

# Applying an Agility/Discipline Assessment for a Small Software Organisation

Philip S. Taylor<sup>1</sup>, Des Greer<sup>1</sup>, Paul Sage<sup>1</sup>, Gerry Coleman<sup>2</sup>, Kevin McDaid<sup>2</sup>,  
Ian Lawthers<sup>2</sup>, and Ronan Corr<sup>3</sup>

<sup>1</sup> Queen's University Belfast, School of Computer Science,  
Belfast BT7 1NN, Northern Ireland, UK  
{p.taylor, des.greer, p.sage}@qub.ac.uk

<sup>2</sup> Dundalk Institute of Technology, Department of Computing and Maths,  
Dublin Road, Dundalk, Co. Louth, Ireland  
{gerry.coleman, kevin.mcdaid, ian.lawthers}@dkit.ie

<sup>3</sup> Servasport, 102 Lisburn Road, Belfast BT9 6AG,  
Northern Ireland, UK  
ronan.corr@servasport.com

**Abstract.** The adoption of agile software development methodologies may appear to be a rather straightforward process yielding instantly improved software in less time and increasingly satisfied customers. This paper will show that such a notion is a misunderstanding and can be harmful to small software development organisations. A more reasonable approach involves a careful risk assessment and framework for introducing agile practices to address specific risks. A case study with a small software development organisation is provided to show the assessment in practice and the resulting risk mitigation strategies for process improvement.

## 1 Introduction

Readers of software process research papers and books may find it difficult to believe that there are software development organisations with no discernable process to help guide development. The authors of this paper have spent over six months observing development meetings and interviewing software engineers and managers from a range of companies varying in size and product domain. Some of the large organisations, particularly in the telecoms domain, have been using specific established processes for years. Some of the smaller organisations have developed their own process and are very successful.

However, there is also a set of smaller organisations, typically with fewer than ten employees, that are not using any defined process. This paper is concerned with such organisations. They can still develop successful products and provide excellent support for their customers but they are at great risk from issues such as an increasing number of new contracts, employee turnover, misunderstood requirements and so forth. Such organisations, to be successful, often work at an unsustainable pace. This situation is obviously detrimental for a small business and something which an agile approach attempts to address. It is in such organisations that an agile approach to development is often seen to be the quick and easy solution for preparing the business to grow by building better products and satisfying more customers. 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 result in instant success.

A better approach for adopting agile methods is to take the time to assess what an organisation’s risks are and what it does to manage these risks. This understanding can then be used to inform process improvement.

This paper aims to show how the authors adapted the assessment developed by Boehm and Turner [1], [2], [3], [4] to help a small software development organisation take a reasoned step towards process improvement and an Agile approach to satisfying their customers.

The paper is organised as follows. The second section provides a short historical context for agile methods by discussing their evolution. The third section will discuss approaches to adopting agile methods and the fourth section will introduce Boehm and Turner’s Agility/Discipline assessment. Section five presents the case study with the company Servasport and how the authors utilised Boehm and Turner’s Agility/Discipline assessment. Section six generalizes the risk mitigation process for introducing agile methods. The seventh section concludes this paper, summarising the key findings.

## 2 Agile Methods in Context

This section will briefly present a historical evolution of agile methods and thereby counter some of the misunderstandings that software organisations may have regarding their validity. For overviews of individual agile methods the reader can consult Abrahamsson et al [5] and Highsmith [6].

Larman and Basili [7], [8] have carefully provided the context for current agile methods. They argue convincingly that many of the practices which appear to be novel in agile methods, most notably incremental and iterative development (IID), have actually been practiced since software began to be developed in the 1950’s.

Figure 1 shows the context of agile methods. When software began to be developed there were two approaches, IID and ad hoc. The waterfall process [9] was developed to improve those ad hoc development efforts and not necessarily to replace IID. At some point 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 IID. This issue began to be addressed in the early to mid 1990’s resulting in what would later be known as Agile Methods.

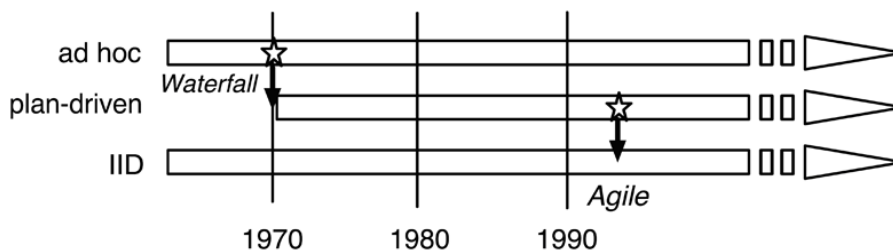


Fig. 1. The historical context for agile methods

Agile methods have been derived from the failure of the plan-driven waterfall processes to be successful with all varieties of software product and team 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. All three streams of software development are likely to continue into the future.

### 3 Adopting Agile Methods

This research has arisen from the Software Process Agility for Competitive Edge (SPACE) project [10]. This project has the primary aim of promoting the merits of agile methods for smaller software development organisations and to enable the adoption of these methods to increase efficiency and competitiveness.

As stated previously, adopting an agile development approach is not a simple solution to an organisation's problems and may, in fact, lead to further problems. Turk et al [11], [12] have discussed some of the problems they perceive with agile methods. Their work is based primarily on examining the underlying assumptions of agile methods and determining for which development scenarios the assumptions do not hold. They arrive at two groups of limitations:

#### *Personnel limitations*

- Limited support for distributed development environments
- Limited support for subcontracting
- Limited support for large teams

#### *Product limitations*

- Limited support for building reusable artifacts
- Limited support for developing safety-critical software
- Limited support for developing large, complex software

There is a certain amount of truth in each of these perceived limitations and, obviously, if a development scenario involved any of the above situations then a careful risk analysis would have to be completed. Agile method proponents are quick to state that much work has yet to be done in each of these areas. Other studies by Keefer [13] and McBreen [14] focus specifically on perceived weaknesses with Extreme Programming (XP) [15]. They also note similar limitations to Turk et al [11], [12].

Given such perceived limitations it is clear that small software organisations require a straightforward guide to adopting, or rejecting, an agile approach to development.

### 4 Boehm and Turner's Agility/Discipline Assessment

As early as 2002 Boehm [1] had already produced an approach to assessing an organisation's suitability for agile methods. Boehm and Turner [4] present the sets of conditions under which agile and plan-driven methods are most likely to succeed:

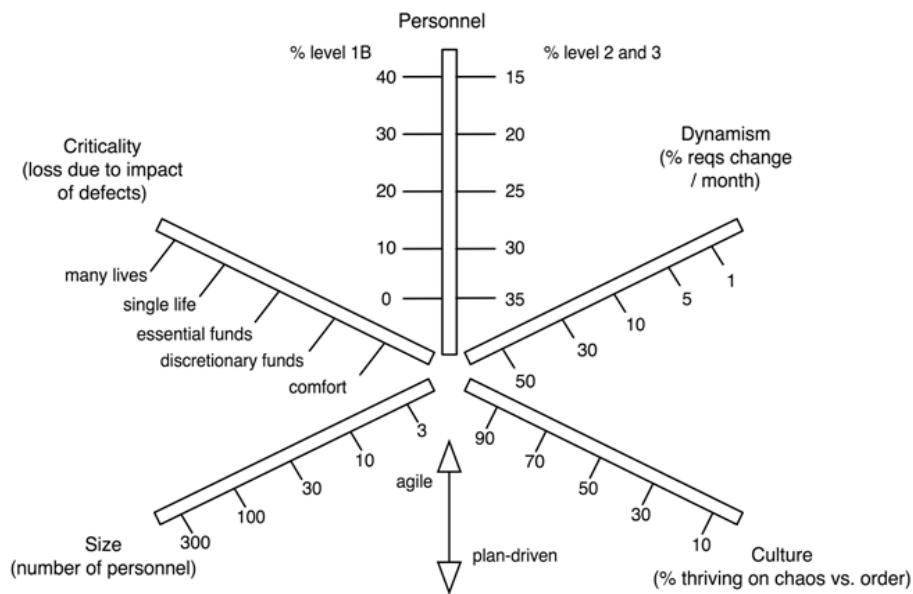
*Application characteristics* – primary project goals, project size, and application environment.

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

*Technical characteristics* – approaches to requirements definition, development, and test.

*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 and plan-driven methods are better suited to stable environments with low change. Five critical factors, as shown in Figure 2 and described in Table 1, are summarised from Boehm and Turner’s [4] analysis of the strengths of agile and plan-driven methods.



**Fig. 2.** Five critical factors affecting method selection [4, p. 56]

Building upon the five critical factors Boehm and Turner [4] describe a five-step, risk-based method. The risks are categorised as follows:

*Environmental* – risks resulting from the project’s general environment. An example is technology uncertainties in the particular project domain.

*Agile* – risks specific to the use of agile methods. For example, agile methods rely on tacit knowledge as the small number of team members communicate on a daily basis, but if there is personnel turnover the tacit knowledge also leaves.

*Plan-driven* – risks specific to the use of plan-driven methods. For example, emerging requirements will cause strain for a plan-driven method as it is structured for up-front requirements elicitation.

**Table 1.** Five critical factors described [4, p. 55]

<b>Factor</b>	<b>Agility Discriminators</b>	<b>Plan-Driven Discriminators</b>
Size	Suited to small products and teams. Not very scalable.	Methods suited to large products and teams. Difficult to scale down.
Criticality	Little testing on safety-critical products. Informal documentation and simple design may be potentially difficult.	Methods suited to highly critical products. Overkill for low-criticality products.
Dynamism	Simple design and continuous refactoring good for dynamic environments. Can be a source of expensive rework in stable environments.	Detailed plans and Big Design Up Front excellent for highly stable environments. Can be a source of expensive rework in dynamic environments.
Personnel	Requires highly skilled software engineering experts. Risky to use inexperienced people.	Requires highly skilled software engineering experts during project definition. Later in project can work with fewer experts and more inexperienced people.
Culture	Thrives on chaos.	Thrives on order.

The assessment is based on a five step process:

*Step 1.* Rate the project's environmental, agile, and plan-driven risks. If uncertain about the ratings use prototyping, data collection, and analysis.

*Step 2a.* If agility risks dominate, use a risk-based plan-driven approach.

*Step 2b.* If plan-driven risks dominate, use a risk-based agile approach.

*Step 3.* If the risks are a mixture of 2a and 2b then architect to encapsulate the agile parts. Use risk-based agile approach on the agile parts and risk-based plan-driven elsewhere.

*Step 4.* Establish an overall project strategy by integrating individual risk mitigation plans.

*Step 5.* Monitor project progress and risk/opportunities; readjust balance and process as appropriate.

A further explanation of factors and how the authors adapted the Agility/Discipline assessment as a tool to aid software process improvement in small organisations will be given in Section Five.

## 5 Servasport Case Study

The SPACE project organises regular industrial events to discuss and inform regarding agile methods. At these events, overviews of various agile methods are presented and industrial speakers who are using agile methods describe their experiences. Through such events the SPACE team promotes the Agility/Discipline assessment primarily to small software development organisations.

Servasport is a specialist sports management company providing 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 relatively successful in their product domain. On average a project will generally take between ten and twelve weeks from initial requirements to customer handover. The contract model used requires them to agree a price and time-scale before commencing the development work. If the deadline is missed further work must be completed within the original budget. Servasport had previously been operating without much specific process guidance. Such an approach worked well but their reputation in the product domain continues to increase and hence they are getting further projects, leading them to explore the benefits of agile methods.

This scenario is common and Servasport are not alone in wanting to be prepared for increased project activity and the vital revenue it generates. They have sought guidance on how to improve the management of parallel projects, prioritise changing and emergent requirements, retain fast development cycles and so forth.

Given the nature of Servasport and the products they produce, it would seem inevitable that an agile approach to development would be particularly beneficial. However, small software organisations such as Servasport cannot risk losing contracts and revenue due to the adoption of a new process that does not suit the team, product domain, or customer relationship.

### 5.1 Adapting the Assessment

Boehm and Turner's intention is "to offer a way to plan your program and incorporate both agility and discipline in proportion to your project's needs." [4, p. 99]. Although this aim suggests that a software team leader or manager should be knowledgeable and confident enough to make process improvements the Agility/Discipline assessment can also prove useful for those small software organisations.

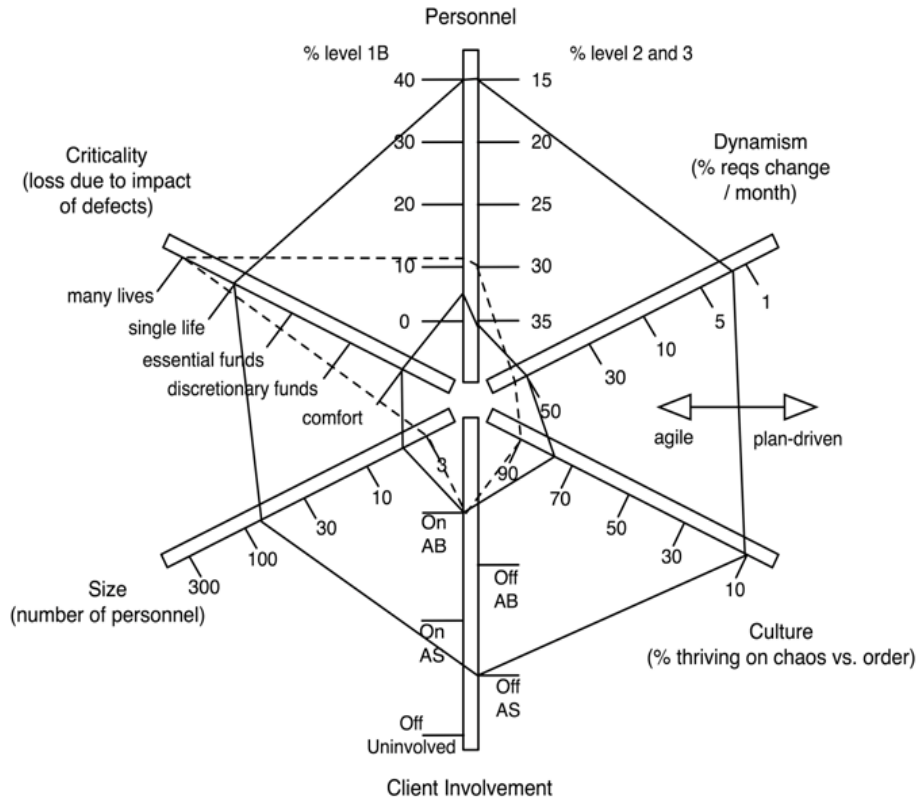
Boehm and Turner do not explicitly split their Agility/Discipline assessment into stages but for the purposes of engaging software organisations with straightforward process improvement strategies we have established two explicit stages.

The first stage was to use the five critical factors graph to reassure Servasport that they are suited to an agile development approach. Before introducing Servasport to the five factors, they were asked to describe their most crucial problems. Customer related issues were deemed most important and entailed most risk. As product providers many small software organisations can be made to feel privileged that they are getting the contract and revenue and the customer will give them initial requirements and remain practically uninvolved until the handover deadline. This scenario was also noticeable in some large software development organisations. As a result, we added a sixth factor, *Client Involvement*, to the graph as shown in Figure 3.

The sixth factor, *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 makes themselves 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.



**Fig. 3.** Six critical factors affecting method selection. This is an update of Figure 2 after surveying industrial risks.

*On AS* – Client is on-site but is an agile skeptic. They may be on-site but they are not convinced about the agile development approach.

*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 between providing the initial requirements and getting the right product delivered.

Servasport then performed a self assessment, using the six critical factors graph and accompanying instructions. The aim was to get the developers discussing process improvement and forming opinions about how their process should be improved.

Plotting the data on the graph is simple. If the project *criticality* is ‘comfort’ and the team *size* is 3, draw a line from one to the other. The other factors require more explanation. *Personnel* is an important part of the self assessment. Each team member is required to give an accurate assessment of themselves and each other. The categories are described in Table 2.

Table 2. Personnel characteristics

Level	Characteristics
3	Able to revise a method (break its rules) to fit an unprecedented situation.
2	Able to tailor a method to fit a precedented new situation. Can manage a small, precedented agile or plan-driven project but would need level 3 guidance on complex, unprecedented projects.
1A	With training, able to perform discretionary method steps (e.g. sizing tasks for project timescales, composing patterns, architecture re-engineering). With experience, can become level 2. 1A's perform well in all teams with guidance from level 2 people.
1B	With training, able to perform procedural method steps (e.g. coding a class method, using a CM tool, performing a build/installation/test, writing a test document). With experience, can master some level 1A skills. May slow down an agile team but will perform well in a plan-driven team.
-1	May have technical skills, but unable or unwilling to collaborate or follow shared methods. Not good on an agile or plan-driven team.

The *Personnel* axis requires more explanation. The outer solid line shows that approximately 15% of staff are level 2 or 3, approximately 40% are level 1B and the remaining at level 1A. An agile approach will be better supported if there is a higher percentage of level 2 or 3 staff as shown by the inner solid line and dashed line. The agile related *Personnel* risk with the outer solid line is that most of the level 2 or 3 staff will be expending much time training and overseeing level 1B staff and therefore contributing less directly to the product. As a team gains more practical knowledge with an agile development approach the *Personnel* percentages should move towards the centre indicating more level 1B staff becoming level 1A.

*Dynamism* can be an exact figure if metrics are kept or a notional estimate if metrics do not exist. Requirements changes include all functional and non-functional requirements. *Culture* is a notional estimate of how much your team or organisation likes to work on the edge of chaos or with more planning and defined procedures.

The self assessment then outlines how a graph, such as that in Figure 3, can be interpreted. The outermost solid line suggests a project that is suited to a plan-driven approach and the innermost solid line indicates a project to be suited to an agile approach. The dashed line suggests that the project would be suited to an agile approach but has a significant risk on the *Criticality* axis. When such a risk is encountered more planning is required than an agile approach typically recommends.

The second stage of the self assessment involved using Boehm and Turner's Agility/Discipline risk ratings. Each team member was asked to provide a rating for each risk item. Once each team member had completed the self assessment we collated the results and discussed with Servasport the issues arising from the exercise.

## 5.2 Primary Risks

As Figure 4 shows, the biggest risk from stage one of the assessment is the situation where there are off-site uninvolved customers who have the potential to break the

agile development approach. The aim was to find a way to bring this risk down to the dashed section of the line in Figure 4 resulting in customers who were off-site agile believers (Off AB clients). 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 needs to be more specifically defined and a workable plan implemented.

Servasport have experienced a customer who desired new functionality at the product handover stage. In this circumstance, the contract arrangement meant that Servasport 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. Figure 4 also shows that 25% of the staff are at level 2 and the rest are level 1A.

Table 3 presents the risk ratings for Servasport from the second stage of the assessment. Looking at the total risk ratings clearly shows that the company are 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. 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 as best they can.

### 5.3 Risk Mitigation as a Framework for Adopting Agile Methods

Having used the adapted two stage Agility/Discipline assessment with the company the following risks require careful management if using an agile development approach:

*Risk 1.* Off-site uninvolved customers.

*Risk 2.* Personnel turnover.

The authors believe that highlighting the risks of a software organisation focuses the need for process improvement and effectively acts as a framework for the introduction of agile methods. Such an approach enables the organisation to see that the new agile method is actually helping to mitigate real risk and hence reduces resistance to the changes required. The risk mitigation strategies developed for the company are described in the following paragraphs. These strategies introduced certain agile practices to Servasport and are a first step towards process improvement. Introducing the agile practices in small stages enables the staff to feel confident using them and provides space to tailor them to the company's specific context.

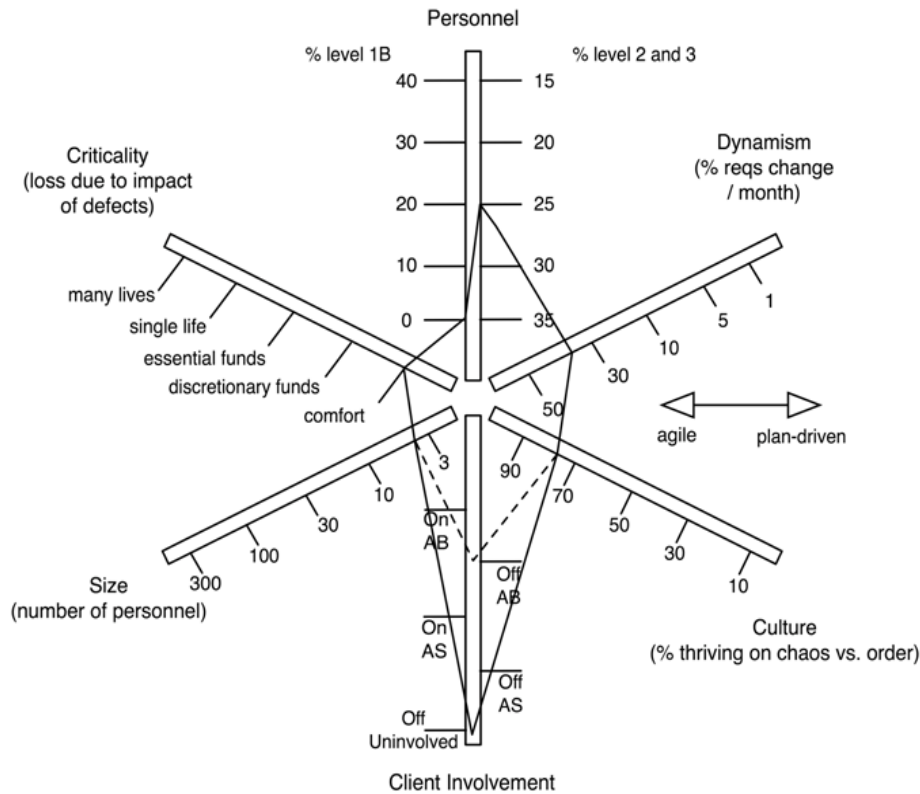


Fig. 4. Six critical factors affecting method selection for Servasport

### 5.3.1 Off-Site Uninvolved Customer Risk

Given an average development cycle of between ten and twelve weeks a risk mitigation strategy making use of incremental delivery was agreed.

- The company should have a weekly incremental delivery for at least the last three weeks of any project. The rationale for this decision is to begin moving an Off-Site Uninvolved customer to an Off AB (see Figure 4).
- The final increment shall typically result in the handover release.

This new approach should be made visible to the customer with the following requirements:

- The customer should understand that involvement in the incremental releases is contractually required. Contractual conditions with small software organisations usually favour the customer. This type of clause ensures active participation by the customer.
- User acceptance testing will be part of the incremental approach. User acceptance testing is the means by which customers can verify their requirements. The tests should be developed with input from the company and the customer. They should be written in the language of the customer, for example:

*Test 1*

“The user must be able to click the bike icon and receive new content.”

*Test 1.1*

“Receiving this new content must be in the following manner – a bulleted list shown on the left side of the news window.”

The objective is not to have the product totally finished before the incremental delivery phase begins. Rather, the company will have planned the work needs to completed in each increment. The first attempt at this was based on previous experience with the time to complete certain types of requirement. The accuracy of planning the amount of work for the increments will increase with each product.

**Table 3.** Risk ratings for Servasport

<b>Risk Items</b>	<b>Risk Rating</b>
<i>Environmental risks</i>	
Technology uncertainties	1
Many stakeholders	1
Complex system of systems	1
	<i>total = 3</i>
<i>Risk of using agile methods</i>	
Scalability and criticality	1
Use of simple design	1
Personnel turnover	4
Not enough poeple skilled in agile methods	2
	<i>total = 8</i>
<i>Risk of using plan-driven methods</i>	
Rapid change	4
Short development cycles	4
Emergent requirements	4
Not enough people skilled in plan-driven methods	3
	<i>total = 15</i>
1 – minimal risk, 2 – moderate risk, 3 – serious but manageable risk, 4 – very serious but manageable risk, 5 – showstopper risk	

The company are aware that each incremental delivery will result in issues arising from failed user acceptance tests or unspecified new requirements. In relation to this, Servasport must discern between a user acceptance test failing because the customer has implicitly changed their requirement or because the specified requirement has not been implemented correctly. Unspecified new requirements must be prioritised with previously specified requirements. In some instances the new requirement will replace a specified requirement and in other instances the new requirement will be previously unspecified. In either scenario the company must work with the customer to determine what can realistically be done in the next increment.

If new requirements arise at the last increment during final user acceptance tests then the customer must provide reasons why they were not originally specified or discovered in a previous increment. At this point it will be necessary to negotiate further increments if the customer must have the new requirements. If all the user acceptance tests pass at the final planned increment then discuss and agree a new costing for any subsequent increments.

The above strategy has helped to mitigate the risk from off-site uninvolved customers by bringing them closer to being off-site agile believers. It also makes the balance of power fairer in the working relationship as Servasport are seen to be driving certain aspects of the product development. The customer was also given the opportunity to be more involved in product development resulting in greater commitment to the product. As the company prepares for larger projects running in parallel the careful use and planning of increments will become essential if they are to continue to be successful in satisfying their customers. When such development situations arise further agile practices will be investigated for suitability.

### 5.3.2 Personnel Turnover Risk

The mitigation strategy for this risk has yet to be refined with Servasport but the following paragraphs outline the basic practices. The advantage of a small development team often relates to the natural occurrence of face to face communication. It is generally accepted that small teams benefit from the ability to communicate frequently about each other's development problems and successes. However, the disadvantages are that if a team member becomes ill or leaves for another employer the other team members find it difficult to orient themselves with often undocumented work practices and development. This scenario is common for many small software development organisations working at or near to their limits.

Contrary to the popular misunderstanding agile methods do use documentation but the emphasis is on working software. Documentation may not be comprehensive but it will relate to the essential aspects of a project. Ruping [16] suggests that agile documentation should be governed by the following principles:

- Project documentation should be lightweight and only include what is necessary.
- Necessary documents can only prove useful if they are high-quality.
- Tools and techniques for documentation are only useful if they aid the production of high-quality documents and make their organisation and maintenance easier.
- The documentation process must adapt to each specific project.

Regarding tools, Servasport use standard word processing software for the production of formal documents for customers such as contractual agreements and user manuals. They use the open source SugarCRM [17] tool as a version control system for documents. SugarCRM can also be used to manage tasks but it is not possible to produce burn-down statistics hence limiting its applicability to other agile practices. It is recommended that Servasport use a standard spreadsheet to track all tasks and manage it with their chosen version control system.

Servasport follow a standard set of coding guidelines and code comment standard. However, more important than this type of documentation is that related to the

product architecture. One of the developers is primarily responsible for deciding the shape of the architecture and the tools used to implement it. It should be evident to all members how to update such tools and integrate other tools within the architecture. In order to keep such documentation lightweight and current it is advised that a wiki [18] system be used which is accessible and editable by all team members. The wiki system used in conjunction with periodic mentored role changing will help reduce the risk accompanying personnel turnover. For example, the developer responsible for architecture changes should change roles temporarily with the developer responsible for interface design. This will at least highlight where the wiki documentation is incorrect or unclear and at most lead to more than one developer who can adequately maintain or change the architecture and interface.

The documentation could be more rigorous but, given the nature of Servasport and the product domain, what has been described above is adequate to begin mitigating the risk of personnel turnover and contribute to process improvement.

## 6 A Risk Based Process for Adopting Agile Methods

Figure 5 summarises the risk based process for adopting agile methods in small software development organisations. Given that many such organisations work near or at their limits, the process for adopting agile methods has to be minimally intrusive yet effective enough to actually begin mitigating risks. The process begins with the two stage self assessment. The results are collated by those skilled in agile methods and process improvement and the risk strategies are developed in conjunction with the software development organisation. Only at this point are agile practices introduced which relate directly to the risks resulting in the overall risk mitigation framework. Feedback and refinement is essential for continual process improvement and changing product domains. This risk based process will also help to reduce the limitations mentioned in Section 3.

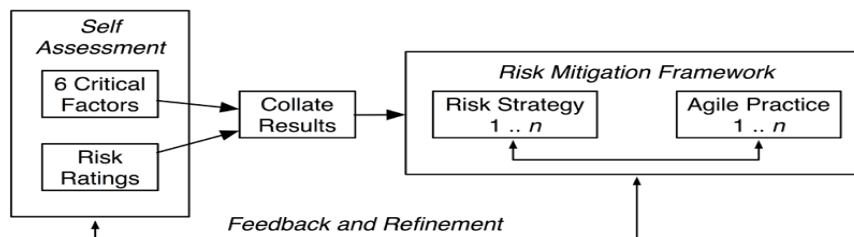


Fig. 5. Minimally intrusive risk assessment process for introducing agile methods

## 7 Conclusions

Many small software development organisations, seeking to improve the efficiency and effectiveness of their development processes, are being drawn by the hype surrounding agile methods.

The aim of this paper has been to present a method, with an accompanying case study, to assess the risks that a small software organisation has and to introduce agile practices to help mitigate these risks. Based on observation and numerous discussions with small software development organisations the approach presented, derived from Boehm and Turner's Agility/Discipline assessment, is a more reasonable attempt to introduce agile methods. The self assessment presented in Section 5.1 highlights the primary risks which then function as a framework for the new agile practices. The framework approach has the advantages of focusing the efforts of those responsible for process improvement and of reducing resistance to the required changes.

The case study has helped to show how the self assessment can be used. It has been effective at highlighting the risks and providing the framework for process improvement with agile practices. For organisations like Servasport it is more likely that such small improvements will need to be made and refined as the business grows rather than a complete change to agile methods in one step. The potential for failure is simply too great to risk complete process change in one step.

## Acknowledgements

The work described in this paper arises from the SPACE project, supported by the EU Programme for Peace & Reconciliation, administered by Co-operation Ireland.

## References

1. Boehm, B.: Get Ready for Agile Methods, with Care. *IEEE Computer*, Vol. 35(1), IEEE Computer Society (2002) 64 – 69
2. Boehm, B., Turner, R.: Rebalancing Your Organization's Discipline and Agility. In: Maurer, F., Wells, D (eds.): *XP/Agile Universe 2003*. Springer-Verlag, Berlin Heidelberg (2003) 1 – 8
3. Boehm, B., Turner, R.: Using Risk to Balance Agile and Plan-Driven Methods. *IEEE Computer*, Vol. 36(6), IEEE Computer Society (2003) 57 – 66
4. Boehm, B., Turner, R.: *Balancing Agility and Discipline – A Guide for the Perplexed*. Addison-Wesley (2004)
5. Abrahamsson, P., Warsta, J., Siponen, M. T., Ronkainen, J.: New Directions On Agile Methods: A Comparative Analysis. *Proc. 25<sup>th</sup> Int. Conf. Software Engineering*. IEEE Computer Society (2003) 244 – 254
6. Highsmith, J.: *Agile Software Development Ecosystems*. Addison-Wesley (2002)
7. Larman, C., Basili, V. R.: Iterative and Incremental Development: A Brief History. *IEEE Computer*, Vol. 36(6), IEEE Computer Society (2003) 47 – 56
8. Larman, C.: *Agile & Iterative Development – A Manager's Guide*. Addison-Wesley (2004)
9. Royce, W. W.: Managing the Development of Large Software Systems. *Proc. WESCON*. IEEE Computer Society (1970) 1 – 9. Available for download at <http://www.cs.umd.edu/class/spring2003/cmcs838p/Process/waterfall.pdf> (last visited January 2006)
10. [www.agileireland.com](http://www.agileireland.com)

11. Turk, D., France, R., Rumpe, B.: Limitations of Agile Software Processes. In: Wells, D., Williams, L. A. (eds): XP/Agile Universe 2002. Springer-Verlag, Berlin Heidelberg (2002) 43 – 46
12. Turk, D., France, R., Rumpe, B.: Assumptions Underlying Agile Software Development Processes. *Journal of Database Management*, Vol. 16(4), Idea Group Inc (2005) 62 – 87
13. Keefer, G.: Extreme Programming Considered Harmful for Reliable Software Development 2.0. Available at <http://www.avoca-vsm.com/Dateien-Download/ExtremeProgramming.pdf> (last visited January 2006). AVOCA GmbH 2003
14. McBreen, P.: Questioning Extreme Programming. Addison-Wesley (2003)
15. Kent, B., Andres, C.: Extreme Programming Explained: Embrace Change. 2<sup>nd</sup> Ed. Addison-Wesley (2005)
16. Ruping, A.: Agile Documentation – A Pattern Guide to Producing Lightweight Documentation for Software Projects. John Wiley & Sons (2003)
17. <http://www.sugarcrm.com> (last visited January 2006)
18. <http://en.wikipedia.org/wiki/Wiki> (last visited January 2006)