

Preparing Small Software Companies for Tailored Agile Method Adoption: Minimally Intrusive Risk Assessment



Research Section

Philip S. Taylor¹, Des Greer^{1*,†}, Gerry Coleman²,
Kevin McDaid² and Frank Keenan²

¹ School of Computer Science, Queen's University Belfast, BT7 1NN,
Northern Ireland, UK

² Computing and Maths Department, Dundalk Institute of Technology
(DKIT), Co. Louth, Ireland

There is often a misconception that adopting and tailoring agile methods is straightforward resulting in improved products and increasingly satisfied customers. However, the empirical nature of agile methods means that potential practitioners need to carefully assess whether they are exposed to the risks that can make agile method adoption problematic. This is particularly the case with small software companies who are less able to absorb the impact of failed experimentation. This study describes a minimally intrusive assessment approach for small software companies preparing for agile method adoption and tailoring in the light of key risks. The approach has been conducted with six small software companies, three of which are presented to show the evolution of the approach, describe the resource commitment that companies have to make, and highlight the type of information generated from an assessment. The contribution of this study is that small software companies have an alternative to 'mere experimentation' with agile methods and can take reasoned steps towards their adoption and tailoring. Copyright © 2007 John Wiley & Sons, Ltd.

KEY WORDS: agile software development; software process assessment; small software companies; software risk assessment

1. INTRODUCTION

The relationship between software process quality and software product quality is complex due to the unique non-repetitive characteristics of software development projects (Trudel *et al.* 2006). The IEEE

(1991) defines software quality as: (i) the degree to which a system, component, or process meets specified requirements and (ii) the degree to which a system, component, or process meets customer or user needs or expectations. The problem with this definition is that many software products are being developed with emergent requirements and with constantly changing customer or user needs and expectations.

Such software products are difficult to manage using defined process control and tend to be

* Correspondence to: Des Greer, School of Computer Science, Queen's University Belfast, BT7 1NN, Northern Ireland, UK

†E-mail: des.greer@qub.ac.uk



more manageable with empirical process control (Schwaber 2004) involving feed-forward and feedback mechanisms (White 2006). There are three foundational aspects to empirical process control for software development projects. First, *visibility* means that process disturbances must be visible to those controlling the process. Second, *inspection* of the process and output must be frequent so that process disturbances can be detected. Third, the process and/or product must be *adapted* if necessary. Agile methods are contemporary examples of empirical processes supporting visibility, inspection and adaptation.

The need for visibility, inspection and adaptation is not merely one of frequent requirements change (Lee and Xia 2005). Van Oosterhout *et al.* (2006) define business agility as being able to swiftly change businesses and business processes beyond the normal level of flexibility to effectively manage unpredictable external and internal changes. Overby *et al.* (2006) similarly state that this level of agility involves sensing environmental change and responding readily. Unpredictable business-level change factors are felt by groups such as software product development, irrespective of an internal software development group or an independent company. Software development groups also have their own widespread problems such as schedule, budget and effort overruns (Keil *et al.* 2000, 2003, Moløkken-Østfold and Jørgensen 2005).

In the last 5 years agile methods (Highsmith 2002, Abrahamsson *et al.* 2003) have been gaining popularity as approaches to avoid software project failure causes. Ceschi *et al.* (2005) provide indicative results suggesting that adopting agile methods improves management of the development process and customer relationships and Syed-Abdullah *et al.* (2006) indicate that agile methods can lead to more enthusiastic development teams. However, agile methods do not suit every problem domain and they require careful strategising to determine how much agility and planning is needed for each project (Boehm and Tuner 2004, Taylor *et al.* 2006).

Agile methods also have a context within the various approaches to software development (Larman and Basili 2003, Larman 2004). When software began to be developed there were two approaches: (i) incremental and iterative development (IID) and (ii) ad-hoc. The waterfall process (Royce 1970) was developed to improve the ad-hoc development efforts and not necessarily to replace IID

approaches. In fact, the original waterfall approach is nuanced and Royce expected iteration between each stage and even supported early product release and close customer involvement. At some point, a crude version of the waterfall process became the dominant approach, possibly due to its conceptual simplicity, and was used on many projects which would have been better suited to Royce's original waterfall approach or IID approach.

The misuse of the waterfall approach began to be readdressed in the early to mid 1990s, borrowing many IID practices and lean manufacturing concepts, resulting in what would later be known as agile methods. Agile methods received their impetus from the shortcomings of heavily planned processes and crude waterfall processes, which aimed to be successful with all varieties of software products and teams, and are now in the IID family. Agile methods are not ad-hoc, and their empirical nature requires discipline on the part of the team using them.

The contrast between agile methods and plan-driven methods is one of degrees. Both approaches rely on some level of gated management. For example, Scrum (Schwaber 2004) is based around 30-day sprints during which time the 'Scrum Master' ensures, within reason, that only work agreed to be in that sprint is completed before entering the next 30-day sprint. A more plan-driven method, such as RUP, will have more gates coming after different phases of software development. Others, for example Karlström and Runeson (2006), appear to pre-suppose a larger contrast between stage-gate processes and agile methods.

A cursory glance at some of the agile literature, or hearing a short talk on the topic, can give the mistaken belief that an agile development approach will be straightforward to adopt and will result in instant success. A better approach for adopting agile methods is to assess an organisation's risks and what it does to manage these risks. The understanding gained from such an assessment can then be used to inform process improvement (Iversen *et al.* 2004).

This article aims to show how the authors adapted the agility/discipline assessment (ADA) developed by Boehm (2002), Boehm and Tuner (2003a,b, 2004) to help small software companies take a reasoned step towards process and quality improvement through adopting and tailoring agile methods. In the interests of conciseness, the article does not



address in detail the results from the assessments. Rather, the assessments are discussed at a level sufficient to explain the ADA approach.

The article is organised as follows. Section 2 presents the guiding principles for small software companies with regard to process improvement and method tailoring. Section 3 highlights the research approach taken. Section 4 provides an overview of the original ADA, and Section 5 describes how the authors use the ADA in small software companies. Section 6 presents the rationale for the adaptations introduced to the original ADA, with Section 7 containing a brief discussion of the case study results. Section 8 summarises the current adapted ADA and the final section concludes the article by discussing the purpose and usefulness of the adapted ADA.

2. TAILORED AGILE METHOD ADOPTION FOR SMALL SOFTWARE COMPANIES

Recent articles by Börjesson *et al.* (2006) and Fitzgerald *et al.* (2006) discuss the adoption of new technologies and processes, and tailoring of agile methods for software process improvement (SPI), respectively. Börjesson *et al.*'s (2006) context was a 500 employee development group with six people dedicated to improvement initiatives. Three people from this group were dedicated change agents on the specific project discussed in the study. The project also made use of a dedicated reference group who acted as the early adopters of the new software product. The time frame for their study was a 16-month period.

Fitzgerald *et al.*'s (2006) context was a 125 employee group, 45 of whom were involved in the development of two product families. The time frame for each product family and the accompanying studies was 18 and 24 months, respectively. The organisation concerned had approximately 5 years experience in using agile methods.

Both papers present useful studies, essentially regarding SPI. However, the context of each study is relatively large. Because of limitations on scale and resources, small software companies, typically employing fewer than 50 people, find SPI a major challenge. Early stage software companies focus on time to market, innovation and creativity (Sutton 2000) and thus often ignore SPI models such as CMM/CMMI whose primary initial emphasis is

on stability and predictability (Boehm and Tuner 2004). A consequence of this is that, to support their business goals small software companies are now increasingly attracted to agile methods, as they promise shorter development schedules and greater delivery flexibility.

The software industry in the Republic of Ireland (ROI) is a key component of the national economy. According to Enterprise Ireland (EI) (ROI's economic development agency for indigenous companies) at the end of 2004, Irish-owned software businesses comprised over 750 companies employing almost 12,000 people (Enterprise Ireland 2007). The majority of these Irish-owned software firms are small software companies, where it is calculated that only 1.9% employ more than 100 people whilst more than 60% employ ten or fewer (Crone 2002). A Mintel study for Momentum (the Trade Association representing Northern Ireland's Information Communications and Technology industry) on the software sector in Northern Ireland (NI) showed that, at the beginning of 2005 there were 348 companies employing approximately 5500 people. Over 75% of the companies employed fewer than 15 people (McCaffery *et al.* 2004).

Given the prevalence of small software companies in ROI and NI, it is important that any agile adoption and tailoring approach be resource-light. The study by Fitzgerald *et al.* (2006) could be repeated in small software companies, but they are rarely in the position to experiment with agile methods before using them on a business critical project. The study by Börjesson *et al.* (2006), building on the work of Moore (1999, 2002) and Rogers (2003), is more indicative of the resources typically used for SPI in large organisations. However, few small software companies have the resources for dedicated change agents and reference groups.

The guiding principles for adopting and tailoring agile methods in small software companies, resulting from numerous surveys, are as follows:

- *Performed by the whole team* – In small software companies there is not enough resource to have dedicated change agents and SPI teams. The responsibility for how to adopt and tailor agile methods should belong to the whole team.
- *Collectively owned by the team* – Due to the likely absence of dedicated change agents and reference groups, there should be collective agreement as to the merits of the agile approach.



- *Based on an assessment of current successes and failures* – The introduction of any new technology must be for a specific purpose, whether enhancing successes or rectifying the effects of failures. Every software development project will have such a context, from a previous project and the previous experience of team members, which needs to be made explicit.
- *Address current and future risks and process disturbances* – Small software companies are particularly prone to the effects of risks and process disturbances, both of which can irreparably damage business. Any new process must address the risks and lessen the impact of process disturbances.
- *Require a minimum time commitment from individual team members* – Each team member will have a steady backlog of work to do and the overall business model of small software companies is generally short time to customer/market. A 16-month study or 5-year experimentation with agile methods on non-critical projects is not an option. However, some extra work will be required by each team member and the goal is to make this minimally intrusive.

These guiding principles correlate closely to some of the key factors for success in SPI developed by Dybå (2005), who defends the fact that SPI success is positively associated with: *business orientation; involved leadership; employee participation; concern for measurement; exploitation of existing knowledge; and exploration of new knowledge*. The ADA is not based on software measurement and the exploration of new knowledge is more likely to be partially substituted by the *exploitation of existing knowledge*. Given these caveats it is still possible for small software companies to engage in successful SPI by tailoring and adopting agile methods.

3. RESEARCH APPROACH

The approach used when conducting the ADA draws from the positive organisational change method, Appreciative Inquiry (Cooperrider and Srivastva 1987, Cooperrider and Whitney 2005, Egan and Lancaster 2005, Preskill and Catsambas 2006) with the assessors acting as facilitators (Schwarz 2002). The assessors then work with the software company and individual teams to support

the implementation of *planned change* and *guided change* (Kerber and Buono 2005).

The research approach for evolving the ADA, as described in the following sections, is an *interpretive action research, generalised case study* methodology as advocated by Walsham (1995, 2006) and Lee and Baskerville (2003). An interpretive approach is valid in new and evolving areas (Walsham 1995) such as the introduction of agile methods. Action research is appropriate as it emphasises the linking of theory and practice to achieve both practical and research objectives (Susman 1983, Baskerville and Wood-Harper 1996).

The software teams involved were effectively using the authors as consultants but the authors have research obligations to their academic institutions. To help meet the objectives of the software teams and the authors, the evolving ADA adopts the action research process used by Iversen *et al.* (2004):

- The process was *iterative*, involving a repeating set of activities.
- The guidance was *fluid*, with loosely defined activities.
- The researchers' involvement was *facilitative*, helping the team members manage SPI risk.
- The purpose was *organisational development* and *scientific knowledge*.

As the software teams belonged to small companies there was justifiably little concern for research that was not going to contribute to real organisational and product improvement. The action research approach taken helped rescue the ADA and its adaptation from being a pure consultancy exercise.

From a research perspective, each ADA conducted was effectively a case study. Case studies are valuable for generating an understanding of reality (Yin 2003). The case study approach has been used to reach a generalised (Lee and Baskerville 2003) ADA that is useful to small software companies in various product domains.

4. AGILITY/DISCIPLINE ASSESSMENT

As early as 2002 Boehm (2002) had already developed an approach to assess an organisation's suitability for agile methods. The terminology of balancing *agility* and *discipline* is somewhat misleading



as agile methods require discipline if they are to be successful. Rather, Boehm developed the ADA because he believed that a middle-ground could be found between *plan-driven* and *agile* approaches. He thinks this middle-ground is where most software development projects actually reside. Hansson *et al.* (2006) agree that there is often a balance between agile and plan-driven approaches and that some software teams find this balance naturally.

Boehm and Tuner (2004) present the sets of conditions under which agile and plan-driven methods are most likely to succeed:

Application characteristics – primary project goals, project size and application environment.

Management characteristics – customer relations, planning and control and project communications.

Technical characteristics – approaches to requirements definition, development and testing.

Personnel characteristics – customer characteristics, developer characteristics and organisational culture.

For example, agile methods work best when the application environment has a high amount of change, whereas plan-driven methods are better suited to stable environments with low change. Boehm and Tuner's (2004) analysis of the strengths of agile and plan-driven methods yielded five critical factors: *size*, *criticality*, *dynamism*, *personnel* and *culture*. Building upon the five critical factors, Boehm and Tuner (2004) describe a six-stage, risk-based method:

- (i) Rate the project's environmental (project's general environment), agile (risk of using agile methods) and plan-driven (risk of using plan-driven methods) risks. If uncertain about the ratings, then use prototyping, data collection and analysis.
- (ii) If agility risks dominate, use a risk-based plan-driven approach.
- (iii) If plan-driven risks dominate, use a risk-based agile approach.
- (iv) If the risks are a mixture of ii and iii then architect to encapsulate the agile parts. Use a risk-based agile approach on the agile parts and a risk-based plan-driven one elsewhere.
- (v) Establish an overall project strategy by integrating individual risk mitigation plans.
- (vi) Monitor project progress and risk/opportunities; readjust balance and process as appropriate.

5. USING THE ASSESSMENT

The ADA is not designed to be a complete assessment method and therefore is useful to fit into other more complete assessment methods. However, the current popularity of agile methods has resulted in small software companies wanting to investigate their suitability. The authors decided to use the ADA as an engaging yet lightweight way to introduce the key concepts of agile methods and to assess software teams' suitability for an agile approach. The ADA was used in the following manner:

- *Step 1 – Agile Overview.* Agile methods are placed within their proper and historical context, and key terms and concepts are explained. This ensures that all employees have as much mutual knowledge as possible. At this meeting, the criticality factors and risk ratings of Step 3 are introduced. This step takes approximately 1 hour.
- *Step 2 – Project Post-Mortem.* The teams and assessors discuss the successes and problems evident in a recently completed or ongoing software development project. This will uncover some of the main risks to the success of a project. This step takes approximately 1 hour.
- *Step 3 – Critical Factors and Risk Ratings.* This step is performed by each team member, on a form provided by the assessors, and is designed to take approximately 1 hour. The form contains explanatory notes, a sample critical factors diagram and interpretation, a blank critical factors diagram for team members to plot, a risk-rating items table, and space for notes.
- *Step 4 – Present Results.* The results from Step 3 are collated and a visual presentation is produced for the whole organisation, lasting no longer than 2 hours. After this stage, with each team member having committed approximately 5 hours to the assessment, the organisation is free to end its involvement with the assessors.
- *Step 5 – Risk Mitigation Framework.* If the organisation chooses to continue, a risk mitigation strategy is developed. This stage is not time-boxed as each strategy is unique to the organisation concerned. The strategy is then implemented and monitored. The framework approach has the advantages of focusing the efforts of those responsible for process improvement and of



reducing resistance to the required changes because the practices introduced are addressing real risks.

6. ADAPTING THE ASSESSMENT

Boehm and Tuner's (2004) original ADA was used initially by three small software companies working in different domains and with varying numbers of employees. The authors desired to explore the usefulness of the original ADA and to determine if any adaptations were necessary, given the differences between the companies. The following subsections show the adaptations resulting from each case study. Adaptations one and two resulted from case studies with Companies 1 and 2 respectively. Adaptations three and four resulted primarily from the Company 3 case study.

Company 1 provides a range of internet-based software solutions to meet the information management, administration, communication, marketing and revenue generation needs of sports organisations and their associated clubs. They have four software developers and a graphic designer and are successful in their product domain.

Company 2 develops customer relationship management solutions and are a fully credited ISO9001 organisation. This ADA had 19 participants in nine teams with some persons in more than one team. The largest team had four persons and the smallest had one person.

Company 3 develops administration software for the plant hire business and financial loan sector. They also produce a photographic/image processing system designed to capture and store digital images within a controlled environment. This system interfaces directly with hardware systems. This ADA had 29 participants in six teams. The smallest team had three persons and the largest had eight.

6.1. Adaptation One

After the project post-mortem of *Company 1*, it was clear that their biggest risk came from uncooperative customers. As product providers, many small software companies can be made to feel privileged if they secure a contract and associated revenue. Often, the customer will provide initial requirements and remain practically uninvolved until the

handover deadline. This scenario was also noticeable in some large software development organisations. As a result, a sixth factor, *Client Involvement*, was added to the original critical factors graph. *Client Involvement* has the following categories:

- *On AB* – Client is on-site and an agile believer. This is the ideal when a client is fully persuaded of the agile approach and is available on-site to work with the team.
- *Off AB* – Client is off-site but an agile believer. Although off-site, the client fully understands the nature of agile development and is open to frequent communication.
- *On AS* – Client is on-site but is an agile sceptic. They may be on-site but they are not convinced about the agile development approach. This is more problematic than Off AB because the relationship is one of resistance rather than facilitation.
- *Off AS* – Same as On AS except the problem is compounded by the client being off-site.
- *Off Uninvolved* – Not only is the client off-site but they want no involvement in providing the initial requirements and getting the right product delivered.

6.2. Adaptation Two

The other component in this stage of the assessment is the *risk ratings*. The risk ratings are characterised as *environmental*, *agile* and *plan-driven*. Table 1 shows an example of the risk ratings. The potential highest total rating for each risk category is 15, 20 and 20 respectively. In keeping with the visual nature of the assessment stages thus far, and at the request of *Company 2*, the *risk ratings* were also converted into risk-rating diagrams.

6.3. Adaptation Three

The third adaptation involved the inclusion of a more standard analysis of quantitative data. The ratings for *environmental*, *agile* and *plan-driven* risk categories are entered into a spreadsheet and various graphs can then be generated. As with the critical factors graphs, analysis of the quantitative data can be given on a per-team basis or as a collation of all teams. A discussion with the company management will help clarify what types of analysis they want.



Table 1. Risk ratings

Risk items	Risk rating
<i>Environmental risks</i>	
Technology uncertainties	1
Many stakeholders	1
Complex system of systems	1
	Total = 3
<i>Risk of using agile methods</i>	
Scalability and criticality (agile methods do not scale easily to large, safety critical products; so if your project is large and safety critical then this would have a high risk rating)	1
Use of simple design (agile methods advocate a simple design which evolves as the project proceeds rather than detailed up-front design; so if your project relies upon detailed up-front design this would have a high risk rating)	1
Personnel turnover (if your project has experienced personnel turnover then this would have a high risk rating)	4
Not enough people skilled in agile methods (if no team member has experience with agile methods then this would get a high risk rating)	2
	Total = 8
<i>Risk of using plan-driven methods</i>	
Rapid change (plan-driven methods are not designed for rapid change; so if your project experiences rapid change in direction then the risk of using a plan-driven method would be high)	4
Short development cycles (releasing working software every few weeks would prove to be high risk for plan-driven methods)	4
Emergent requirements (new requirements late in a project life cycle have the potential to break a plan-driven method and are therefore considered high risk)	4
Not enough people skilled in plan-driven methods (similar to the agile scenario)	3
	Total = 15
1 – minimal risk, 2 – moderate risk, 3 – serious but manageable risk, 4 – very serious but manageable risk, 5 – show stopper risk	

6.4. Adaptation Four

The final adaptation resulted from comments on the assessment forms and further discussion with team members. Firstly, the *culture* factor was deemed to be least useful as most people were in the middle, enjoying change but valuing stability. This was replaced with the *team distribution* risk factor. Each company was eager to explore how the agile approach might work with global software development. *Team distribution* is defined as *co-located, not seated together, different floors/buildings, more than one organisation, and different time zones and cultures*.

Secondly, the *personnel* factor appeared to confuse people. It was simplified by using a count instead of the percentage. The scale supports most team sizes that the authors have encountered in small software companies but it can be adjusted to cope with the team size of each company, through information gathered from the first stage of the assessment.

7. DISCUSSION OF ADA RESULTS

As the original ADA was used and adapted with each small software company, it became progressively clearer what it could indicate about

the teams and individuals. The following subsections describe the type of information the authors were able to provide to the companies after they had undergone the ADA. As the ADA is based on subjective ratings the results are indicative rather than empirical.

7.1. Company 1

The biggest risk for Company 1 is the situation where there are off-site uninvolved customers who have the potential to break the agile development approach they were wishing to adopt. For Company 1 the objective was to get their customers from *Off Uninvolved* to *Off AB*. Such customers work well with internet-based product development due to the relatively straightforward nature of accessing working versions of the product. Before reaching the *Off AB* category for *Client Involvement* the risk needed to be more specifically defined and a workable plan implemented.

Company 1 has experienced a customer who desired new functionality at the product handover stage. In this circumstance, the contract arrangement meant that they did not obtain full payment



until the customer was satisfied, which led to further development having to be completed within the budget of the original time-scale. For a small software team providing competitively priced work based on the originally approved project, such a scenario can leave them in a vulnerable position in the working relationship. The subsequent work to satisfy the customer can lead to unplanned and badly paced software development. There is also the danger that the customer perceives an inability to meet their desires first time.

Table 1 actually presents the combined risk ratings for Company 1 from Step 3 of the ADA. The total risk ratings clearly show that the company is relatively unaffected by environmental risks. The technology they use, whilst changing regularly, is not uncertain. Their systems are not overly complex and there are few stakeholders. Using a plan-driven approach has greater risk when compared to using an agile approach resulting in the decision to work towards adopting an agile approach.

The primary risk with using the agile approach is personnel turnover (rated as a very serious but

manageable risk). Many small software organisations can lose employees to bigger organisations offering a more stable future and better benefits packages. It is crucial for them to manage this risk effectively. It was clear from the first ADA conducted with Company 1 that the simple visual nature of presenting the critical factors resulted in more team members willing to discuss problems and solutions. For some companies this is enough to begin meaningful software process and quality improvement.

The initial ADA performed with Company 1, and discussed more completely in Taylor *et al.* (2006), resulted in the following risks being deemed most urgent:

- *Off-site uninvolved customers.* To begin addressing this risk, weekly incremental delivery and corresponding user acceptance testing for the last 3 weeks of any project became a contractual obligation for customers.
- *Personnel turnover.* This risk is partially

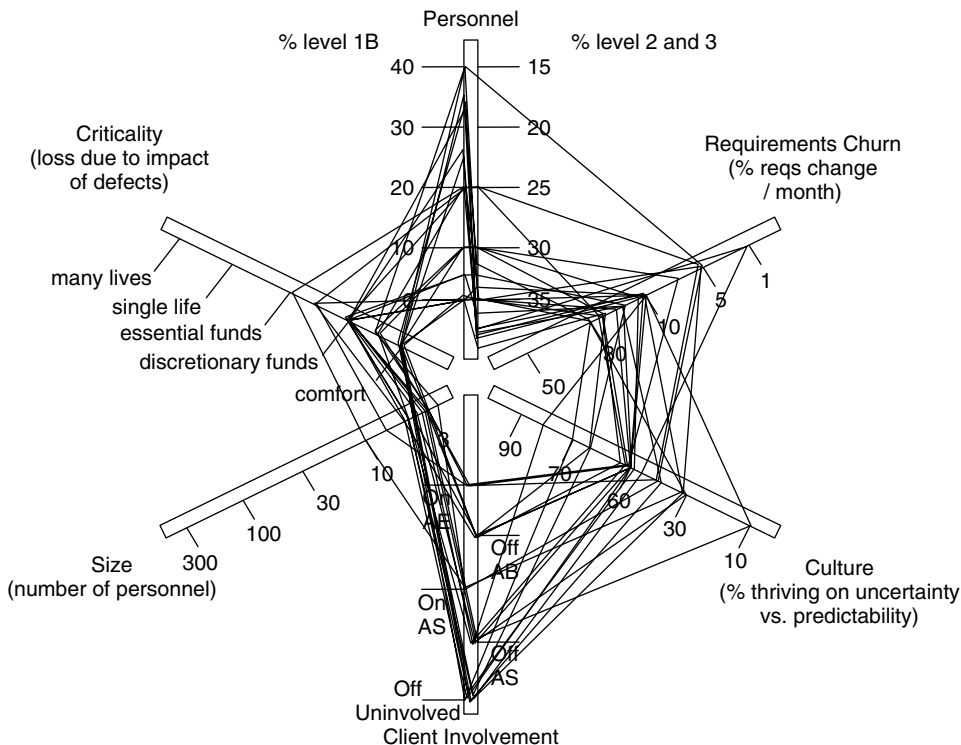


Figure 1. Combined six critical risk factors graph for Company 2



addressed by the use of documentation governed by Ruping's (2003) agile documentation principles in conjunction with a community editable website (Wiki) and mentored role changing.

7.2. Company 2

Figure 1 shows the graph for the combined critical risk factors of Company 2. The purpose of the combined graph is to show any obvious: (i) commonalities, indicating suitability for agility or discipline, or (ii) variances, indicating different opinions amongst team members. Figure 1 indicates that Company 2 has no clear suitability for agility or discipline except on the *criticality* and *size* dimensions, which are nearer the centre of the graph. The other dimensions show significant variance amongst the teams regarding the critical risk factors.

This part of the ADA presentation primarily generates discussion amongst the team leaders and managers, particularly focusing on the range of *requirements churn* and *client involvement*.

Figure 2 shows the risk factors for an individual team. Nine such graphs were produced for Company 2 with the objective of getting each team member thinking about process improvement and quality. The commonalities and variances on a per-team basis generate significantly more discussion, often continued beyond the 2 hour presentation.

Considering that Figure 2 shows the critical risk factors for one team, consisting of four members, there is significant variance on all dimensions:

Criticality – One team member thought that the failure of the system they were producing would only result in loss of comfort whilst the other three thought it would result in the loss of discretionary funds. This variance is not important to the function of the team but it may suggest that there needs to be a better understanding of the customer domain.

Size – One team member thought that there were more than four people in the team. This variance may indicate that the team is inadequately defined but seemed to have no negative effect in this instance.

Client Involvement – The variances on this dimension are crucial. One team member thought that the client

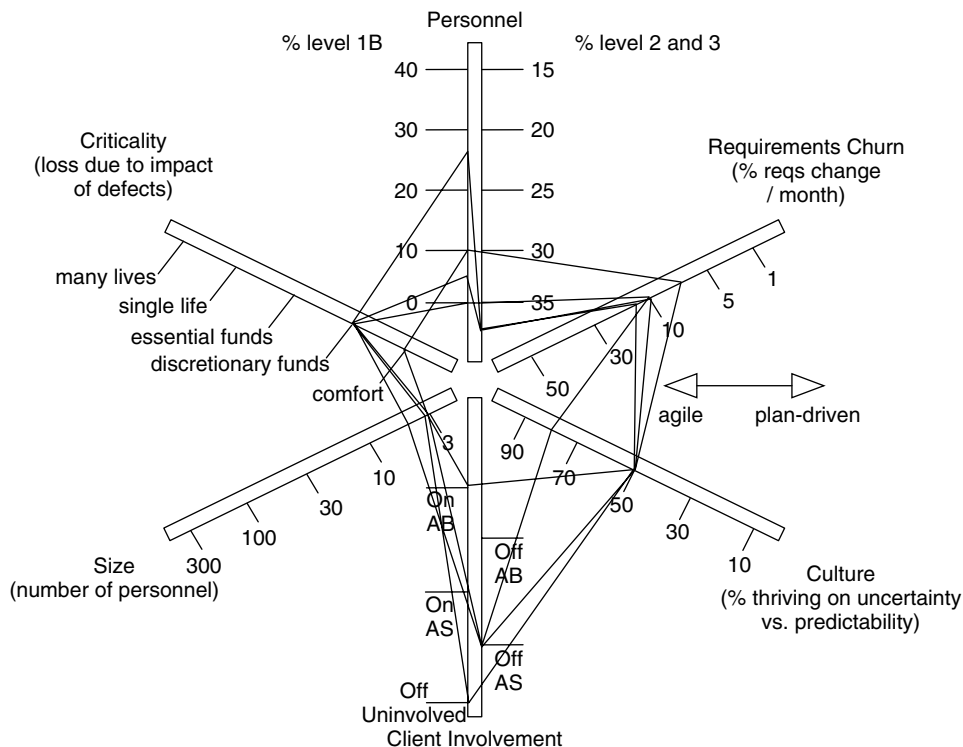


Figure 2. Team six critical risk factors graph for Company 2



Table 2. Personnel variances

Team member	Initial values			Adjusted values (after discussion)		
	% Level 2/3	% Level 1B	% Level 1A	% Level 2/3	% Level 1B	% Level 1A
1	50	5	45	50	0	50
2	50	25	25	50	25	25
3	35	0	65	50	0	50
4	30	10	60	25	25	50

Level	Characteristics
3	Able to revise a method (break its rules) to fit unprecedented situations.
2	Able to tailor a method to fit precedented new situations.
1A	With training, able to perform discretionary method steps (e.g. task estimation, composing patterns, architecture re-engineering).
1B	With training, able to perform procedural method steps (e.g. coding a class method, using a CM tool, performing a build/installation/test).
-1	May have technical skills, but unable or unwilling to collaborate.

was *On AB* whilst the others thought that the client was off-site. In this case the *On AB* client actually belonged to Company 2 and was external client facing, giving the requirements to the developer. The other team members, working in a different subsystem, interacted directly with the external client. Both scenarios can be problematic, but when an external client is an agile believer they will want and require direct interaction with team members.

Culture – One team member enjoyed the times when uncertainty was the dominant feature and generally enjoyed development projects characterised by this. The other team members preferred a healthy balance of predictability and uncertainty. In other case studies, this variance can point to team members who work longer hours and get more work assigned as a result. They can get the reputation as the problem solver and often become indispensable to the team. This in itself reveals a risk of over reliance on this team member. However, since variance on this factor was so uncommon it was replaced with *Team Distribution* after subsequent case studies.

Requirements Churn – In this instance, the variances reflected the experience of the developers. The least experienced developer was working on an area of the system that was less likely to change.

Personnel – Table 2 shows the ratings for this *Personnel* dimension. In this case, the team members had not grasped the personnel concept clearly. Further discussion resulted in more accurate assessment

of each other (see Table 2). The differences in personnel assessment come from a number of things such as modesty, kindness, friendship, vendetta and inexperience. However, although the assessment is subjective, vastly differing levels can suggest a team that is not performing together optimally. If your assessment of a colleague is lower than it should be you are less likely to involve him/her in complex design decisions. Likewise, if it is higher than it should be, you may be giving him/her work that he/she is incapable of completing in the time-scale and to the quality necessary.

7.3. Company 3

From Figure 3 the obvious company wide commonalities are *criticality* and *size*. The other critical factors show wide variance and therefore require analysis on an individual team basis. Figure 4 shows the critical factors for one team, consisting of eight members, in Company 3:

Criticality – Only two team members had mistaken opinions on the criticality. It did not appear to have any effect on team performance.

Size – The team was clearly defined.

Client Involvement – One team member believed their client was on-site and convinced of the benefits of agility. Upon further discussion this team member was confusing their client with their team leader. They are now being given experience with external client interaction. With regard to the off-site clients, the team members who thought their

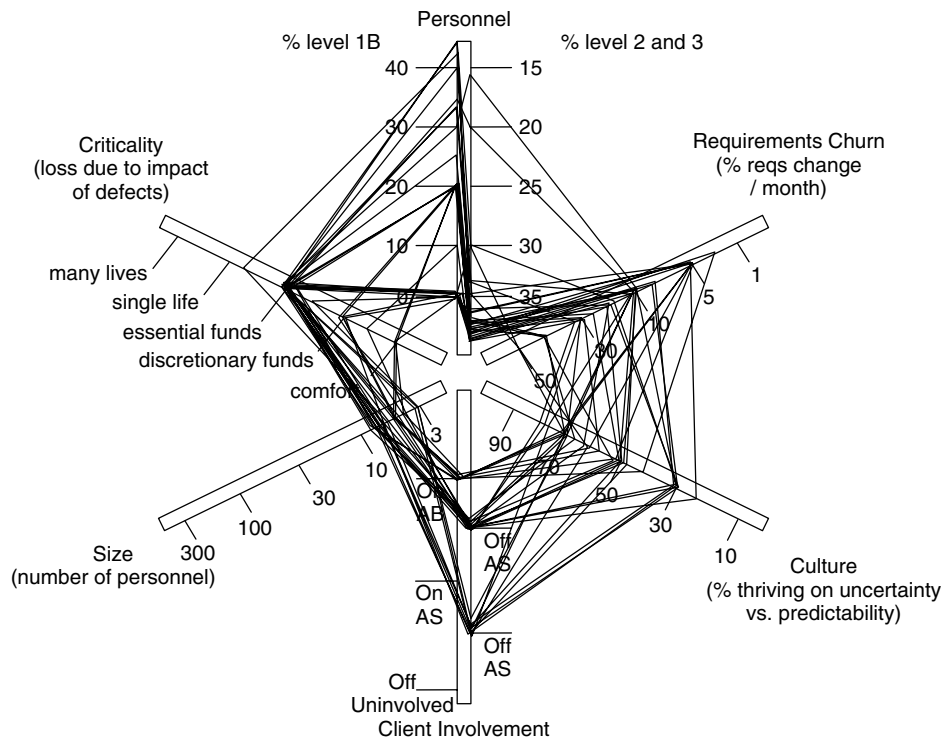


Figure 3. Combined six critical risk factors graph for Company 3

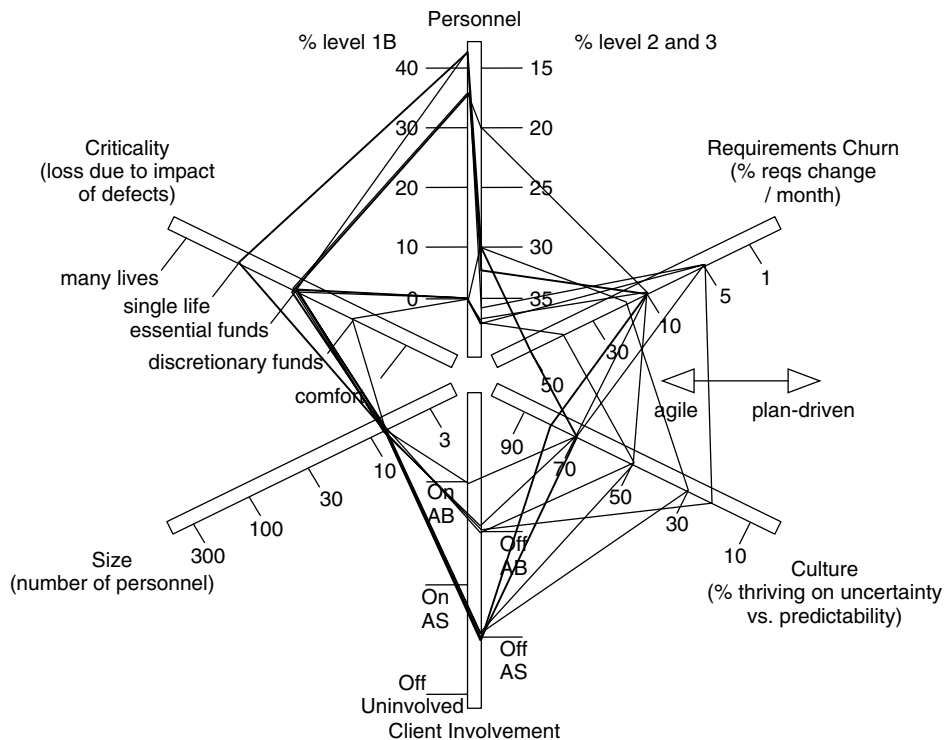


Figure 4. Team six critical risk factors graph for Company 3



clients were agile believers had assumed this fact, such that, if they were following an agile method their clients would be convinced of the approach and work with the team. The team members who thought their clients were agile sceptics assumed that, because they do not follow an agile approach to release management now, the client would be unlikely to in the future. Both assumptions can strain a team and stifle its ability to do meaningful process and quality improvement.

Culture – A wide variance on this dimension appeared to reflect people’s personal characters. Once the team leader was aware of these preferences, motivating them, assigning new work and assessing them became more straightforward.

Requirements Churn – In this case, the 50% requirements change per month was actually service requests and fixes. After discussion the real figure was nearer 10%.

Personnel – Significantly, four team members thought that there were no level 1B members and a higher percentage of level 2/3 staff. The question in this case for the team is which assessment is more accurate? If over half of the team are level 2/3, perhaps some of them can be redistributed to other teams who have less experienced members.

Alternatively or additionally, the level 1A members could be trained to help them become level 2/3.

Figure 5 shows the combined Risk Item Ratings for Company 3. As with the combined Critical Factors graph, the objective of the combined risk item ratings graph is to visually show any obvious commonalities and variances. In Figure 5 there is a general trend suggesting that the plan-driven risks are more significant than the agile risks. There were only three instances of team members who disagreed with this general trend and rated the risks of using agile methods higher than the risks of using plan-driven methods.

The combined risk item ratings can also be shown on a per-team basis as in Figure 6. In this example, there is close commonality regarding the environmental risks and the risk of using agile methods. However, there is significant variance in the risk of using plan-driven methods. In this instance, the team appears to be equally split between those who have rated agile risks higher than plan-driven risk and vice versa.

Serious discussion and reflection can begin once each of the risk item groups are examined on a per-team basis. For example, Figure 7 shows the graph for the agile risk item groups on a

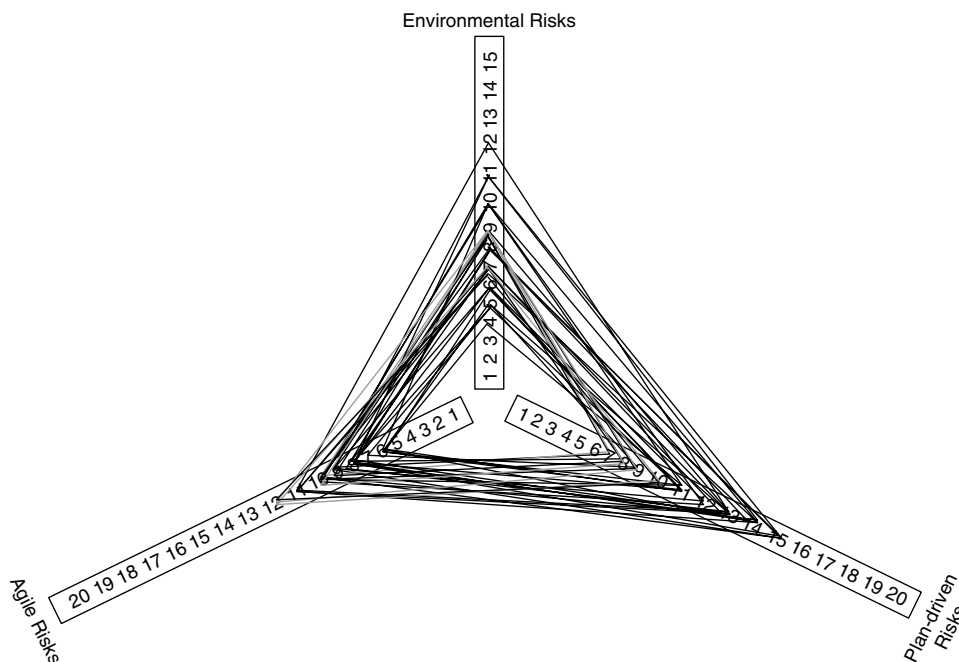


Figure 5. Combined risk item ratings for Company 3

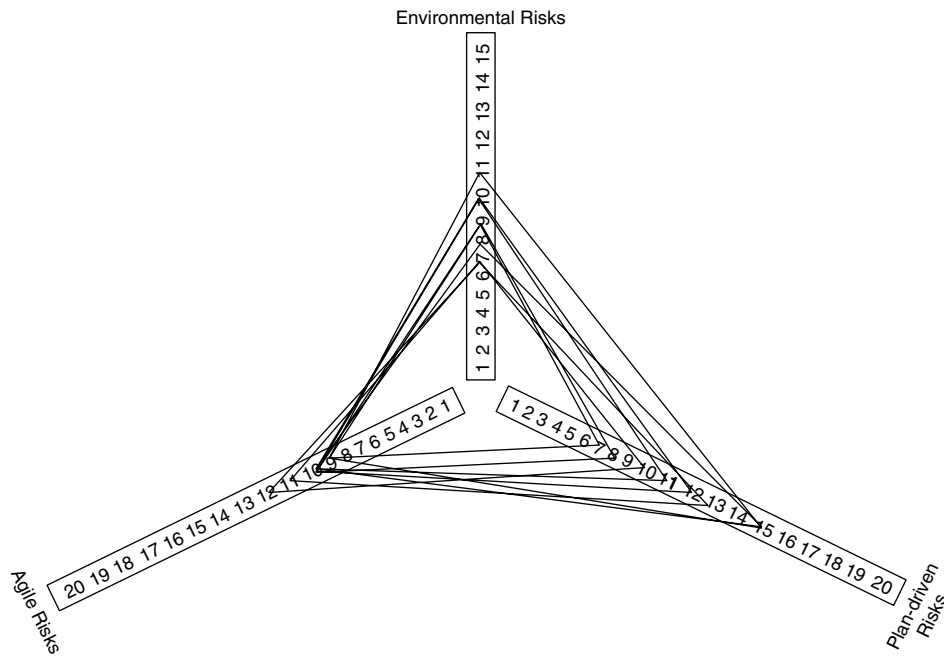


Figure 6. Combined risk item ratings for one team

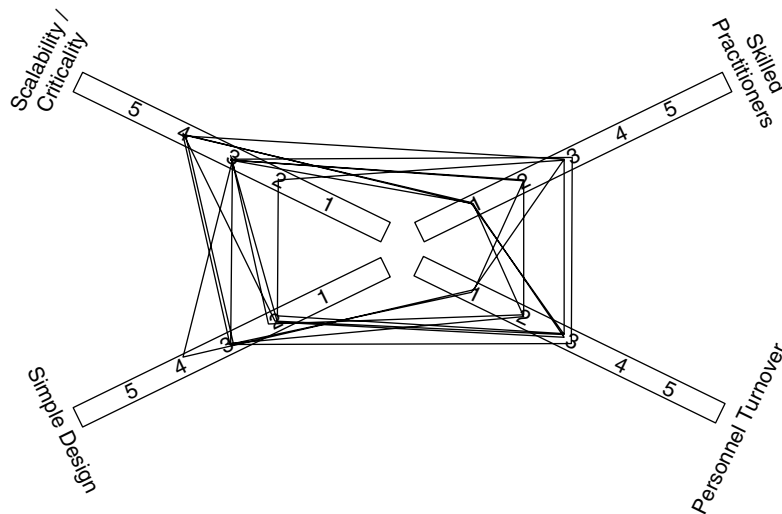


Figure 7. Agile risk ratings for one team

per-team basis. Again, the objective is to locate commonalities and variances. The diagrams are powerful when commonality is displayed but when there is variance, as in the case of Figure 7, the diagrams provide little explanatory power.

When the individual risk ratings graphs do not exhibit commonality, a more traditional quantitative presentation can be used. Figure 8 shows the

frequency of each type of risk. *Serious but manageable* risks are the most frequent in Company 3. However, there are a significant number of *very serious but manageable* risks and even some risks assessed as *show stoppers*. Figure 9 shows the frequency of ratings for each environmental risk. *Technical uncertainty* accounts for most of the moderate risk whereas the *presence of many stakeholders* and *system complexity*

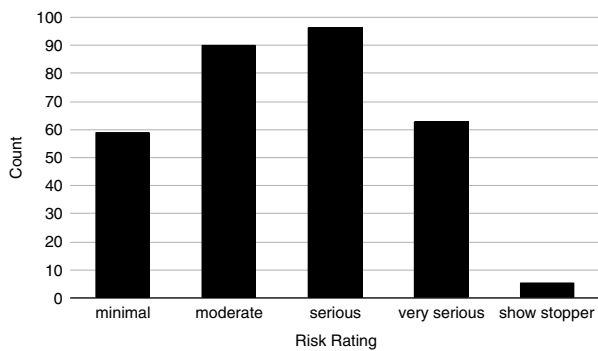


Figure 8. Risk rating count for Company 3

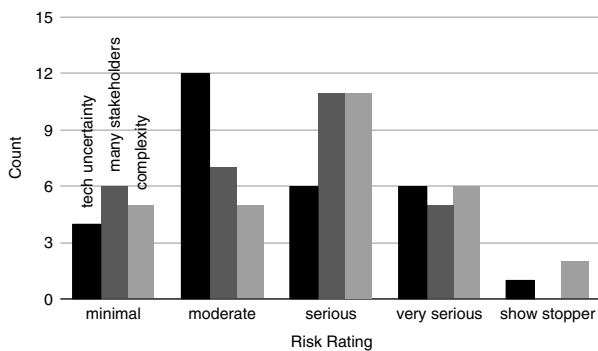


Figure 9. Environmental risk rating count for Company 3

account for most of the serious risk. Each team in Company 3 worked with such results to obtain a risk mitigation framework that helped towards process improvement.

8. THE CURRENT ADAPTED ADA

The current adapted ADA has been used in an assessment (McCaffery *et al.* 2007) and it is believed that the *Critical Factors* graph is now clearer and caters to the primary risks that most small software companies need to address when making process changes.

The current adapted ADA includes two new critical factors, *Client Involvement* (see Figures 1–4) and *Team Distribution*, and an altered *Personnel* critical factor. These changes have been driven by using the ADA to help small software companies assess how much agility or discipline they require for any given project.

Presenting the assessment results is achieved by three groups of diagrams. First, the *Combined 6 Critical risk Factors* graph (see Figure 1) is a high level summary with the aim of showing general commonalities or variances. The *Team 6 Critical risk Factors* graph (see Figure 2) can then be used to show the same dynamics on a per-team basis. Second, the risk items rated by team members in the provided table are used to get a *Combined Risk Item Ratings* graph (see Figure 5). This graph is used to show general commonalities and variances. *Team Risk Item Ratings* can also be produced for team-specific discussion purposes (see Figure 6). It can then be helpful to provide further analysis of environmental, agile (see Figure 7) and plan-driven risk ratings on a per-team basis. Third, if there is little commonality shown in the previous diagrams then a more conventional presentation of quantitative data can be produced (see Figures 8 and 9).

9. CONCLUSIONS

For many small software companies, process and quality improvement now involves a close examination of customer, product domain, software engineers and software development approaches. Software products that have a clear goal but unclear solution or, an unclear goal and unclear solution are increasingly frequent and software development approaches must take account of this.

The ADA is one example of an assessment that seeks to help small software companies decide which approaches are most suitable for specific projects. However, the ADA is not prescriptive and is extendable and adaptable for different software development contexts. This article has sought to highlight how the ADA can be used with small software companies and what adaptations make it more useful. A team or software company can also further adapt the ADA to uncover opinions on its domain-specific risks.

The primary purpose of the adapted ADA is to get all members of a small software company involved in discussions regarding how to tailor and adopt agile practices. The combined graphs show general trends for a company and will elicit discussion from managers and those with process improvement responsibility. The team-based graphs provide a visually stimulating way to draw comments from



all team members. As each team member has given his/her own perception of critical risk factors and risk ratings, the commonalities and variances can be used to correct erroneous views and highlight strengths and weaknesses within teams.

The ADA approach has a minimal resource overhead and can fit into other assessment programmes if required (McCaffery *et al.* 2007). However, due to the minimal resource overhead, it is recommended that the adapted ADA be used at the following stages of a project:

- *As part of project launch.* This stage incorporates pre- and post-project assessments as the launching of a new project will mostly come after previous projects, whether within the company being assessed or from the collective experience of its staff. The successes and failures experienced on previous projects will become important for any subsequent projects.
- *During a project life cycle.* In some software development projects, it is clear that change is required if the product is to be successful and the adapted ADA can be a useful catalyst for such change. If a project is particularly complex and of a long duration it can be productive to have team members focus on the bigger picture of the product and realign their efforts to produce a better product.

After the main assessment portion of the adapted ADA has been completed, the *Risk Mitigation Framework* can be used to address real issues, with practices agreed upon by those who will be using them.

Further work has commenced on the ADA described in this article. The first new adaptation is based on the work of Tsumaki and Tamai (2006) who have developed a framework for matching requirements elicitation techniques to project characteristics. Specific approaches for customer involvement and requirements elicitation are still deemed to be a crucial and ongoing risk by small software companies. The requirements elicitation technique will also have a bearing on how agile or how plan-driven a project can be. The second new adaptation is the development of a software tool that can be used to gather opinion and chart the results. It is hoped that the tool can be rolled out to numerous small software companies in ROI and NI, so that collective trends can be analysed and used

by EI and Momentum to help better support small software companies.

REFERENCES

- Abrahamsson P, Warsta J, Siponen MT, Ronkainen J. 2003. New directions on agile methods: a comparative analysis. *Proceedings 25th International Conference Software Engineering*. IEEE Computer Society Press: Los Alamitos; 244–254.
- Baskerville RL, Wood-Harper AT. 1996. A critical perspective on action research as a method for information systems research. *Journal of Information Technology* 11(3): 235–246.
- Boehm B. 2002. Get ready for agile methods, with care. *IEEE Computer* 35(1): 64–69, DOI 10.1109/2.976920.
- Boehm B, Tuner R. 2003a. Rebalancing your organization's discipline and agility. In *XP/Agile Universe 2003*, Maurer F, Wells D (eds). Springer-Verlag: Berlin; 1–8.
- Boehm B, Tuner R. 2003b. Using risk to balance agile and plan-driven methods. *IEEE Computer* 36(6): 57–66, DOI 10.1109/MC.2003.1204376.
- Boehm B, Tuner R. 2004. *Balancing Agility and Discipline – A Guide for the Perplexed*. Addison-Wesley: Boston, MA.
- Börjesson A, Martinsson F, Timmerås M. 2006. Agile improvement practices in software organizations. *European Journal of Information Systems* 15(2): 169–182, DOI 10.1057/palgrave.ejis.3000603.
- Ceschi M, Sillitti A, Succi G, De Panfilis S. 2005. Project management in plan-based and agile companies. *IEEE Software* 22(3): 21–27, DOI 10.1109/MS.2005.75.
- Cooperrider DL, Srivastva S. 1987. *Appreciative Inquiry in organizational life*. In *Research in Organizational Change and Development*, Vol. 1., Woodman R, Pasmore W (eds). JAI Press: Greenwich, CT; 129–169.
- Cooperrider DL, Whitney D. 2005. *Appreciative Inquiry: A Positive Revolution in Change*. Berrett-Koehler Publishers, Inc: San Francisco, CA.
- Crone M. 2002. The Irish indigenous software industry: explaining the development of a knowledge-intensive industry cluster in a less favoured region. *Proceedings of 42nd Congress of the European Regional Science Association*. <http://www.erini.ac.uk/publications/PDF/Nierc-Conf2.pdf> (last visited April 2007), Dortmund.
- Dybå T. 2005. An empirical investigation of the key factors for success in software process improvement. *IEEE*



Transactions on Software Engineering 31(5): 410–424, DOI 10.1109/TSE.2005.53.

Egan TM, Lancaster CM. 2005. Comparing appreciative inquiry to action research: OD practitioner perspectives. *Organization Development Journal* 23(2): 29–49.

Enterprise Ireland. 2007. Software industry statistics 1991–2005. <http://www.nsd.ie/htm/ssii/stat.htm>, (last visited April 2007).

Fitzgerald B, Hartnett G, Conboy K. 2006. Customising agile methods to software practices at Intel Shannon. *European Journal of Information Systems* 15(2): 200–213, DOI 10.1057/palgrave.ejis.3000605.

Hansson C, Dittrich Y, Gustafsson B, Zarnak S. 2006. How agile are industrial software development practices? *Journal of Systems and Software* 79(9): 1295–1311, DOI 10.1016/j.jss.2005.12.020.

Highsmith J. 2002. *Agile Software Development Ecosystems*. Addison-Wesley: Boston, MA.

IEEE. 1991. *IEEE Software Engineering Standards Collection*. IEEE Press: Los Alamitos, CA.

Iversen JH, Mathiassen L, Nielsen PA. 2004. Managing risk in software process improvement: an action research approach. *MIS Quarterly* 28(3): 395–433.

Karlström D, Runeson P. 2006. Integrating agile software development into stage-gate managed product development. *Empirical Software Engineering* 11(2): 203–225, DOI 10.1007/s10664-006-6402-8.

Keil M, Mann J, Rai A. 2000. Why software projects escalate: an empirical analysis and test of four theoretical models. *MIS Quarterly* 24(4): 631–664.

Keil M, Rai A, Mann JEC, Zhang GP. 2003. Why software projects escalate: the importance of project management constructs. *IEEE Transactions on Engineering Management* 50(3): 251–261, DOI 10.1109/TEM.2003.817312.

Kerber KW, Buono AF. 2005. Rethinking organizational change: reframing the challenge of change management. *Organization Development Journal* 23(3): 23–38.

Larman C. 2004. *Agile & Iterative Development: A Manager's Guide*. Addison-Wesley: Boston, MA.

Larman C, Basili VR. 2003. Iterative and incremental development: a brief history. *IEEE Computer* 36(6): 47–56, DOI 10.1109/MC.2003.1204375.

Lee AS, Baskerville RL. 2003. Generalizing generalizability in information systems research. *Information Systems Research* 14(3): 221–243.

Lee G, Xia W. 2005. The ability of information systems development project teams to respond to business and technology changes: a study of flexibility measures. *European Journal of Information Systems* 14(1): 75–92, DOI 10.1057/palgrave.ejis.3000523.

McCaffery F, Taylor PS, Coleman G. 2007. Adept: a unified assessment method for small software companies. *IEEE Software* 24(1): 24–31, DOI 10.1109/MS.2007.3.

McCaffery F, Wilkie FG, McFall D, Lester N. 2004. Northern Ireland software industry survey. *Proceedings of Fourth International SPICE Conference on Process Assessment and Improvement*, SPICE User Group, Lisbon, 159–161.

Moløkken-Østfold K, Jørgensen M. 2005. A comparison of software project overruns – flexible versus sequential development models. *IEEE Transactions on Software Engineering* 31(9): 754–766, DOI 10.1109/TSE.2005.96.

Moore G. 1999. *Inside the Tornado: Marketing Strategies from Silicon Valley's Cutting Edge*. HarperCollins: New York, NY.

Moore G. 2002. *Crossing the Chasm: Marketing and Selling Technology Products to Mainstream Customers*. HarperBusiness: New York.

Overby E, Bharadwaj A, Sambamurthy V. 2006. Enterprise agility and the enabling role of information technology. *European Journal of Information Systems* 15(2): 120–131, DOI 10.1057/palgrave.ejis.3000600.

Preskill HS, Catsambas TT. 2006. *Reframing Evaluation through Appreciative Inquiry*. Sage Publications, Inc: Thousand Oaks, CA.

Rogers EM. 2003. *Diffusion of Innovations*, 5th edn. Free Press: New York, NY.

Royce WW. 1970. Managing the development of large software systems. *Proceedings of WESCON*, IEEE Computer Society: 1–9, Available for download at <http://www.cs.umd.edu/class/spring2003/cmssc838p/Process/waterfall.pdf> (last visited April 2007), Los Alamitos.

Ruping A. 2003. *Agile Documentation – A Pattern Guide to Producing Lightweight Documentation for Software Projects*. John Wiley and Sons: Hoboken, NJ.

Schwaber K. 2004. *Agile Project Management with Scrum*. Microsoft Press: Redmond, WA.

Schwarz R. 2002. *The Skilled Facilitator*, 2nd edn. San Francisco: Jossey-Bass.

Susman G. 1983. Action research: a sociotechnical systems perspective. In *Beyond Method: Strategies for Social Research*, Morgan G (ed.). CA Sage: Newbury Park, CA; 95–113.



- Sutton SM Jr. 2000. The role of process in a software start-up. *IEEE Software* **17**(4): 33–39, DOI 10.1109/52.854066.
- Syed-Abdullah S, Holcombe M, Gheorge M. 2006. The impact of an agile methodology on the well being of development teams. *Empirical Software Engineering* **11**(1): 143–167, DOI 10.1007/s10664-006-5968-5.
- Taylor PS, Greer D, Sage P, Coleman G, Mcdaid K, Lawthers L, Corr R. 2006. Applying an Agility/Discipline assessment for a small software organisation. *Product-Focused Software Process Improvement, LNCS 4034*. Springer-Verlag: Berlin; 290–204.
- Trudel S, Lavoie J-M, Paré M-C, Suryan W. 2006. PEM: The small company-dedicated software process quality evaluation method combining CMMISM and ISO/IEC 14598. *Software Quality Journal* **14**(1): 7–23, DOI 10.1007/s11219-006-5997-8.
- Tsumaki T, Tamai T. 2006. Framework for matching requirements elicitation techniques to project characteristics. *Software Process: Improvement and Practice* **11**(5): 505–519, DOI 10.1002/spip.293.
- van Oosterhout M, Waarts E, van Hillegersberg J. 2006. Change factors requiring agility and implications for IT. *European Journal of Information Systems* **15**(2): 132–145, DOI 10.1057/palgrave.ejis.3000601.
- Walsham G. 1995. Interpretive case studies in IS research: nature and method. *European Journal of Information Systems* **4**(2): 74–81.
- Walsham G. 2006. Doing interpretive research. *European Journal of Information Systems* **15**(3): 320–330, DOI 10.1057/palgrave.ejis.3000589.
- White AS. 2006. External disturbance control for software project management. *International Journal of Project Management* **24**(2): 127–135, DOI 10.1016/j.ijproman.2005.07.002.
- Yin RK. 2003. *Case Study Research: Design and Methods*. SAGE Publications: Thousand Oaks, CA.