted that the incidence of social loafing would be higher in the project that followed more closely the Agile philosophy on software development. The remainder of this paper is structured as follows: the second section discusses research on Agile Software Development teams and the literature on team dynamics with respect to social loafing. The third section then presents the research approach, while the fourth discusses ad analyse the findings. The final section then offers some conclusions and makes recommendations for future research.

2 SOCIAL LOAFING IN AN AGILE ENVIRONMENT

The term software crisis was first used at a NATO conference in Germany in 1968 (Hazzan & Tomayko 2003). Thirty years on, Wastell (1999, p. 582) argues that "[d]espite impressive technical advances in tools and methodologies and the organizational insights provided by many years of research, IS failures remain all too common." This may be explained in part by the fact that researchers and practitioners responded to the crisis by attempting to introduce engineering principles to software development, and these principles are visible in a wide variety of methodologies in use: yet despite this, the software crisis continues to persist (Fitzgerald 1999). Yourdon (1985) predicted that the software crisis would be solved by the year 2001, with the growing array of tools, methods, and procedures that would then be available; paradoxically, however, Yourdon felt that that the problem was really a sociological rather than a technical one. Nevertheless, Yourdon chose tools over people, due to the difficulties in dealing with the people problems, and the hope that tools and techniques could overcome such difficulties. Bahli and Buyukkurt (2005) acknowledge the central role teams in software development, yet state that there has been little research to date. Accordingly, Carreira and Silva (1998) argue that there is a paucity of research on human factors: Sawyer and Guinan (1998, p. 552) echo this and state that "[s]ince software development is, at the least, partly a

social process means that understanding how people work together to build software is critical". Sawyer and Guinan argue that software development needs to be refocused from product to process, particularly the social processes that underpin development activities as methodology and tool use have less of an affect on project outcomes than the socialization of developers in a team. This study therefore concentrates on the social dimension of software development, by specifically examining the software development team and the impact that the team can have on the development process.

2.1 The importance of teams to Agile Software Development

Software development depends on teamwork because "the scale of work is beyond any one person" (Kelley & Caplan 1997, p. 49). Sawyer and Guinan (1998, p. 553) conceptualise a software development team as "two or more software developers who are engaged in building a defined product to be delivered within a certain time frame. A team relies on the collective skills of its members because of the scope of the effort, the inherent complexity of the effort, and the number of tasks needed to develop modern software that normally exceeds the ability of any one developer." Despite the emphasis on tools and techniques, teamwork came to be regarded as a 'silver bullet' for the majority of problems that beset the work environment (Cartwright 2002, p.3). Stewart, Manz and Sims (1999, p. 4), for example, proposed that autonomous teams have the potential to increase productivity, while Martin (1991, p. 155) held that "better team working leads to better performance." Agile software development approaches place teams at the centre of the agile process. Highsmith (2004), for example, emphasises the importance of a good team for the success of agile projects, while Hazzan and Tomayko (2003) describe XP (eXtreme Programming) as being based on team interaction—more so than other software development methodologies. Stephens and Rosenberg (2003, p. 94) therefore point out that "agile methods have a much higher emphasis on people then previous methodologies." Hence it may be argued that the social aspects of Agile development align with those that underpin communities of practice. In articulating the latter, Wenger (1998) describes the commitment of members to communities or groups, while Wenger and Snyder (2000) discuss the mutual engagement that binds team members. Ferlie, Fitzgerald, Wood and Hawkins (2005), building on Wenger's work and concentrating on professional communities of practice, argue that communities of practice can lead to teams or groups that are so cohesive that they distance themselves from other groups to maintain its own identity. These levels of cohesion in communities of practice have also been described in Agile teams (McAvoy & Butler 2006).

The basic principles of agile methodologies are reported in Abrahammson, Salo, Ronkainen and Warsta (2002), and Fowler and Highsmith (2001) as:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

Together these principles are referred to as the 'Agile Manifesto' and form the core values of agile methodologies. The first principle of Agile software development highlights the importance of groups (the interaction of individuals) to agile software development. The importance of group work is reflected in the devolution of power to software development teams and the expectation that the team as a unit is responsible for development of software. Here, the code is collectively owned (Hazzan & Tomayko 2004), fostering a democratic, cooperative approach with regard to responsibilities. This agile concept of collective ownership is also described in Beck (2000) and Cohn (2004). Stephens and Rosenberg (2003) describe collective ownership as one of the twelve core XP practices. Allied to this, collective ownership is reflected in the trust shown to developers where they are empowered, rather than monitored and controlled. Schuh (2004) specifically associates agile with empowerment and trust, to the extent that the team has responsibility for the delivery of all functionality (Cohn 2004). To enable empowerment to work, teams needs to be well-functioning and cohesive: Auer, Meade and Reeves (2003) describe such teams as effective social networks, that interact well (Boehm & Turner

2003) and which jell together well (Highsmith 2004). This empowerment of, and belief in, software development teams is not new, however, as researchers on Agile Software development argue for Theory Y over Theory X (McGregor 1960) approaches to managing software development teams as Theory Y postulates that team members want responsibility and enjoy work (cf. Cartwright 2002, Landy & Conte 2004).

2.2 From Agile to Social Loafing

Stewart *et al*. (1999) propose two major drivers for teamwork: functional perspectives and interpersonal perspectives. The functional perspective posits that, all things being equal, individuals working together enable certain tasks to be performed better than when individuals working alone. This seems to hold in other situations as Triplett (1897) found that the mere presence of others, cyclists in his experiment, improves individual performance—this is referred to as the Dynamogenic Theory. The interpersonal perspective maintains that individuals require social interaction. Zajonc (1965) found that the performance of many tasks improved in the presence of others; the term he uses to describe this is Social Facilitation (Williams, Karau & Bourgeois 1993). Interestingly, Baron, Kerr and Miller (1999) describe how Zajonc found the same effect in other animals, including cockroaches —although it is not the aim of this study to compare software developers to cockroaches! A further benefit of working in teams is an increase in productivity; for example, Hare (1994) points out that the group effect arises from the collective memory, greater problem solving ability, and creativity of groups.

Research into groups has often produced paradoxical findings (Baron *et al*. 1999). Nunamaker, Briggs, Mittleman, Vogel and Balthazard (1997) indicate that while teams are vital in many situations, teams bring with them their own problems. For example, they illustrate that a group of four people will not perform four times better than one individual. This shortfall in the group's performance is described as process loss, where a group does not act in the most effective way. Other forms of inefficiency in group activities include coordination loss, where the group does not effectively coordinate the work, and motivational loss, where the group members do not try as hard as they would if they were working as individuals (Baron *et al*. 1999). Balthazard, Potter and Warren (2004) add that group performance is generally worse than the performance of its best member, but better than the average performance of all members. Part of these problems can be explained by the difference between the ideal of teams and the reality of teams. Robbins and Finely (1998, p. 51) differentiate between ideal teams and real teams. Ideal teams comprise "perfect people whose egos and individuality have been subsumed into the greater goal of the team." Real teams, the reality of teams in the workplace, "are made up of living, breathing, and very imperfect people."

While Social Facilitation brings positive effects to groups, its opposite – Social Loafing – will reduce a group's performance (Baron *et al*. 1999, Brooks & Ammons 2003). Thomson (2003, p. 100) defines "Social loafing [as] the tendency for people in a group to slack off—i.e., not work as hard either mentally or physically in a group as they would alone." Thomson holds that it is human nature for individuals to 'slack off' if given the opportunity. The ability to hide in a group, where individual lack of effort may not be noticed, provides this opportunity. Brooks and Ammons (2003) use the term free-riding to describe what is in effect social loafing, identifying its prevalence in group-based projects in education. It should be noted, however, that Mulvey and Klein (1998) differentiate social loafing from free-riding. Free-riding, although very similar to social loafing, involves the perception of one or more team members that other team members will put in sufficient work, making their own contribution less of an issue. Nevertheless, Mulvey and Klein accept that the terms are generally used interchangeably.

Williams *et al*. (1993) describe the factors that affect social loafing:

- Social loafing increases when evaluation of the work is based on group, rather than individual performance.
- Social loafing is less likely to occur if the work is interesting

- Group cohesiveness can reduce social loafing: Williams *et al.* (1993) and Mulvey and Klein (1998) also found that social loafing can occur in cohesive groups, where group members trust each other to do their tasks. This level of trust means that individual performance is not monitored by team members, allowing the opportunity for loafing (this appears pertinent to Agile teams).

Pearce and Ensley (2004) elaborate on this by stating that role ambiguity is a further cause of social loafing, while Landy and Conte (2003) argue that a lack of monitoring can lead to social loafing. The factors influencing social loafing above, imply a potential problem in software development teams adopting an agile methodology.

2.3 Shirking, Free-Riding and Opportunism in Joint Team Production

The similarity between social loafing and free riding has been mentioned above, when discussing sociological theories. The difference between them may be explained, in part, by the underlying assumptions about human nature and the organisational responses required (c.f. Scott 1995, on the regulative normative and cognitive influences perspective in and on organisations). In economics, Alchian and Demsetz (1972: Theory of Joint Team Production), Jensen and Meckling (1976: Agency Theory and the Principal-Agent problem) and Williamson (1985: Transaction Cost Economics and Opportunism) address the problem of asymmetrical information in economic organizations. This situation is said to lead to problems such as opportunism, shirking or free riding by social actors. Hence, the existence of moral hazard leads firms to monitor employee activities and introduce incentives to prevent and counter inappropriate behaviour (Fransman 1998). Shirking by team members is countered using a central contracting agent—the employer—who monitors and meters the inputs of team members and adjusts their contracts accordingly. The incentive for employers to monitor employee input is that residual rewards will accrue to the firm and will not be lost through inefficiencies associated with shirking (ibid.). Thus, the prerequisite for software teams to religiously adhere to rigorous methods and techniques aimed at improving software processes may be viewed as a regulative response to such problems. Pfeffer (1994), however, provides a scathing critique of these perspectives, particularly agency theory and opportunism, and cites empirical evidence in support of his point that prior negative or pejorative assumptions about human behaviour are in many ways self fulfilling theories and that the removal of such assumptions will have the opposite effect (Ferraro, Pfeffer & Sutton 2005).

2.4 Theoretical Predictions

Agile approaches stress the empowerment of, and trust in, software development teams, to the extent that teams are permitted to monitor themselves to a large extent. The agile team is responsible as a unit for the development of the product, so the group is evaluated as a whole on this, as opposed to individual evaluation of the team members. The collective ownership of code by agile teams permits a degree of anonymity for individual developers. The requirement for a cohesive agile team is also deemed to be important. These features of agile software development match closely two of the three factors which affect social loafing: group evaluation and group cohesiveness. The third factor associated with social loafing – interesting work – does not appear to be restricted specifically to agile approaches, as there is nothing in the agile philosophy which addresses how agile development would be more or less interesting than traditional development. A logical prediction or proposition may be drawn from the forgoing[1]: *The incidence of social loafing will be higher in software development project teams that rigorously adopt Agile approaches and methods than those who adopt a more eclectic approach to agile development by applying formal software development methods with high levels of monitoring and control over team members and their work.*

[1] The work of Allen Lee in his 'A Scientific Methodology for Case Studies' informed the authors' approach to this study.

3 RESEARCH APPROACH

Several studies have employed participant observation to conduct research into agile software development projects. Martin, Biddle and Noble (2004) argue that interpretative, in-depth case studies are the most appropriate method of investigating agile software development. A qualitative approach involving participant observation was used by Robinson and Sharp (2004) to investigate the characteristics of an agile team and provided rich insights that could not be obtained by other research methods. The trust gained by the researchers in such studies enabled them to examine factors that would otherwise have remained concealed. Schwartzman (1993, p. 4) argues that it is the "taken for granted" that is worth observing in social contexts, for it is the seemingly trivia of daily work that influence organisational outcomes as this everyday life constitutes reality (Jorgensen 1989). Thus a case study-based research approach using participation observation was chosen to research the phenomenon of interest.

Two software development project teams were chosen using purposeful sampling to test the theory and its predictions. The first project team designed and developed a knowledge management system for a European government organisation. A team of seven developers and a project manager were involved in the project, the first phase of which lasted eight months. The second team was charged with developing fault tolerant applications for a large US-based multinational telecommunications company: this team consisted of eight developers and a team leader. Similarly sized project teams were purposively selected to eliminate team size as a contributing factor in social loafing, as Williams *et al*. (1993) found a correlation between an increase in social loafing and an increase in group size.

Participant observation was chosen as the primary research technique to investigate the phenomenon of interest, as it is a particularly relevant approach when "the phenomenon is obscured from the view of outsiders" (Jorgensen 1989, p. 12)—social loafing is one such phenomenon. Participant observation of the first development team occurred over an eight-month period, in what was a longitudinal research study. Both researchers participated as members of the team, one as a developer and one as the project manager, and were therefore integrated into the group under study more or less continuously—participating in all team meetings, formal and informal discussions, and so on: such activities are argued to be vital in participant observation (Ezey 2003), as it "allows you to experience activities directly to get a feel of what events are like, and to record your own perceptions" (Spradley 1980, p. 51). Research on the second team also involved a longitudinal approach over a one year period, however, only one of the researchers participated in the team and his involvement was intermittent and occurred at specific intervals, especially during team meetings and informal discussions with developers: however, the researchers also engaged with two key informants—the team leader and one of the developers—at regular intervals in order to monitor team progress towards development objectives and obtain additional insights into team dynamics. Detailed field notes were taken throughout and these were reflexively analysed and recorded by the researchers. The various themes, issues and group interactions were identified as the study progressed and initial observations subsequently confirmed. Thus the study's credibility was ensured due to prolonged engagement, persistent observation, and triangulation (Erlandson & Harris & Skipper & Allen 1993): the researchers also peer debriefed each other, while member checks also occurred informally and formally at the end of the study when developers and team leader/project manager were interviewed.

4 OBSERVATIONS AND ANALYSIS

As indicated, the two cases were chosen because of the differences in their respective approaches to software development. The project team developing the knowledge management application followed the agile philosophy to a greater degree than the project team developing telecommunications software. The latter team used a diluted version of agile, as they had to ensure compliance with company standards. One of these is TL9000, which is a standard governing the development of

communications software (Clancy 2002) which was adopted by the organisation. In addition, as the organisation also adopted the Capability Maturity Model (CMM), all software development project teams are regularly audited for compliance with the CMM: thus there were a variety of processes and procedures that the project team had to follow when writing code and performing supporting and related activities. This, however, went somewhat against the agile philosophy of individuals and interactions over processes and tools; nevertheless, the organisation still considered this to be an agile project.

Various researchers argue that agile approaches that are altered to facilitate mandated company processes and standards still conform to the agile philosophy (Auer *et al*. 2003, Aveling 2004). It is therefore accepted that not all of the agile practices need to be followed for software project to be agile; for example, Wright (2003) describes how one company successfully adapted an agile approach to achieve ISO 9001 certification. The two cases therefore present two contrasting views of the implementation of agile approaches: the complete adoption of agile philosophies by the knowledge management project team and a partial adoption of the agile philosophies by the telecommunications team. Based on the nature of these projects, it was proposed that the more formalised, monitored, and audited telecommunications project should demonstrate a lower likelihood of social loafing. As indicated, both sets of field notes were analysed for the prevalence of social loafing; however, it was not observed in the knowledge management project, contrary to expectations.

4.1 Case #1: The Knowledge Management Project Team

Field observations revealed that team members in the knowledge management project team demonstrated a collective ownership of the project that did not have the expected consequences. One incident that occurred during the development of the graphical user interface for the knowledge management tool highlights how the individual members of the team did not slack off. At a critical point in the development of the software, the project manager voiced concerns with the overall design of the Web-based graphical user interface (GUI). While the team as a whole acknowledged that the design of the GUI required particular resources, it had, nevertheless, accepted responsibility for its development—hence the project manager's displeasure at what he considered to be a "clunky GUI" in a project review meeting. His style of project management was very much in keeping with Agile's philosophy, in that he devolved responsibility to and trusted the team to apply their considerable expertise to ensure that all aspects of the IT artifact met the highest design standards.

However, the team had, informally and unconsciously, taken the collective decision to focus its efforts on the design and development of the IT artifact's internal architecture, while one team member had 'volunteered' to build industrial strength screens for the user interface. If social loafing had occurred, the developers could have "hidden" within the group as the GUI was developed by the entire team. Rather than hiding within the group, the developer who had informally taken it upon himself to design the GUI, demonstrated a degree of ownership of the GUI by disagreeing quite forcefully with the opinions of the project manager. There was, in this particular instance, no need for this developer to "stand out from the crowd", yet he did so. Furthermore, other developers in the team joined in the defence of the GUI. Thus the team defended the GUI collectively, rather than individuals hiding within the group, or indeed the group behind the individual developer. The ensuing argument was quite forceful, with raised voices and red faces all round and it continued for nearly an hour.

A few days after the event, the lead developer on the internal architecture was asked for his perceptions about this disagreement on the GUI. He said that he felt that "our work was being called into question." It is interesting the note the use of the word "our" as it highlights that the group defended the work of all the team members, whether or not they were personally responsible for perceived deficiencies. This incident was not the only one where the team had defended itself collectively; although the GUI incident above was the most vociferous argument—other disputes about the conceptual design and features of the IT artifact tended to be more amicable. These incidents highlight a high degree of cohesion among the team members. Beal, Cohen, Burke and McLendon

(2003) define the three components of cohesion as interpersonal attraction, task commitment, and group pride. While interpersonal attraction was not specifically demonstrated in the group defence seen, observations of the team over eight months showed that cohesiveness of the team as a social unit was reflected in a high degree of affective attraction and familiarity that resulted in these relationships being extended beyond the workplace into sporting events, pubs and nightclubs. Task commitment is seen in the group's sense of ownership of the work. The dedication to achieving high standards of robustness in terms of the internal architecture and the arguments and defence of the team's work on the GUI would not have been as strident if the team had not been committed to the task. Finally, group pride is also obvious from the defence of their work. According to extant 'theory', the knowledge management project team should have shown signs of social loafing, due to the evaluation of the team rather than the individual and the high cohesion of the team, yet the team demonstrated what would be better defined as social facilitation rather than social loafing. Rather than demonstrating that a team adopting an agile philosophy will be inclined towards social loafing, the observations demonstrated the improved performance described as social facilitation.

4.2 Case Study #2: The Telecommunications Project Team

It was expected at the outset this case study that social loafing would be limited by the presence of both individual and group evaluation or monitoring, which was required by company Software Process Improvement procedures and the application of software development standards. Added to this were the lower levels of social cohesion present in the telecommunications project team, which, according to theory, should have reduced the likelihood or incidence of social loafing. Nevertheless, while cohesion was lower than the knowledge management project team, and extracurricular cohesion limited, the team was cohered as work group; furthermore, it simply did not have as much autonomy as the first team and ownership of the software development process as the first. Social Loafing theory and Agency Theory would both predict that lower levels of social cohesion and higher levels of individual and group monitoring would have led to the existence of social loafing or opportunism/free-riding.

Prior to the introduction of agile methods, one of the approaches to improving software quality in the telecommunications organisation was the code review (as is common in many organisations). Under this scheme of things, code reviews assisted the developer whose code was under review; however, problems with, or deficiencies in, the software code were evaluated against this individual rather than the team. When agile methods were introduced into the team, the response to the practice of group ownership of code reviews and their consequences was met with extreme disapproval from team members. So strong was the disapproval was that the inclusion of the practice of group ownership was dropped by the team leader. Closer investigation of the existing code review process was undertaken to determine if, counter to expectations, social loafing existed in the team.

It was quite apparent from the outset, however, that social loafing was quite prevalent in the code review process; and was quite probably the reason why the team members did not want to modify the existing process to render it closer to the agile model in the first place. While fellow team members always provided advice and assistance to fellow software developers who were being evaluated on their code base, such advice and assistance was limited to the duration of the code review. It was noted, and notable, that the team members conducting the reviews rarely inspected the code under review before the review meeting—most could clearly be seen reading the code and marking problems during the review meetings. It was clear that they were assisting their colleagues, but only to the extent that they had to; it was, to the eyes of an independent observer, clearly not a whole-hearted effort on behalf of the reviewers. The team lead was in attendance at code reviews, so the developers/reviewers had to do what was required of them during the review process, as they were being evaluated on their performance at the review by the team lead. Developers therefore performed as expected during the review; however, as the work they put in prior to the review as not monitored (and therefore not evaluated), there was little incentive to subject the code under review to a detailed analysis before the

review process proper. Thus, it was concluded that the process reviews were essentially an exercise in 'going through the motions' and while problems were identified, it was clear to the field researcher (who was an experienced programmer/IT professional) that they could have been more effective if team ownership of, and responsibility for, all code produced by the team was the norm.

What was significant were the responses from the two key informants on this observation. The developer informant agreed that he, and his colleagues, "slacked off" from the work before the code review. Interestingly, though, the second key informant – the team leader – was not actually evaluating the team during the code review, as he was himself guilty of the team's crime. The team leader admitted that he too was busy reading and analysing the code during the review, as he had not read the code before the review. He had not noticed that the majority of the team were doing the same.

Further examples of social loafing were seen in this team and in each case it was related to the institutionalised processes around software improvement. For example, in one of the regular meetings with the team leader, he admitted that he was influenced by his discussions with field researcher on this case study on the causes and consequences of social loafing; he therefore became more aware of its existence in his team. The team leader noticed that the developers sometimes "hid behind" the company's software quality and improvement processes. He reported that the developers sometimes (and he stressed sometimes) used the company's processes as a means of avoiding evaluation altogether. For example, if a developer was late in delivering required functionality, the company's processes were attributed as being the cause of the delay. The team leader described this as akin to the statement that "I was only following orders". Although existing research on social loafing does not mention process compliance as a factor, this "process loafing" does appear to be having a similar impact as social loafing. Pugh (1993) made a similar observation when describing the dynamics of organisations. Pugh found that, in some cases, processes that are used to ensure uniformity of performance can create a tendency to hide behind the rules.

5 CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

This paper does not posit that the use of Agile Software Development Methodologies are the panacea for problems of a social origin that occur in development teams. Nor does it posit that formalised software process improvement and/or quality standards will give rise to such problems. The longitudinal aspect and wider focus of the overall research undertaking, which is outside the scope of this paper, illustrated the benefits and problems with both formalised and Agile approaches. What is clear, though, is that both approaches can have an impact on the incidence of social loafing. Again, the use of the word "can" is important. It is impossible, even with the much touted benefits of longitudinal research, to make generalisations from two case studies. We limit the scope of our conclusions to state that the choice and institutionalisation of a formal or an Agile approach in an organisation can impact the incidences of social loafing in software project teams. The two main findings of this paper are:

- The existence of a highly cohesive software development team, with ownership of, and control over, their work is, all things being equal, less likely to lead to the existence of social loafing among team members. It is also evident that Agile software development methodologies promote these high levels of social cohesion and a sense of ownership among team members. These values align with, what are considered to be, attributes of communities of practice.
- Individual monitoring and evaluation of software developers in teams who religiously apply formal processes can have the opposite effect to that predicted by extant research on social loafing and the problem of opportunism and free-riding in economics. Previous research predicts that individual, versus group, monitoring will lower social loafing. What was actually found was that individual evaluation allowed those that were not the focus of the evaluation to engage in social loafing.

This paper's theoretical proposition regarding the circumstances that give rise to or prevent social loafing was therefore not validated. In fact the opposite may be stated, as the Agile philosophies of

empowerment and group ownership appear to work against the tendency towards social loafing among members of software development teams. If a team fully adopts an Agile philosophy, team members tend to work better as a group by supporting each other, rather than hiding within the group to disguise problems with the quality of their work. The ability to remain anonymous within the group (a diffusion of responsibility) appears to be mitigated by high levels of social cohesion within the group or community. Landy and Conte (2003) argue that a lack of monitoring can lead to social loafing. While the Agile methodologies may appear to have a lack of monitoring of the individual, Barker (1988) does provide an explanation. Although not addressing social loafing, Barker notes how the social control exerted by a self-managing team can exert a greater level of control over an individual than the traditional command and control management structure. Software development processes that strive to ensure adherence, by monitoring and evaluating individuals in a software development team, do not appear to be as effective at eliminating social loafing as the Agile philosophy of empowerment and collective ownership.

On a general note, this paper also provides further empirical support for the observations of Pfeffer (1994) and Ferraro *et al*. (2005) who argue against the perverse and self-fulfilling effect of economic theory (e.g. on Agency Theory and Opportunism) on the nature of work. In this scheme workers (and almost everyone else) are not to be trusted—they will shirk, free-ride and be opportunistic, unless closely monitored, and so on. Pfeffer (1994) and Ferraro *et al*. (2005) argue that such perspectives colour management thinking, especially the perspectives of managers in US-based organisations. That the behaviour of the software development team in the US telecommunications team conformed to negative perspectives on human nature and motivations (e.g. Theory X) is interesting, as is the behaviour of the team in the European organisation in which a softer view of human motivations and behaviour existed (e.g. Theory Y).

References

Abrahamsson, P., Salo, O., Ronkainen, J. and Warsta, J. (2002). Agile software development methods. Review and analysis, VTT Publications. Finland.

Alchian, A. A. and H. Demsetz, (1972). Production, Information Costs, and Economic Organization, American Economic Review, 62, 777-795.

Auer, K., Meade, E. and Reeves, G. (2003). The rules of the game. in Maurer, F., Wells, D. (eds) Extreme programming and agile methods – Xp/Agile universe 2003. New Orleans, Springer-Verlag. Berlin, Germany, 35-42.

Aveling, B. (2004). XP Lite considered harmful? in Eckstein, J., and Baumeister, H. (eds) Extreme programming and agile processes in software engineering. 5th International Conference. Germany, Springer-Verlag. Berlin, Germany, 94-103.

Bahli, B. and Buyukkurt, M. (2005). Group performance in information systems project groups: An empirical study. Journal of Information Technology Education. 4. 97-113.

Balthazard, P., Potter, R. and Warren, J. (2004). Expertise, extraversion and group interaction styles as performance indicators in virtual teams, The Database for Advances in Information Systems. 35(1), 41-64.

Barker, J. (1988). Tightening the iron cage: Concertive control in self managing teams. In Van Maanen, J. (ed) Qualitative studies of organizations, Sage Publications. CA, USA, 126-158.

Baron, R., Kerr, N. and Miller, N. (1999). Group process, group decision, group action. Open University Press. Buckingham, UK.

Beal, D., Cohen, R., Burke, M. and McLendon, C. (2003). Cohesion and performance in groups: A meta-analytic clarification of construct relations, Journal of Applied Psychology, 88(6), 989-1004.

Beck, K. (2000). Extreme programming explained. Embrace change. Addison Wesley. NY, USA.

Boehm, B. and Turner, R. (2003). Rebalancing your organizations agility and discipline. in Maurer, F., and Wells, D. (eds) Extreme programming and agile methods – Xp/Agile universe 2003. New Orleans, Springer-Verlag. Berlin, Germany, 1-8.

Brooks, C. and Ammons, J. (2003). Free riding in group projects and the affects of timing, frequency, and specificity of criteria in peer assessments. Journal of Education for Business, 78(5), 268-272.
Carreira, J. and Silva, J. (1998). Computer science and the Pygmalion effect. IEEE Computer, 31(2), 116-117.
Cartwright, R. (2002). Mastering team leadership. Palgrave Macmillan. Wales.
Clancy, B. (2002). Are there real benefits from TL9000. Quality Times, 1(3), 7-9.
Cohn, M. (2004). User stories applied for agile software development. Addison-Wesley. MA, USA.
Erlandson, D.A., Harris, E.L., Skipper, B.L. and Allen, S.D. (1993). Doing Naturalistic Inquiry: A Guide to Methods. Sage Publications Ltd., London.
Ezey, P. (2003). Integration and its challenges in participant observation. Qualitative Research, 3(2), 191-205.
Ferlie, E., Fitzgerald, L., Wood, M. and Hawkins, C. (2005). The non-spread of innovations: The mediating roles of professionals. Academy of Management Journal, 48(1), 117-143.
Ferraro, F., Pfeffer, J. and Sutton, R. (2005). Economics Language and Assumptions: How Theories can Become Self-Fulfilling, Academy of Management Review, 30(1), 8-24.
Fitzgerald, B (1999). Information systems with RAD in Adam, F., and Murphy, C. (ed) A mangers guide to current issues in information systems. Blackhall Publishing, Dublin, Ireland, 105-116.
Fowler, M. and Highsmith, J. (2001). The agile manifesto. Software Development. 9(8). CMP Media
Fransman, M. (1998). Information, Knowledge, Vision, and theories of the Firm. In G. Dosi, D.J. Teece and J. Chytry (Eds), Technology, Organisation, and Competitiveness: Perspectives on Industrial and Corporate Change, Oxford University Press Inc., New York, NY, 147-192.
Hare, A. (1994). Individual versus group. in Hare, A., Blumberg, H., Davies, M., and Kent, M. (eds) Small group research: A handbook. Ablex Publishing, NJ, USA, 261-270.
Hazzan, O. and Tomayko, J. (2003). The reflective practitioner perspective in extreme programming in Maurer, F., and Wells, D. (eds) Extreme programming and agile methods – Xp/Agile universe 2003. New Orleans, Springer-Verlag, Berlin, Germany, 51-61.
Hazzan, O. and Tomayko, J. (2004). Human aspects of software engineering. in Eckstein, J., and Baumeister, H. (eds) Extreme programming and agile processes in software engineering. 5th International Conference. Germany, Springer-Verlag, Berlin, Germany, 303-311.
Highsmith, J. (2004). Agile project management. Pearson Education. MA, USA.
Jensen, M. and W. Meckling (1976). Theory of the Firm: Managerial Behavior, Agency Costs, and Ownership Structure, The Journal of Financial Economics, 3, 305-360.
Jorgensen, D. (1989). Participant observation: A methodology for human studies, Sage Publications. CA, USA.
Kelley, R. and Caplan, J. (1997). How Bell Labs creates star performers. in Katz, R. (ed) The human side of managing technological innovation, Oxford University Press, NY, USA, 47-59.
Landy, F. and Conte, J. (2004). Work in the 21st century. McGraw-Hill, NY, USA.
Lee, A.S. (1989). A Scientific Methodology for MIS Case Studies, MIS Quarterly, 13 (1), 33-52.
Martin, R. (1991). Working in groups. in Smith, M (ed) Analysing organisational behaviour. Macmillan Press. London, UK. 154-177.
Martin, R. (2003). Agile software development. Principles, patterns, and practices. Prentice Hall. NJ, USA
Martin, A., Biddle, R. and Noble, J. (2004). When XP met outsourcing in Eckstein, J., and Baumeister, H. (eds) Extreme programming and agile processes in software engineering. 5th International Conference. Germany. Springer-Verlag, Berlin, Germany, 51-59.
McAvoy, J. and Butler, T. (2006) Resisting the change to user stories: A trip to Abilene. International Journal of Information Systems and Change Management. 1(1), 48-61.
McGregor, D. (1960). Theory X and theory Y. in Pugh, D. (1990) Organization theory. Third Edition, Penguin Books, London, UK, 358-374.
Mulvey, P. and Klein, H. (1998). The impact of perceived loafing and collective efficacy on group goal processes and group performance. Organizational Behaviour and Human Decision Processes. 74(1). 62-87.

Nunamaker, J., Briggs, R., Mittleman, D., Vogel, D. and Balthazard, P. (1997). Lessons from a dozen years of group support systems research: A discussion of lab and field findings. Journal of Management Information Systems, 13(3), 163-207.
Pearce, C. and Ensley, M. (2004). A reciprocal and longitudinal investigation of the innovation process: The central role of shared vision in the product and process innovation teams. Journal of Organizational Behaviour, 25(2), 259-278.
Pfeffer, J. (1994). Competitive Advantage through People: Unleashing the Power of the Workforce, Harvard Business School Press, Boston, MA.
Pugh, D. (1993). Understanding and managing organizational change. in Mabey, C., and Major-White, B. (eds) Managing change. Second edition, Paul Chapman Publishing, London, England, 108-112.
Robbins, H. and Finley, M. (1998). Why teams don't work. Orion Publishing. London, England.
Robinson, H. and Sharp, H. (2004). The characteristics of XP teams. in Eckstein, J., and Baumeister, H. (eds) Extreme programming and agile processes in software engineering. 5th International Conference. Germany, Springer-Verlag. Berlin, Germany, 139-147.
Sawyer, S. and Guinan, P. (1998). Software development: Processes and performance. IBM Systems Journal, 37(4), 552-569
Schuh, P. (2004). Integrating agile development in the real world. Delmar Thomson Learning, NY, USA
Schwartzman, H. (1993). Ethnography in organizations. Sage Publications, CA, USA.
Scott, W.R. (1995). Institutions and Organizations. Sage Publications Ltd., Thousand Oaks, CA.
Spradley, J. (1980). Participant observation. Holt, Rinehard, and Winston. NY, USA.
Stephens, M. and Rosenberg, D. (2003). Extreme Programming refactored: The case against XP. Springer-Verlag.,Heidleberg, Germany.
Stewart, G., Manz, C. and Sims, H. (1999). Team work and group dynamics. John Wiley and Sons, NY, USA.
Thompson, L. (2003). Improving the creativity of organizational work groups. Academy of Management Executive, 17(1), 96-111.
Triplett, N. (1897). The dynamogenic factors in pacemaking and competition. in Coats, E., and Feldman, R. (2001) Classic and contemporary readings in social psychology. Third Edition. Pearson Education, NJ, USA, 3-13.
Wastell, D. (1999). Learning dysfunctions in information systems development: overcoming the self defences with transitional objects" MIS Quarterly, 23(4), 581-600.
Wenger, E. (1998). Communities of practice: Learning as a social system. The Systems Thinker, 9(5), 1-5.
Wenger, E. and Snyder, W. (2000). Communities of practice: The organizational frontier. Harvard Business Review, 78(1), 139-145.
Williams, K., Karau, S. and Bourgeois, M. (1993). Working on collective tasks: Social loafing and social compensation. in Hogg, M., and Abrams, D. (eds) Group motivation: Social psychological perspectives. Harvester Wheatsheaf, Hertfordshire, UK, 130-148.
Williamson, O.E. (1985). The Economic Institutions of Capitalism. The Free Press: New York.
Wright, G. (2003). Achieving ISO 9001 certification for an agile company. in Maurer, F., and Wells, D. (eds) Extreme programming and agile methods – Xp/Agile universe 2003. New Orleans. Springer-Verlag, Berlin, Germany, 43-50.
Yourdon, E. (1985). Impact of the computer revolution: 1985-2001. Proceedings of the 1985 ACM thirteenth annual conference on Computer Science. New Orleans, ACM Press. NY, USA, 23-28.
Zajonc, R. (1965). Social facilitation. in Coats, E., and Feldman, R. (2001) Classic and contemporary readings in social psychology. Third Edition, Pearson Education, NJ, USA, 272-280.