The Dilution of Effort in Self-Evaluating Development Teams: Agile Loafing

John McAvoy and Tom Butler
University College Cork, Ireland
j.mcavoy@ucc.ie
tbutler@afis.ucc.ie

Abstract: Attempts to resolve the problems in software development have concentrated on the tools and methodologies used, despite an acceptance by many that it is a sociological problem. An example of this is the procedures and processes surrounding evaluations within projects, yet ultimately it depends on individuals more than process. This paper examines one of the sociological factors inherent in a software development team to determine its impact on evaluation within a project. Social loafing occurs where individual members of a team demonstrate a tendency not to work as hard as they could or should. This "slacking off" occurs because the team provides a degree of anonymity – the individual feels their lack of work will be hidden from evaluation within the overall output of the team. Some authors purport that Agile Software development teams have low incidences of social loafing (though these are opinions rather than research findings); the contrary can also be argued. An examination of the philosophy behind Agile Software Development, demonstrated by the Agile Manifesto, highlighted the possibility of occurrences of social loafing brought about by the Agile values. Agile espouses the importance of cohesive teams, the empowerment of these teams, and the collective ownership and self-evaluation of work by the team. These values map onto factors which are described as affecting social loafing. An investigation of two teams over an eight month period examined if the Agile values could lead to incidences of social loafing, specifically when their work is being evaluated The investigation determined that the opposite was actually the case. This paper then goes on to determine why the findings go against the initial hypothesis and to show the impact this can have on those evaluating software development projects.

Keywords: teams, agile software development, social loafing, self-evaluation, participant observation, sociological factors

1. Introduction

The development and evaluation of Information Systems relies heavily on the individual developers, yet the main area of concentration appears to be on the methods and tools used in a project. Code reviews, or inspections, are an example of this concentration, where procedures have evolved or have been proposed; an example of this is a statistical process for control of code reviews proposed by Nelson and Schumann(2004). Many authors accept that there is a need to refocus on the teams of developers rather than the processes they follow. Martin (2003, p.4) describes it best by stating that "*a good process will not save the project from failure if the team doesn't have strong players*." Strong players on their own, though, will not guarantee success. Teams of software developers bring with them problems in how the team work and how they evaluate their own work. One particular concern with teams is the possibility, if not prevalence, of social loafing in teams. Social loafing occurs when an individual lessens their effort when in a group or team. A developer may lessen their effort because they perceive that other members of the team will take on the extra load, or they may believe that the team gives them anonymity as the team is being evaluated as a unit rather than the individual being evaluated.

A review of existing literature on social loafing highlighted a potential problem for development projects using an Agile Methodology. The Agile methodologies espouse empowering cohesive groups to take collective ownership of their work and responsibility for evaluating that work. These map directly onto factors that are described as likely to cause social loafing. This study describes two case studies, undertaken by the authors, that examined the occurrence of social loafing in team evaluations. The two cases were projects that were based on the Agile philosophies, yet one project's application of the Agile methods was diluted by the need to adhere to company standards and processes (for code reviews in this case). Based on existing literature, it was hypothesised that the project that more closely followed the Agile philosophy would be more likely to demonstrate occurrences of social loafing during evaluations. After eight months of observation and interviews with key informants, doubts were cast on this hypothesis.

In the next section, a review of existing literature on Agile software development is presented, along with its approach to evaluation within projects. The phenomenon of social loafing is then explained,

Reference this paper as:
McAvoy, J and Butler, T (2009) "The Dilution of Effort in Self-Evaluating Development Teams: Agile Loafing" *The Electronic Journal Information Systems Evaluation Volume 12 Issue 2 2009, (pp141 - 152), available online at www.ejise.com*

and the possibility of its impact on evaluations within Agile projects is then examined. The research approach to determine if this impact exists is then presented, along with a description of two case studies where the teams' evaluations of their work were examined. Finally conclusions are presented with suggestions as to how social loafing can impact on evaluations within software development projects and proposals as to how these can be mitigated or taken into account.

Note that, throughout this paper, the terms group and team are both used. While the words group and team are often used interchangeably, the distinction is that a group is a collection of individuals with a common aim, whereas a team has the additional attribute where the skills of each member fit in with those of the others towards a specific purpose. This research concentrates on teams based on the definition above, but the word 'group' is used herein when describing previous research that examined groups as a collection of individuals rather than being specifically restricting to teams.

2. Social loafing in an agile environment

While 'software crisis' may be extreme, the term was first used at a NATO conference in Germany in 1968 (Hazzan and Tomayko, 2003), and "*despite impressive technical advances in tools and methodologies and the organizational insights provided by many years of research, IS failures remain all too common*" (Wastell, 1999, p.582). The response to these failures has been an attempt to introduce engineering principles to software development, and these principles are visible in a wide variety of methodologies in use. Where evaluating code, Fagan inspections, first described in Fagan (1999), are a well used example of using a process as an engineering fix. Despite this work, the problems in software projects still remain; studies have continually shown that improvements resulting from the introduction of these advances have been disappointing (Finnegan and Murray, 1999, p 91). Glass (2003) agrees that the improvements have been disappointing, regarding much of the promises of increased productivity as hype. Bahli and Buyukkurt (2005) acknowledge the importance of teams in information systems development, yet state that there has been little research to date. Specific to evaluating code, Rodgers *et al*. (1998) acknowledge the positive influence teams have on code evaluations yet ultimately propose further tool development. Sawyer and Guinan (1998) argue that software development needs to be refocused from the production elements to the social aspects of development. As determinants of software development performance, methodology and tool use have less of an affect than the socialization of developers in the team. Domino et al. (2003, p.44) put it succinctly - "*software development is a human endeavour*"

This research concentrates on the social aspect of software development, specifically examining the software development team and the impact that the team can have on the development process and the evaluation of the output. In doing so, we concur with the view of Jones and Hughes (2003) that "*IS evaluation is a socially embedded process in which formal procedures entwine with the informal assessments by which actors make sense of their situation*"

2.1 The importance of teams to Agile software development.

Argyle (1989) describes how work is predominantly performed in groups. Software development is no different; teamwork is the basis of software development projects. Martin (1991, p.155) describes the common view that "*better team working leads to better performance*", while Jones and Roelofsma (2000, p.1129) state that "*teamwork is essential if competitive advantage is to be achieved*." The positive endorsement of teams by researchers has been translated into practice. Research by Finnegan and Murray (1999) showed that practically all organisations develop software in teams. More 75% of all development is performed in teams in 98% of organisations.

Agile software development concurs with this belief in the value of teams, placing teams at the centre of the agile methodologies. Highsmith (2004) emphasises the importance of a good team for the success of agile projects, while Hazzan and Tomayko (2003) describe XP (eXtreme Programming – one of the most commonly used Agile methods) as being based on team interaction – more so than other software development methodologies.

The basic principles of agile methodologies are qualified in Abrahammson *et al*. (2002) 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

These principles are referred to as the 'Agile Manifesto' and form the core values of the Agile methodologies. The first principle of Agile software development highlights the importance of teams (the interaction of individuals) to agile software development. The importance of teams is seen by the devolution of power to the software development teams and the expectation that the team as a unit is responsible for development. Code is collectively owned (Hazzan and Tomayko, 2004), fostering a democratic approach with regard to responsibilities. Allied with this collective ownership is 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). The team is also expected to evaluate their own work, and this is one of the unique areas of Agile. Rather than the traditional code reviews to evaluate work, Agile proposes that the team self-evaluate using a variety of methods. Pair programming is one such method, where while one developer codes, a second developer inspects the code alongside the first developer. In effect, this is code evaluation in real-time, which Cockburn (2001) describes as a continuous code review. Even if the pair-programming process is not used by the team, the emphasis has moved from traditional formalised evaluations to informal self-evaluations by the team; the concept of self-evaluation by Agile teams is noted in Turk *et al*. (2005). To allow this level of empowerment, there is a requirement for well functioning cohesive teams. Auer *et al*. (2003) describe this as Agile's need for an effective social network, Boehm and Turner (2003) describe it as the need for good interaction, while Highsmith (2004) talks about jelled teams.

2.2 From Agile to social loafing

In the 19th Centruy, Triplett (1897) found that the mere presence of others, cyclists in his experiment, improves performance – the dynamogenic theory. Zajonc (1968) found that the performance of other tasks than the physical ones investigated by Triplett improved in the presence of others; the term he uses to describe this is social facilitation. Interestingly, Baron *et al* (1999) describe how Zajonc found the same effect in other animals, including cockroaches – although it is not the aim of this research to compare software developers to cockroaches!

While the positive benefits of teams is noted by many researchers, research into groups has often produced paradoxical findings (Baron et al., 1999). Researchers have noted the positive and negative aspects of teams, often based on the same attribute where one researcher notes its positive impact and another notes its negative impact. So, while teamwork brings obvious benefits, it should be noted that it can also bring problems. Nunamkaer *et al*. (1997) propose that, while teams are vital in many situations, teams bring with them their own problems. For example, a team of four people will not perform four times better than one individual. This shortfall in the team's performance is described as process loss, where a team does not act in the most effective way. Other forms of loses in team activity include coordination loss, where the team does not effectively coordinate the work, and motivational loss, where the team members do not try as hard as they would as individuals (Baron et al, 1999). 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 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, described above, brings positive effects to teams, its opposite – social loafing – will reduce a team's performance. "*Social loafing is 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, 2003, p100). Human nature dictates that, as individuals, we are all capable of slacking off, given the opportunity. The ability to hide in a team, where our lack of effort may not be noticed, gives us this opportunity. Brooks and Ammons (2003) use the term free riding to describe social loafing, identifying its prevalence in group based projects in education. It should be noted, though, that Mulvey and Klein (1998) differentiate social loafing and free-riding. Free-riding, although very similar to social loafing, involves the perception of a team member that other team members will put in sufficient work, making their own contribution less necessary. Mulvey and Klein do accept, though, that the terms are generally used interchangeably.

Moving from the general principle for all teams, to the specifics of Agile teams, it appears that the Agile principles should mitigate against, if not completely eliminate, social loafing in teams. Rising and

Janoff (2000) argue that the small teams inherent in Agile projects should mitigate against social loafing, while Whitworth and Biddle (2007) argue that increased awareness of accountability to the Agile team should also lower the incidences of social loafing. Based on this, social loafing should not be a problem in Agile teams, but, in both of the research above, these are offered opinions rather than research findings. In fact, the opposite could also be argued, as discussed below.

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

- Social loafing increases when evaluation of the work is based on team, rather than individual performance.
- Social loafing is less likely to occur if the work is interesting.
- Group cohesiveness can reduce social loafing (also noted in Liden *et al*. (2004)).Williams *et al*. (1993) and Mulvey and Klein (1998) did find, though 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 the trusting team, allowing the opportunity for loafing (this appears pertinent to Agile teams).

Pearce and Ensley (2004) add further to this by stating that role ambiguity is a further cause of social loafing while Landy and Conte (2004) 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.

The agile methodologies stress the empowerment of, and trust in, the team, to the extent that the team monitors itself to a large extent, and evaluates its own work. The agile team is responsible as a unit for the development of the user requirements, so the group is evaluated as a whole, as opposed to individual evaluation of the team members. Collective ownership, for example the collective ownership of code, allows for a degree of anonymity for the individual developers. Further, the requirement of a cohesive team is often noted as a necessity for an agile team. These traits of agile software development match closely two of the three factors which affect social loafing: group evaluation and group cohesiveness. The second factor of social loafing – interesting work – does not appear to be restricted specifically to agile methodologies as there is nothing in the agile philosophy which addresses how agile development would be more or less interesting than traditional development.

3. Research approach

A longitudinal study of the development project was undertaken, using participant observation of two agile development teams as its primary method. The study concentrated on how the teams evaluated their work, while noting other factors which may be impacting on this self-evaluation. The first project concerned the design and development of a knowledge management system for a European government organisation. A team of seven developers and one project manager were involved in the project, the first phase of which lasted eight months. The second team, of eight developers, was based in a large multinational telecommunications company, developing fault tolerant applications. The choice of similar team size for both cases was deliberate. The aim was to eliminate a further factor of social loafing where there is a correlation between an increase in social loafing and an increase in the size of the group (Williams *et al.*, 1993). As both teams were of similar size, the size of the team could be removed as a contributing factor. Further, the team sizes were the same as those studied by Clutterbuck *et al*. (2009) who also examined evaluation in an Agile setting.

Participant observation was chosen as the method for identifying the presence of social loafing. Participant observation is relevant where "*the phenomenon is obscured from the view of outsiders*" (Jorgensen, 1989, p.12). Asch (1952) goes further than this, by arguing that observations of groups can discover areas that even group members themselves are unaware of. Social loafing may be an effect that an individual, or team, are unaware of.

The first development team was observed over an eight-month period. Both researchers had roles in the team. In longitudinal research, observations lasting weeks or months allow the researcher to develop a relationship with those being studied, which goes beyond a superficial short-term relationship (Gurney, 1991; Fetterman, 1991). Describing those performing research as participant observers, Burgess (1982) argues in that their activities involve sharing "*in the lives and activities of those whom they study and take roles which are effective in the setting under study*." Integration into

the group under study is vital in participant observation (Ezey, 2003). "*Participation 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).

As in this study, others have used participatory observation to investigating agile software development projects. Martin *et al*. (2004) describe interpretative in-depth case studies as the best method of investigating agile software development. A qualitative approach was used in another investigation of the characteristics of an agile team, involving participant observation, described in Robinson and Sharp (2004). Examining two specific cases provided insights that may not be visible in other research methods. A benefit of the trust gained by the researcher is the ability to examine areas that may be secret or concealed. It is the "*taken for granted*" that is worth observing, for it is the seemingly trivia of daily work that influence organisations (Schwartzman, 1993, p.4). It is this everyday life that provides a description of reality (Jorgensen, 1989).

The second team was observed by one of the researchers, with a working role in the team, and supplementary details were elicited from two key informants in the team – a developer and a team leader - in monthly meetings. There is considerable justification for the use of key informants - examples include: Kumar et al. (1993), Schwenk (1985), and Holloway and Tordres (2003) - and key informants have been used to examine social influences (Jasperson *et al*., 1999) and projects (Van Fenema, 1997).

4. Observations and analysis

The first software development team studied were involved in the development of a knowledge management system for use in a Government department. The design of this Information Systsem was informed by the experiences of the team in developing a Knowledge Management System (KMS) for the United Nations. The majority of the team had worked together on the previous project and was highly rated by the customer. Despite the success of the United Nations project, the team decided that it needed to introduce a more formalised approach to development, while ensuring that the approach was not overly rigid. Based on this criteria they chose to adopt and Agile approach to development.

The second case chosen for study was a large US multinational telecommunications company, which designs and manufactures a range of products, from cell-phone switches and base stations, to mobile telephone products. Its software division develops software used in the monitoring, control and operation of mobile phone networks throughout the US, Europe, the Middle East, and Africa. It is significant that the organization's policies and culture recently promoted software process improvement initiatives, such as Agile approaches. The team developed system administration applications for the installation, upgrade, and maintenance of a large application which monitored mobile phone networks.

The two cases were chosen because, while both had adopted an agile approach, they were sufficiently different. The knowledge management project team followed the agile philosophy to a greater extent than the telecommunications project team. The telecommunications project team used a diluted version of agile, as they had to ensure compliance with company standards. TL9000 is a communications standard, which Clancy (2002) describes as necessity for telecommunications companies, which was used by the organisation. In addition, as the telecommunications company was regularly audited for CMM accreditation, there were a variety of processes that the project team had to follow. Of interest to this research were the processes and procedures concerning evaluation of work. The primary process here were Fagan inspections (described in Fagan (1999)), which was the process for the evaluation of code and the software as a whole. This went somewhat against the agile philosophy of individuals and interactions over processes and tools, but it was still considered to be an agile project; others describe code inspections taking place in an Agile project (cf. Fitzgerald, Hartnett and Conboy, 2006). Various authors would agree that, although the agile philosophy had to be altered to take the enforced company processes and standards and into account, this still constitutes an agile approach. The two cases therefore present two views of an agile approach: an adoption of the 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 hypothesised that the more formalised, monitored, and audited telecommunications project should demonstrate a lower likelihood of social loafing. The researchers had access to both formal and informal meetings where the teams evaluated their work. Field notes were kept during the period

of the project, which was later analysed for the prevalence of social loafing in the knowledge management project, yet this was, in fact, not what was observed. Evaluation was observed at two levels within the project: evaluation of the team by others, and the self-evaluation by the team of their work. These were seen in two major aspects of both projects:

- The evaluation of and by the teams through bug reports
- The evaluation of and by the teams through reviews of the work done

4.1 Evaluation of the team through bug reports

Fixing bugs, and the evaluation of this work, was handled differently in both projects. The telecommunications team worked within the company process of PR (Problem Report) lists. The team had a list of PR's, based on their project, which was maintained centrally by the project management office; the project management office had overall responsibility for multiple projects within the company. As such, they monitored the team for the number and severity of PR's; each PR was assigned to an individual developer who was the assigned "owner" of the relevant code or feature. This was due to existing processes in the company, prior to the team adopting Agile, but was not used to evaluate individuals. The Quality group within the organisation was supportive of the team's adoption of the Agile philosophy of collective ownership, yet still maintained PR lists against individuals as this was the policy in the other, non-Agile, teams in the company; the quality group, though, agreed with the principle of PR's being the responsibility of the team as a whole.

Despite agreement from the Quality group that individual evaluation of PR numbers against individuals was a formality rather than a process for the team, it appeared to have a different affect than intended. The PR list contained all PR's for the team and the principle agreed by the team was that any developer could fix any PR. It was accepted that certain PR's were better fixed by certain individuals, but on average this would still have been spread evenly across the developers. Over the timeframe of the research, it became clear from monitoring the list, which the researcher was given access to, that the principle of collective ownership was not strictly adhered to. The norm throughout the project was that developers fixed the PR's that were listed against their name. Developers did fix PR's assigned to other team members, but this was the exception rather than the norm. If a developer was on holiday, sick, or temporarily unavailable, another team member would take the PR and work on it. There was no formal, or informal, rule as to when to work on another's PR, but the developers themselves were aware of when their colleagues were unavailable and took ownership.

It was initially unclear as to why the developers only took on another's PR if they were absent. They were willing to cover an absent colleague, yet not to take collective ownership at other times. It became clear, over time, that the team manager was having an impact. When the project management office queried the team, through the team manager, about the number of PR's, the team manager used the list to determine who to ask about an individual PR. Even though technically there was collective ownership of all PR's, having names assigned to PR's directed the team manager to each individual. As this happened regularly throughout the project, the developers came to recognise that the only time they would be evaluated on a PR was when their name was assigned to it (even if the naming was only a formality). The team manager was asked about this and he seemed genuinely unaware of what he was doing. He accepted the concept of collective ownership and merely used the names in the PR list as a simple way of finding out what was happening with a PR. This ultimately lead to a form of social loafing within the team, even if was not at a conscious level. It was relatively easy to "hide behind" the PR list rather than take collective ownership. The developers felt that the only evaluation of their work on PR's was based on the list with names assigned (even though the name on the list was a formality).

The Knowledge Management team had a different approach to the evaluation of the bug list. This listed bugs in the code, yet was not assigned individually against a developer. Bugs were, in general, fixed by whichever developer was available at the time. On occasion, certain bugs were informally assigned to a particular developer as they would have been assumed to have been the expert in the particular area. There were no formal rules as to when this assignment would occur, and was generally based on a developer requesting help, as was seen in the following conversation.

> *Developer 1: Did you write the code that checks for valid input into this function.*
>
> *Developer 2: Yes, I wrote that.*

Developer 1: Can you explain how this section works as there is a bug in it I am looking at.

Developer 2: Actually I will take that. I think I know what the problem is.

Conversations, such as this, were common and lead to different developers concentrating on different areas of the code, but it did not restrict others from working problems in those areas. An interview with the tester confirmed observations that the bug fixing load was fairly spread amongst the team. He acknowledged that developers worked bugs that they themselves did not "own" and even volunteered to take work from a team member who appeared overloaded. The tester mentioned that this was because the developers did not want to let the team down by not taking their fair share of bug fixing work. At meetings, where bugs were discussed, individual developers were not associated with a PR. The project manager discussed the bugs with all team members and asked for information from the whole team as opposed to individuals. It was clear to all developers, and the project manager in many cases, who "owned" the PR, but the team, rather than the individual, was held responsible and evaluated as such. Over the period of the research, it was clear that, each developer took an equal share of responding to queries about bugs from the project manager. At times, certain developers seemed to have more input into these meetings, but this averaged out over the project.

Despite previous research suggesting that team based evaluation would lead to social loafing, the cohesion of the team mitigated against this and lead to the situation where allegiance to the team overcame tendencies towards loafing.

4.2 Evaluation of work through reviews

The team members of the knowledge management project demonstrated that the collective ownership of the project did not have a detrimental effect. A specific example during the development of the graphical user interface highlights how the individual members of the team did not slack off (to use the term of Thomson (2003)). The project manager had a concern with the design of the graphical user interface. At an informal meeting where the team evaluated their work, the team gathered to evaluate the design of the graphical user interface. If social loafing had occurred, the developers could have "hidden" within the group as the GUI was developed by the entire team – collective ownership. Rather than hiding within the group, one developer demonstrated a degree of ownership of the GUI by disagreeing quite forcefully with the opinion of the project manager. There was no need for this developer to "stand out from the crowd" as he was not the sole owner of the GUI, yet he did so. Further to this, the other developers in the team joined in the defence of the GUI. Each developer defended the GUI as a collective, rather than individuals hiding within the group. The argument itself was quite forceful, with raised voices, and continued for nearly an hour. When asked about this disagreement, a few days after the event, one developer said that he felt that "our work was being called into question." It is interesting to note the use of the word "our" as it highlights that the team defended the work of all the team members during the evaluation. Analysis of the field notes showed further examples of this "team defence", although the GUI incident above was the most vociferous argument – other disputes tended to be more amicable. This collective defence also highlights a high degree of cohesion among the team members. 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 during an evaluation of their collective work, the observations demonstrated the opposite.

It was anticipated at the start of this research that social loafing would be limited in the telecoms by the presence of individual versus group evaluation (as was required by company processes and standards). Added to this was the lower cohesion in the telecommunications project team that also should have reduced the likelihood of social loafing. It should be noted that that this team was a cohesive team, it simply did not have as much autonomy as the first team. Again, the observations, and interviews with key informants, did not show a reduction in social loafing as expected. Code reviews were highlighted as a potential example of social loafing when it was suggested by the author that there should be group ownership of the code reviews, which is more in line with the agile philosophy. Traditionally, code reviews assisted a particular developer, but problems with the code were evaluated against this individual rather than the team. The response to the suggestion of group ownership of code reviews was met with extreme disapproval. The author, in a team meeting, made

the suggestion and the project team members shouted down the suggestion. Based on their response to the suggestion, the team leader immediately dropped the idea. The author, though, noted the response and further investigated the code review process.

Social loafing was quite apparent in this process (and was quite probably the reason why the team members did not want to change the process.) While the team always assisted the developer who would be evaluated on the code, their assistance was limited to the duration of the code review. It was noted that the responsible developer's colleagues rarely read the code before the meeting – most could clearly be seen reading the code and marking problems during the meeting. They were assisting their colleague but only to the extent that they had to. Their role was to evaluate the work and provide feedback and suggestions to their colleague, which they did, but as their work prior to the review was not seen, it was not a whole-hearted effort. A team lead, or technical lead, would often be in attendance at a code review so the developer's colleagues performed what was required of them at the review. They were in effect being evaluated on their performance at the review, by the team lead, so they performed as expected during the review (in fact the work in these reviews was often exceptionally good). As their work prior to the review was not seen (and therefore not evaluated), they did not appear to put in the work beforehand.

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.

A further example of loafing was seen in the telecommunications team and, again, it was related to processes. The team leader key informant admitted that he was influenced by his discussions with the author on social loafing and became more aware of it in his team. An observation made by the team leader was that he noticed that the developers sometimes "hid behind" the company processes. The developers sometimes (and he stressed sometimes) used the company's processes as a method of avoiding evaluation. If a developer was late in delivering some required functionality, the company's processes could be blamed. The team leader described this as akin to the statement "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 evaluate and 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 the social problems that occur in development team evaluation and self-evaluation. Nor does it posit that formalised standards and processes for evaluations will give rise to these social problems. The longitudinal aspect of the overall research (much of which lies outside of the domain of this paper) showed the benefits and problems with both formalised and Agile approaches. What is clear, though, is that they can have an impact on social loafing, and the effectiveness of evaluations of, and self-evaluations by, the team. Again, the use of the word "can" is important. It is impossible, even with the benefits of longitudinal research, to make generalisations from two case studies. We restrict our findings to state that the choice of a formal or an Agile approach can impact the incidences of social loafing in evaluations and self-evaluations. The two main findings of this paper are:

- A highly cohesive software development team, with a sense of ownership of their work, is less likely to demonstrate social loafing. The Agile software development methodologies promote this cohesion and sense of ownership. This leads to more effective self-evaluation of work within the project.
- Individual monitoring and evaluation of software developers can have a negative impact on the team. Previous research predicts that individual, versus group, evaluation will lower social loafing. What was actually found was that individual evaluation allowed those not the focus of the evaluation to engage in social loafing. The Agile software development methodologies advocate the evaluation of the entire group as a unit.

Based on these two findings, how teams are evaluated and how they evaluate themselves are impacted by the use of Agile methods. The use of Agile allows for better evaluations by the team of their own output. Conversely, how a team is evaluated is also impacted by the use of Agile. Evaluation of the team, as opposed to the individual, reduces the ability to hide within the team.

The Agile philosophies of empowerment and group ownership appear to be overcoming a tendency towards social loafing during evaluations. The team that fully adopts an Agile philosophy tend to work better as a unit by supporting each other rather than hiding within the group. The ability to remain anonymous within the group (a diffusion of responsibility) appears to be mitigated by the cohesion of the group. 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 and evaluation 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 and self-evaluating team can exert a greater level of control over an individual than the traditional command and control management structure. Development processes that strive to ensure adherence, by monitoring and evaluating the 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. So how we evaluate our work, and how others evaluate us, is impacted positively by the cohesive and empowered teams espoused by Agile. For those managers ultimately responsible for evaluating developers and their work, the move to empowering the team and evaluating the team collectively, can bring positive benefits to a project.

References

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

Argyle, M. (1989) *The social psychology of work*, Penguin Books, Middlesex, England.

Asch, S. (1952) *Social Psychology*, Prentice-Hall, NJ, USA.

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, pp35-42.

Bahli, B., and Buyukkurt, M. (2005) Group performance in information systems project groups: An empirical study. Journal of Information Technology Education. Volume 4. pp97-113.

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, pp126-158.

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

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, pp1-8.

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*, Volume 78, No. 5, pp268-272.

Burgess, R. (1982) Some role problems in field research, In Burgess, R. (Ed) *Field research: A sourcebook and field manual,* George Allen and Unwin Publishers, London, UK, pp45-49

Clancy, B. (2002) Are there real benefits from TL9000. *Quality Times*, Volume 1, No. 3, pp7-9.

Clutterbuck, P., Rowlands, T. and Seamons, O. (2009) A case study of SME web application development effectiveness via Agile methods, *Electronic Journal of Information Systems Evaluation*, Volume 12, Number 1, pp13-26.

Cockburn, A. and Williams, L. (2001) The costs and benefits of pair programming, In Succi, G. and Marchesi, M. (Eds) *Extreme Programming examined*, Pearson Education, NJ, USA, pp223-243.

Cohn, M. (2004) *User stories applied for agile software development*. Addison-Wesley. MA, USA.

Domino, M., Collins, R., Hevner, A. and Cohen, C. (2003) Conflict in collaborative software development, 2003 SIGMIS Conference on Computer Personnel Research ACM Press, Philadelphia, USA, pp44-51.

Ezey, P. (2003) Integration and its challenges in participant observation. *Qualitative Research*, Volume 3, Issue 2, pp191-205.

Fagan, M. (1999) Design and code inspections to reduce errors in program development*, IBM Systems Journal,* Volume 38, No. 2-3, pp258-287.

Fetterman, D. (1991) A walk through the wilderness: Learning to find your way, In Shaffir, W. and Stebbins, R.(Eds) *Experiencing fieldwork: An inside view of qualitative research*, Sage Publications, CA, USA, pp87-96.

Finnegan, P., and Murray, J. (1999) Managing IS Staff: The key to improved systems development. in Adam, F., and Murphy, C. (eds) *A mangers guide to current issues in information systems*, Blackhall Publishing, Dublin, Ireland, pp 91-103.

Fitzgerald, B., Hartnett, G. and Conboy, K. (2006) Customising agile methods to software practices at Intel Shannon, *European Journal of Information Systems*, Volume 15, No. 2, pp200-213.

Glass, R. (2003) *Facts and fallacies of software engineering*, Pearson Education, MA, USA.

Gurney, J. (1991) Female research in male-dominated settings: Implications for short-term versus long-term research, In Shaffir , W. and Stebbins , R (Eds) *Experiencing fieldwork: An inside view of qualitative research*, Sage Publications, CA, USA, pp53-61.
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, pp51-61.
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 Conferenc*e. Germany, Springer-Verlag, Berlin, Germany, pp303-311.
Highsmith, J. (2004) *Agile project management*. Pearson Education. MA, USA.
Holloway, I. and Todres, L. (2003) The status of method: flexibility, consistency and coherence, *Qualitative Research*, Volume 3, No. 3, pp345-357.
Jasperson, J., Sambamurthy, V. and Zmud, R. (1999) Social influence and individual IT use: Unraveling the pathways of appropriation moves, *20th International Conference on Information Systems Association for Information Systems*, North Carolina, pp113-118.
Jones, S. and Hughes, J. (2003) An exploration of the use of grounded theory as a research approach in the field of IS Evaluation, *Electronic Journal of Information Systems Evaluation*, Volume 6, Number 1.
Jones, P. and Roelofsma, P. (2000) The potential for social contextual and group biases in team decision-making: biases, conditions and psychological mechanisms. *Ergonomics*, Volume 43, No. 8, pp1129-1152.
Jorgensen, D. (1989) *Participant observation: A methodology for human studies*, Sage Publications. CA, USA.
Kumar, N., Stern, L. and Anderson, J. (1993) Conducting inter organizational research using key informants, *Academy of Management Journal*, Volume 36, No. 6, pp1633-1651.
Landy, F., Conte, J. (2004) *Work in the 21st century*. McGraw-Hill, NY, USA.
Liden, R., Wayne, S., Jaworski, R. and Bennet, N. (2004) Social loafing: A field investigation, *Journal of Management*, Volume 30, No. 2, pp285-304.
Martin, R. (1991) Working in groups. in Smith, M (ed) *Analysing organisational behaviour*. Macmillan Press. London, UK. pp154-177.
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, pp51-59.
Mulvey, P., Klein, H. (1998) The impact of perceived loafing and collective efficacy on group goal processes and group performance. *Organizational Behaviour and Human Decision Processes*. Volume 74, No. 1. pp62-87.
Nelson, S. and Schumann, J. (2004) What makes a code review trustworthy?, *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, Hawaii, IEEE, NJ, USA.
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*, Volume 13, Number 3, pp163-207.
Pearce, C., 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*, Volume 25, No. 2, pp259-278.
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, pp108-112.
Rising, L. and Janoff, S. (2000) The scrum software development process for small teams, *IEEE Software*, Volume 17, No. 4, pp26-32.
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, pp139-147.
Rodgers, T., Vogel, D., Purdin, T. and Saints, B. (1998) In search of theory and tools to support code inspections, *Proceedings of the 31st Hawaii International Conference on System Sciences*, Hawaii, IEEE, NJ, USA, pp370-378.
Sawyer, S., Guinan, P. (1998) Software development: Processes and performance. *IBM Systems Journal*, Volume 37, No. 4, pp552-569
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.
Schwenk, C. (1985) The use of participant recollection in the modelling of organizational decision processes, Academy of Management Review, Volume 10, No. 3, pp496-503.
Spradley, J. (1980) *Participant observation*. Holt, Rinehard, and Winston. NY, USA.
Thompson, L. (2003) Improving the creativity of organizational work groups. *Academy of Management Executive*, Volume 17, No. 1, pp96-111.
Triplett, N. (1897) The dynamogenic factors in pacemaking and competition, *American Journal of Psychology*, Volume 9, Number 4, pp507-533.
Turk, D., France, R. and Rumpe, B. (2005) Assumptions underlying Agile software development processes, *Journal of Database Management*, Volume 16, No. 4, pp62-87.

Van Fenema, P. (1997) Coordination and control of globally distributed software development projects: The GOLDD case, *8th International Conference in Information Systems Association for Information Systems*, Atlanta, pp474-475.

Wastell, D. (1999) Learning dysfunctions in information systems development: overcoming the self defences with transitional objects, *MIS Quarterly*, Volume 23, No. 4, pp581-600.

Whitworth, E. and Biddle, R. (2007) The social nature of agile teams, *Proceeding of Agile 2007*, Washington DC. pp26-26.

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, pp130-148.

Zajonc, R. (1968) Social facilitation. in Cartwright, D., and Zander, A. (eds) *Group dynamics*. Harper & Row Publishers. NY, USA, pp63-73.

