ASSESSING A METHODOLOGY'S PROJECT RISK MANAGEMENT COMPETENCE

E MNKANDLA (University of Johannesburg)

Abstract: Many organisations do not perform risk management in their systems development projects; hence many projects tend to be like uninsured assets. Since the massive increase in computer programs, supply and maintenance of information systems has remained a problem. The maintenance problems of these large systems were mainly due to the informal structure which was followed in the development of the applications. Systems development methodologies were therefore authored to deal with these fears or risks. It has thus been a general assumption that methodologies take care of risks in a project and generally improve the effectiveness of the products. This article presents a technique for analysing what components of a systems development methodology address what type of risks. Using this technique it is possible to determine how well a methodology addresses certain risks and therefore results in better decisions about what methodologies would be suitable for your project.

Key phrases: *Agile methodologies, methodologies, project management, risk management, software crisis, software development, software engineering*

1 INTRODUCTION

The development of computing applications has been with us for about four decades now. The demand for software applications has increased from simple applications that replaced the tasks of the human computer (Parr 1982:4) to the complex military applications for controlling weapons, such as patriot and stinger missiles to name but a few. The initial demand for software increased at such a rate that it became obvious that a more formalised approach to the development of software had become inevitable. Surprisingly, even in the twenty-first century the demand for more software still outstrips supply.

During this era of the history of computing, software development problems led to what became known as the software crisis (Brooks 1987:13; Friedman 1992:15; Glass 1998:104; Naur, Randell & Buxton 1996:5; Olerup 1991:216; Pressman 2009:12; Van Vliet 2008:6; Yeh 1991:120). The software crisis not only refers to problems associated with approaches to developing software, but also includes the broader aspects of how to maintain software. Efforts towards solving these problems led to the birth of a new discipline called software engineering ascribed to the 1968 NATO Software Engineering Conference held in Garmisch, Germany. The continued effort to solve these problems resulted in the emergence of many software development methodologies from the early 1970s to date, with the addition of contemporary systems development methodologies such as agile methodologies over the last ten to fifteen years.

While it is possible to agree that there is no one methodology that fits all systems development environments, it may be difficult to justify the continued increase in the number of methodologies. Are the methodologies effective for their intended purposes? If so, why is there a need for more and more methodologies? Can methodologies be analysed to find their effectiveness in certain hidden areas such as risk management? To answer these questions a qualitative review will be conducted that will reveal the richness of the implied concepts. Based on literature review, content analysis and experience with methodologies, this article contributes knowledge towards a better understanding of project risk management in systems development methodologies. The technique discussed here analyses a methodology's risk management ability in a given project environment.

The rest of the paper starts by providing a brief background to project risk management and systems development methodologies. Then the major contribution of this paper is explained, which is a technique for analysing a methodology's risk management ability in a given project environment. The paper ends with concluding remarks.

2 BACKGROUND

This section starts with a discussion of the meaning of what risk is in systems development projects and then continues with the meaning of methodologies and how they should guide the management of risks in projects.

2.1 Project risk management

The PMBOK defines project risk as an uncertainty that can have a negative or positive effect on meeting project objectives (Project Management Institute 2008:275). The International Organisation for Standards in its ISO31000 risk management standard defines risk as the "effect of uncertainty on objectives" (International Organisation for Standards 2009:Internet). Hillson (2009:Internet), however, argues that whilst the ISO definition for risk seems clear and unambiguous, with just five words the focus of the definition is on 'effect' instead of 'uncertainty', which is the traditional focus of the definition of risk. Hillson further argues that managing effect would turn out to be rather different from managing uncertainty and yet risk is about managing uncertainty.

There are many issues surrounding project risk management such as apparent failure by many organisations to actually implement risk planning, monitoring and control. This was reflected in various surveys such as the one conducted by Ibbs and Kwak (2000:34) in which the maturity of project management was assessed. The results showed, among other things, the fact that project risk management was the least implemented knowledge area. Such implications are due to the fact that carrying out project risk management does not necessarily have any visible artefacts and yet the lack of project risk management will certainly negatively affect the success of a project if the risk does occur.

Project risk management involves the planning, monitoring and control of risks in a project. During project planning, risk management involves developing project risk plans, identifying risk, performing qualitative and quantitative risk analysis and determining how to respond to risks (Project Management Institute 2008:53). As the project progresses, risks will need monitoring in order to implement contingency plans either when the triggers for the risks occur or the risks themselves occur. From a security perspective risk differs from a threat and a vulnerability; however, in most systems development projects literature, all these terms are included under risk.

In terms of security a threat would be anything that can take advantage of a weakness in the system intentionally or by accident to obtain, damage, or destroy what the system is trying to protect. A threat is therefore that entity which the system is trying to protect against. A vulnerability would be a weakness or gap in a system that can be exploited by threats to gain unauthorised access to a system. Hence a risk is the potential for loss, damage or destruction of a system as a result of a threat exploiting a vulnerability. When applying this whole concept to the management of projects it becomes inevitable to identify the source or root cause of a risk. Risks occur when a threat which could be internal or external to the project processes successfully exploits a vulnerability within the project processes; hence risks can be better managed by dealing with their root causes, i.e. vulnerabilities in the project processes or threats to the project processes.

2.2 Risk management in software development projects

The next issue to consider is how much guidance a systems development methodology should provide for effective planning and implementation of projects. Let's start by defining a methodology. Systems development methodologies can be defined as a group of processes used in the development of applications. Methodologies will give details of what should be done in each phase of the systems development process. You will notice that the methodologies do not necessarily specify how to do things. That level of detail is usually left to the organisation to tune the methodology to its environment by, for example, developing templates and other documents that spell out how things should be done (Adams & Barndt 1988:215; Mnkandla 2009:3).

Based on the preceding definition, should it not be expected for a methodology to specify how to manage risks in projects? The answer is that a methodology should indeed provide guidelines on how to manage risk in projects (Avison & Fitzgerald 2006:13; Song & Osterweil 1992:46; Verzuh 2011:105). For example, a methodology could say what project areas should be considered in risk identification. The main problem is that a number of systems development methodologies are silent about risk management. In fact, it is assumed that risk management is implied in methodologies.

2.3 Project risk analysis techniques

Software projects may seem to be inundated with inherent risks, but there are strong indications that with considerable care these risks can be successfully controlled (Raz, Shenhar & Dvir 2002:105). This section takes a look at some of the techniques available for analysing risk in projects.

The various processes for effective risk management in projects have been summarised by Kwak and Stoddard (2004:916) from various standards and guides as follows:

- The risk management process provided in the PMBOK (Project Management Institute 2008:273) is a good overview of the typical processes, yet it is often too generic to meet the specific needs of software projects.

- The Software Engineering Institute (SEI) has developed the Team Software Process (TSP) for the team as a whole, and the Personal Software Process (PSP) for the individual in software development projects (Software Engineering Institute 2001:Internet). Keshlaf and Hashim (2000:298) have developed models for tools to aid the software risk management process.

Jantzen, Adens and Armstrong (2006:Internet) advocate for a cost and schedule balanced approach in estimating risk for software development projects. They warn against being overly optimistic (assuming that no risk will occur) or overly pessimistic (assuming that all risks will occur). The limitation of their approach is the focus on only two knowledge areas, i.e. cost and time, and yet risk in software development projects can emanate from any knowledge area.

DeMarco and Lister (2003:12,13) agree that greater risk brings greater reward, particularly for software development projects. In their experience, a company that runs away from risk will soon find itself lagging behind its more adventurous competition. By ignoring the threat of negative outcomes in the name of positive thinking or a can-do attitude, software managers drive their organisations into the ground. They believe that the only reason people do project risk management is not to avoid risks but to enable aggressive risk-taking. DeMarco and Lister (2003:141) recommend an early start to software projects as a sure way to mitigate schedule-related risks and their technique is to identify the most common risks for software projects, such as schedule flaws, requirements inflation, turnover, specification breakdown and under-performance.

Kwak and Stoddard (2004:919) recommend the implementation of formal risk management processes to manage complex issues associated with software development projects. Their research, however, revealed that organisations face challenges in the integration of processes into the organisations due to the disparity between the theoretical nature of risk management processes and the practical challenges of organisations (Kwak & Stoddard 2004:920).

2.3.1 Specific activities for risk mangement

Boehm (1991:32) says that high-risk elements in software development projects must be identified and resolved early in the project to balance the enthusiasm for new software capabilities. He observed software project managers who applied various software development processes such as the waterfall, evolutionary and a combination of these two, and found that the successful project managers managed risk well, applying a factor called risk exposure (i.e. the product of the size of the loss and the probability of loss) to risk assessment. Ultimately what Boehm (1991:35) recommends is the identification of the top ten risk elements in a project and then developing a priority list for them, followed by risk management planning and risk monitoring.

Symonds (2011:Internet) lists the most important areas of a project in which risks must be identified as scope, equipment, technology, existing data, people, timescale and budget. Software development projects are generally known to be complex and if they happen to be large, then risk management would require a full-time project manager who has been trained in risk management (Symonds 2011:Internet). Slater (2010:Internet) warns against over-reliance on risk aspects of a project and emphasises the importance of applying risk processes that are relevant to the size and complexity of the project. Murphy (2009:Internet) advocates for an iterative risk-driven approach in order to reduce risk and increase the probability of project success for software projects. Reifer (2011:47) discusses processes for conducting risk analysis in software projects and uses a number of specific case studies such as make or buy analysis in a telecommunications case study, or a defence project contract case study, and includes a commercially off-the-shelf risk mitigation guide. In all these cases the focus is a deliberate process for the identification of risks and developing a plan for the management of the identified risks.

Published literature reveals various limited approaches used by some organisations to manage risk for software-related projects (Fairley 2009:373; Griffiths 2009:Internet; Lima 2010:349; Mcmanus, Carr & Adams 2011:140; Molloy, Dickens, Morisset, Cheng, Lobo & Russo 2012:158; Smuts, Van der Merwe, Kotz & Loock 2010:308). In some cases the poor management of risk in projects has led to estimation and

decision problems (Dodson, Sterling & Bennett 2012:60; Eisenberg 2012:2; Kholoud & Mohammed 2012:25; Portny 2010:167; Shaker 2010:Internet; Zimmerman, Katzmarzik & Kundisch 2012:27).

2.3.2 A comprehensive risk identification technique necessary

Comprehensive identification of risks in software development projects has been shown to improve the rates of success of such projects (Koopman 2010:3). The use of techniques that integrate risk management during the requirement engineering phase have been investigated by Islam (2009:6). Ma, Liu, Feng, Shan and Peng (2009:8) present a detailed analysis of risks in software projects using a model that is based on the social context of software development. They use Hofstede's national culture index and Chinese culture as an example to explain how cultural factors impact software risk management.

Hillson (2011:20) tackles the topic of risk management from the perspective of failure. He shows that the fear of failure when taken to extremes can lead to over-protectiveness which could prevent organisations from taking necessary risks or pursuing profitable opportunities in the fear that things could go wrong. Murrah (2012:5) presents a formal risk assessment model that can be applied to software development projects in the early stages of the software development life cycle. The model has been validated against thousands of postmortem projects, with applicability to any software development activity (Murrah 2012:8). Hillson (2010:8) maintains that the key to applying structured risk management is to remember that no matter how complex and uncertain things may seem in projects, risk management works.

Running through the various techniques for software project risk management, a general theme emerges that starts from the identification of risks, prioritisation of risks down to the formulation of the aversion strategy and finally the monitoring of the risks. What is clear from the literature is that there is a general concern for the management of risks in software development projects, but a comprehensive approach to incorporating risk management in the software development methodologies is lacking. To bring risk management closer to what software development teams and their project managers do, there is need to treat software project management not as a

separate group of activities that parallel software development activities, but as an inseparable part of software development. There is need therefore to incorporate project risk management into the methodologies that are used to develop software. The next section considers how clearly software development methodologies cater for risk management.

3 ANALYSIS OF RISK MANAGEMENT FACTORS IN METHODOLOGIES

This section proposes a unique technique for analysing the risk competence of a methodology. In this case methodology risk competence refers to the degree to which a methodology provides guidance on how project risk management can be implemented in a given project.

3.1 Defining the parameters of the technique

The selection of parameters used to define this technique is based on the methodology selection framework in Mnkandla (2008:101) and the PMBOK's nine knowledge areas and methodology representation as defined in Avison and Fitzgerald (2006:93). The selected parameters were deemed sufficient to capture major areas where potential risks can arise in software development projects. The reason for the PMBOK alignment is to make the technique more user-friendly since many project managers either use the PMBOK-based processes or have a reasonable understanding of the PMBOK (Lotz, Brent & Steyn 2009:253). The application of this technique assumes a detailed understanding of project risk management processes combined with a good understanding of the methodology under investigation.

Table 1 shows the parameters used for the analysis; the first column lists the project areas where most risks could come from. There are, however, other more generic ways to represent sources of risks in projects such as those discussed in Blake (2005:132). The columns under "methodology" represent the methodology elements that help in analysing a methodology to determine how risk competence is represented. The elements of a methodology may differ depending on how the methodology being investigated is represented. The analysis technique presented here looks at the main areas of a project from which risks are most likely to arise. Each of the areas will be explained in detail.

TABLE 1: RISK MANAGEMENT FACTORS

Project Area	Methodology			
	Values	Principles	Practices	Process
Time				
Cost				
Task				
Quality				
Communication				
Process control				
Resources				
External				

Source: Own compilation

3.1.1 Time management

Project time represents the total time it will take for the project to be completed. The risks related to time include poor duration estimates and poor schedule management (Schwalbe 2010:434). Through its principles and practices a methodology should emphasise the importance of good time estimations and schedule control. Some techniques and tools for these activities could be included in the methodology.

3.1.2 Cost management

Cost management involves estimating the total cost of the project. It is important to note that time and cost estimates are the two most poorly performed activities in software development projects leading to budget overruns and schedule slips, the most poorly managed risks in IT projects (Kholoud & Mohammed 2012:22). A methodology should explicitly provide guidance on what should be done to do better and more accurate cost estimates. It is worth noting that project costing can be affected by external economic risk factors such as inflation. Such factors are generally more difficult to deal with in planning, but determining their impact on the project and providing contingency reserves are important.

3.1.3 Tasks

Project tasks are derived from the total scope of the project, i.e. all the work that must be done as part of the project. Scope means all the requirements and the subsequent changes in the requirements. Requirements management should in principle enable scope control, which in turn should reduce the possibility of the risk

of scope creep. However, requirements are very difficult to manage in software development projects due to the rate at which business requirements change (Lehman & Sharma 2011:753; Shemer 1987:508; Zowghi & Jin 2010:271). Scope creep is not the only scope-related risk. The other sources of scope-related risks are poorly defined scope or work packages and incomplete definition of the work to be done (Schwalbe 2010:433). Methodologies should therefore provide clear guidelines on how to manage requirements and clarification of how to deal with possible changes in scope.

3.1.4 Quality management

Quality management involves satisfying the customer through the delivery of a product that meets the customer's requirements and is usable as intended (Schwalbe 2010:434). Quality management is therefore a rather complex undertaking considering that changing requirements are inevitable for software development projects. The changes in requirements imply that the customer's quality expectations will also change as the project progresses. Based on the definition of quality given here, since there is a direct relationship between quality and requirements, the quality plan for the project will have to be changed. Software development methodologies should therefore provide guidance for the proper planning of quality to avoid the risks of low quality assurance standards and the number of defects in the final product.

3.1.5 Communications management

Communications management involves the generation, collection, representation, dissemination and storage of project information. Communication also includes effectively disseminating project information to all the major stakeholders (Schwalbe 2010:383). Poor communication is usually the main source of stakeholder-related risks such as poor consultation with key stakeholders. A methodology should therefore provide clear guidelines for good communication planning and techniques. For instance, the phased approach to the management of risks explained in Lotz *et al*. (2009:260) can go a long way in providing risk information per phase in a project.

3.1.6 Process control

Process control involves the management of the entire systems development process and the coordination of all the project activities including change management. The main risk conditions in process control are inadequate planning, poor resource allocation, poor change management processes, lack of post-project reviews, among other things (Schwalbe 2010:433). A methodology should therefore explicitly show what can be done to manage the systems development process and its related changes effectively.

3.1.7 Resource management

Resource management involves the effective use of project personnel. The risk conditions that relate to project personnel are poor conflict management, poorly defined project responsibilities and general poor project leadership (Schwalbe 2010:433). Methodologies must therefore provide guidance for effective resource management in a project.

3.1.8 External risks

External risks are more difficult to plan for since they are caused mainly by things beyond the influence of the project, the organisation and its environment and may be beyond the control of the project manager. There are many possible risk conditions which may range, for example, from suppliers to environmental issues, natural disasters and global economic forces. Methodologies should provide guidelines ranging from planning legal contracts for outsourcing to disaster recovery, business continuity plans and other contingency measures.

3.1.9 Methodology attributes

The technique proposed here requires that each of the project areas be mapped into methodology attributes to determine whether or not a methodology provides for risk management. The major attributes of a methodology considered in this analysis technique are:

- *Methodology values:* represent the reason for the existence of that methodology, or the philosophical underpinning on which the methodology was created. The methodology values usually differentiate a methodology from others.
- *Methodology principles:* these are the general truths that guide how decisions are made in the methodology. The methodology principles provide channels through which a methodology's values are realised.
- *Methodology practices:* these are the actionable things and activities that are performed to implement the methodology's principles. It is in the set of practices that all the things that are done should be clarified. However, it is worth noting that a number of methodologies may have vaguely defined practices leading to confusion about what should and should not be done in the methodology (see Abrahamsson, Warsta, Siponen and Ronkainen (2003:3) for details on vaguely defined practices).
- *Methodology processes:* these are the series of developments in a methodology starting from the initial phases of the work to the closing phases. Most methodology processes are defined in the methodology life cycle. However, some methodologies do not have a defined life cycle and it may therefore be difficult to understand their processes.

This technique for analysing risk management in methodologies considers all these methodology attributes in order to try and exhaustively cover all characteristics of a methodology, since not all software development methodologies are clearly defined. Some methodologies may cover issues of risk in their values, while others may cover risk in their principles or practices or process, or sometimes not at all. Not all software development methodologies are represented using the attributes explained in preceding paragraphs; for detailed methodology representations see Mnkandla (2008:102).

3.2 Methodology

A qualitative research methodology was followed to provide understanding of risk competence in software development methodologies. A qualitative methodology was selected in favour of a quantitative methodology because the researcher wanted to

gain an in-depth understanding of project risk management in software development methodologies. In qualitative research methodology the method of choice was content analysis which proves useful when the research purpose is best answered through the analysis of the data in a thematic way (Cameron & Price, 2009:23).

A prerequisite for successful content analysis is that the research question must be clear and specific: do software development methodologies incorporate risk management in their processes? The information was gathered by analysing the principles, values and practices of software development methodologies to identify themes that indicate how the methodology caters for project risk management. The approach used for the analysis is detailed in the next two sections and examples for the analysis of two methodologies are given.

3.3 Analysing risk management factor for extreme programming

The technique can be applied to any systems development methodology, and Extreme Programming (XP) is used as an example to illustrate the use of the technique. According to Beck and Andres (2004:5), XP is built on the following:

- Five values: communication, simplicity, feedback, courage and respect.
- Five fundamental principles: rapid feedback, assume simplicity, incremental change, embracing change and quality work.
- Ten further principles: teach learning, small initial investment, play to win, concrete experiments, open honest communication, work with people's instincts not against them, accepted responsibility, local adaptation, travel light and honest measurement.
- Twelve practices: pair programming, planning game, whole team, test-driven development, small releases, continuous integration, refactoring, system metaphor, simple design, collective code ownership, coding standards and sustainable pace.

The question is: can it be determined with certainty whether these XP attributes would clearly guide developers to manage risk in a given software development project? Let's analyse each XP attribute against each methodology attribute.

The first row has project time management; the main risk in this case would be poor time estimates. The methodology parameters that can help in risk management are shown under each XP attribute as writing code for the simplest solution that will take less time, remembering that in XP work is done in iterations of one to four weeks, at the end of which incremental changes to the initial scope are accepted. This together with honest measurement of the work covered and working at a sustainable pace will reduce uncertainty and other schedule-related risks.

The second row has project cost management; the main risk in this case would be cost under-estimation. The methodology parameters that can help in risk management are shown under each XP attribute as cost estimations for the simplest solution. This reduces the risk of overestimation and underestimation, which are possible when estimating for the entire project. Planning is therefore done in small initial investments for each iteration. The decision to continue or not can be made at the end each iteration.

In project task management the main risk is scope creep, which involves failure to control the growing project tasks. XP, like many other agile methodologies, controls scope through values such as simplicity which encourages taking a simple workload. The concept of time-boxing is also used to fix the time within iterations and doing the tasks that fit the allocated time. This way the risk of scope creep is highly controlled.

The risks associated with project quality have to do with failure to produce a product that meets the stipulated standard. XP therefore takes care of the risks through careful feedback and a learning process that allows for rework of the product and redesign through the practice of refactoring.

The risk of poor communications management is usually singled out as the major cause of the failure of projects. XP therefore, like other agile methodologies, values communication so much that communication is done every day through short and precise stand-up meetings. XP does not clarify how to deal with external risks. Table 2 summarises the analysis.

TABLE 2: RISK MANAGEMENT FACTORS FOR XP

Project Area	Methodology			
	Values	Principles	Practices	Process
Time	Code the simplest solution that will take less time	Incremental change, honest measurement	Work at a sustainable pace	Work in iterations (one to four weeks)
Cost	Cost the simplest solution	Plan small initial investments	Cost of small releases	Cost per iteration
Task	Gather initial simplest requirements	Rapid feedback, allow for incremental change	The planning game clarifies scope	User stories per iteration
Quality	Frequent review provide feedback	Quality work, rapid feedback, honest measurement	Test-driven development, refactoring to improve quality, coding standards	Working with customer in the development team
Communica-tion	Feedback, communication	Rapid feedback, open honest communication	Metaphors, pair programming, planning game	Daily stand-up meetings
Process control	Communication	Incremental change	Collective code ownership	Iterative development
Resources	Communication, feedback, respect	Working with people's instincts not against them, accepted responsibility	Pair programming, collective code ownership, on-site customer	Small teams
External	Not covered	Not covered	Not covered	Not covered

Source: Own compilation

3.4 Analysing risk management factor for Scrum

The second example used to illustrate the use of the technique is Scrum software development methodology. According to Guang-yong (2011:220), Scrum is a continuous improvement process that helps project teams to improve on process and product quality, team dynamics and productivity, leading to on-time delivery of the final product.

According to Schwaber (2007:102), the Scrum philosophy is based on the empirical model of industrial process control used to control complex and unpredictable systems. The fundamental notion of Scrum is that the set of variables that constitute software development such as people, requirements, resources and technology are not predictable, and therefore the use of rules for a defined process in software development would not be adopted. This scenario puts software development

projects under complex unpredictable systems and the Scrum methodology then spells out the best practices that can be used to manage such processes. The Scrum attributes are:

- Product backlog: a prioritised list of project requirements with estimated times for completion.
- Sprint backlog: list of items to be completed in a Sprint.
- Sprint: a period of thirty calendar days during which a team turns the Product Backlog into an Increment (Schwalbe 2010:142).
- Daily Scrum: short daily meetings conducted to check what work has been completed since the previous Scrum meeting and report on any impediments to the project.
- Burndown charts: a trend of work remaining across time in a Sprint, release or product.
- Increment: product functionality that is developed by the team during each Sprint (Schwalbe 2010:141).

Let us determine whether Scrum would clearly guide developers to manage risk in a given systems development project by analysing each Scrum attribute against each methodology attribute. The first row of Table 3 has project time management. Each of the Scrum parameters takes care of risk as in the case of XP by estimating length of each user story and using time-boxing, giving feedback in 15-minute meetings per day.

The second row has project cost management. Each of the Scrum parameters takes care of risk by costing the project per user story, bringing the cost together for each Sprint and dealing with any cost-related issues at the daily Scrum meeting. The decision to proceed or not is made based on the cost of each Sprint.

In project task management the main risk is scope creep. Each of the Scrum parameters takes care of risk by allowing the team to work with a prioritised user story list and forbidding any changes to the list until the end of the Sprint. During the

daily Scrum meetings scope-related issues and issues to do with the rate at which work is done are discussed and resolved.

Project quality-related risks in Scrum are taken care of through activities such as conducting daily meetings in which the standards applicable to the work are used and doing product reviews before the deliveries can be accepted at the end of each Sprint.

To deal with the risk of poor communications management the Scrum parameters provide for Sprint planning meetings, the use of task boards to communicate the work being done, 15-minute face-to-face meetings for accountability and feedback, burndown charts to provide visualised progress and documentation of changes to help in the decision of the way forward.

In process control, integration of the entire project processes is facilitated through working with only those requirements that are currently known and leaving the unknown for the next iteration, ensuring that there are no changes during the current Sprint, resolving project issues and ensuring accountability during the daily Scrum meeting.

In project resources management Scrum emphasises the use of small teams and involving everyone in the project. Like XP, Scrum does not clarify how to deal with external risks. Table 3 summarises the analysis.

TABLE 3: RISK MANAGEMENT FACTORS FOR SCRUM

Project Area	Methodology				
	Product backlog	**Sprint**	**Daily scrum**	**Burndown charts**	**Increment**
Time	Estimated length per story	Time-boxing	15 minutes per day	Rate of performance per task	Every one to four weeks
Cost	Cost per user story	Cost per Sprint	Cost issues	Earned value	Proceed yes or no based on Sprint cost
Task	Prioritised user stories	No changes allowed	Scope issues	Performance rate	More user stories
Quality	Customer needs per Sprint	Standard used	Daily reviews	Quality control	Delivery acceptance

Project Area	Methodology				
	Product backlog	Sprint	Daily scrum	Burndown charts	Increment
Communica-tion	Sprint planning meeting	Task boards	15-minute face-to-face meetings	Visualise progress	Documented changes and way forward
Process control	Use only what is currently known	No changes allowed	Accountability	Task versus time	Delivery and way forward
Resources	Everyone involved in the project	Allocate per Sprint	The whole team	Team performance	Project committee
External	Not covered	Not covered	Not covered	Not covered	Not covered

Source: Own compilation

4 CONCLUSION

In this paper a technique was presented for analysing the level of project risk management for which software development methodologies provide guidance. The technique analyses the major project areas and maps them to methodology attributes. The main contribution of this technique is to get systems developers thinking deeply about the value that a systems development methodology provides for the management of project risks. Example methodologies were used to illustrate the implementation of the technique. The results show that XP practitioners do not just claim that their methodology takes care of project risk management, but XP actually provides for risk management.

Scrum methodology was also shown to cater for project risk management. Neither of these examples seem to cater for external risks. The main reason for such an omission could be that external risks are usually considered to be organisation-level issues and covered under organisational risk policies. The examples used were purposely aimed agile methodologies which are still considered relatively new and surrounded by controversy in many IT circles. The application of this technique to other methodologies would be interesting as future work.

REFERENCES

ABRAHAMSSON P, WARSTA J, SIPONEN MT & RONKAINEN J. 2003. New directions on agile methods: a comparative analysis, Proceedings of the 25th International Conference on Software Engineering (ICSE'03), pp. 2-5. Portland: IEEE.

ADAMS JR & BARNDT SE. 1988. Behavioural implications of the project life cycle. In CLELAND DI & KING WI (eds). Project management handbook. 2nd edition, New York: Van Norstrand Reinhold. pp. 206-230.

AVISON DE & FITZGERALD G. 2006. Information systems development: methodologies, techniques and tools. 4th edition. New York: McGraw-Hill.

BECK K & ANDRES C. 2004. Extreme programming explained: embrace change. 2nd edition. Upper Saddle River: Addison-Wesley Professional.

BLAKE CM. 2005. Risk management and the role that certain risks play. *Journal of Contemporary Management*, 2:128-142.

BOEHM BW. 1991. Software risk management principles and practices. *IEEE Software*, 8(1):32-41.

BOEHM BW & DEMARCO T. 1997. Software risk management. *IEEE Software,* 14(3):17-19.

BROOKS F. 1987. No silver bullet: essence and accidents of software engineering. *IEEE Computer Magazine*, 21(4):10-19.

CAMERON S & PRICE D. 2009. Business research methods - a practical approach. London: Chartered Institute of Personnel and Development.

DEMARCO T & LISTER T. 2003. Waltzing with bears: managing risk on software projects. Illustrated edition. New York: Dorset House.

DODSON LL, STERLING SR & BENNETT JK. 2012. Considering failure: eight years of ITID research. Proceedings of the Fifth International Conference on Information and Communication Technologies and Development. Atlanta: ACM.

EISENBERG RJ. 2012. A threshold based approach to technical debt. *SIGSOFT Software Engineering Notes*, 37(2):1-6.

FAIRLEY RE. 2009. Managing and leading software projects. Hoboken: Wiley-IEEE Computer Society.

FRIEDMAN A. 1992. Computer systems development: history, organisation and implementation. New York: Wiley.

GLASS RL. 1998. Is there really a software crisis*? IEEE Software*, 15(1):104-105.

GRIFFITHS M. 2009. The top five software project risks. [Internet: http://www.projectsmart.co.uk/top-five-software-project-risks.html; downloaded on 2012-03-28.]

GUANG-YONG H. 2011. Study and practice of Import Scrum agile software development. 2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN), pp. 217-220. Xi'an: IEEE.

HILLSON D. 2009. The definition debate–continued. [Internet: http://www.risk-doctor.com/docs/Hillson% 20article%20Definition%20Debate%20plus%20JPPPM%20reflections.pdf; downloaded on 2012-03-09.]

HILLSON D. 2010. Exploiting future uncertainty. Farnham: Gower.

HILLSON D. 2011. The failure files: perspectives on failure. Axminster: Triarchy Press.

IBBS CW & KWAK YH. 2000. Assessing project management maturity. *Project Management Journal*, 31(1):32-43.

INTERNATIONAL ORGANISATION FOR STANDARDS. 2009. ISO 31000–Risk management. [Internet: http://www.iso.org/iso/iso_catalogue/management_standards/specific_applications/specific-applications_risk.htm; downloaded on 2012-03-12.]

ISLAM S. 2009. Software development risk management model: a goal driven approach. Proceedings of the doctoral symposium for ESEC/FSE on doctoral symposium. Amsterdam: ACM.

JANTZEN K, ADENS G & ARMSTRONG R. 2006. Estimating the effects of project risks in software development projects. 16th International Workshop on Software Measurement and DASMA Metrik Kongress. [Internet: http://www.tassc-solutions.com/downloads/EstimatingRisks.pdf; downloaded on 2012-03-12.]

KESHLAF AA & HASHIM K. 2000. A model and prototype tool to manage software risks. Proceedings of First Asia–Pacific Conference on Quality Software, pp. 297-305. Hong Kong: IEEE.

KHOLOUD SA & MOHAMMED AA. 2012. IT project risk management: a case study of Madar a Saudi IT project. *IJITCS Journal,* 2(2):22-29.

KOOPMAN P. 2010. Risk areas in embedded software industry projects. Proceedings of the 2010 Workshop on Embedded Systems Education. Scottsdale: ACM.

KWAK YH & STODDARD J. 2004. Project risk management: lessons learned from software development environment. *Technovation*, 24(11):915-920.

LANG U & SCHREINER R. 2009. Model driven security accreditation (MDSA)for agile, interconnected it landscapes. Proceedings of the First ACM Workshop on Information Security Governance. Chicago: ACM.

LEHMAN TJ & SHARMA A. 2011. Software development as a service: agile experiences. Proceedings of the 2011 Annual SRII Global Conference, pp. 749-758. San Jose: IEEE.

LIMA AM. 2010. Risk assessment on distributed software projects. Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2. Cape Town: ACM.

LOTZ M, BRENT AC & STEYN H. 2009. Investigating the risk management potential of a stage/phase-gate project management approach. *Journal of Contemporary Management*, 6(2009):253-273.

MA W, LIU L, FENG W, SHAN Y & PENG F. 2009. Analysing project risks within a cultural and organizational setting. Proceedings of the 2009 ICSE Workshop on Leadership and Management in Software Architecture. Vancouver: IEEE Computer Society.

MCMANUS D, CARR H & ADAMS B. 2011. Wireless on the precipice: the 14th century revisited. *Communications of the ACM*, 54(6):138-143.

MNKANDLA E. 2008. A Selection framework for agile methodology practices: a family of methodologies approach. Unpublished doctoral thesis. University of the Witwatersrand: Johannesburg.

MNKANDLA E. 2009. About software engineering frameworks and methodologies. *AFRICON, 2009. IEEE Xplore Digital Library*. Nairobi: IEEE.

MOLLOY I, DICKENS L, MORISSET C, CHENG PC, LOBO J & RUSSO A. 2012. Risk-based security decisions under uncertainty. Proceedings of the Second ACM Conference on Data and Application Security and Privacy. San Antonio: ACM.

MURPHY C. 2009. Reducing risk and increasing the probability of project success. [Internet: http://www.projectsmart.co.uk/reducing-risk-increasing-probability-of-project-success.html; downloaded on 2012-04-20.]

MURRAH MR. 2012. Proposal to develop enhancements and extensions of formal models for risk assessment in software projects. Kindle edition. Amazon Digital Services.

NAUR P, RANDELL B & BUXTON J. 1996. Software engineering: concepts and techniques, Charter Publisher, in FITZGERALD B (ed). Formalized systems development methodologies: a critical perspective. *Information Systems Journal*, 6(1):3-23.

OLERUP A. 1991. Design approaches: a comprehensive study of information system design and architectural design. *The Computer Journal,* 34(3):215-224.

PARR EA. 1982. Beginner's guide to microprocessors. London: Butterworth & Company.

PORTNY SE. 2010. Project management for dummies. 3rd edition. Hoboken: Wiley

PRESSMAN RS. 2009. Software engineering: a practitioners approach. 7th edition. New York: McGraw Hill.

PROJECT MANAGEMENT INSTITUTE. 2008. A guide to the project management body of knowledge (PMBOK guide). 4th edition. Newtown Square.

RAZ T, SHENHAR AJ & DVIR D. 2002. Risk management, project success, and technological success. *R&D Management*, 32(2):101-109.

REIFER DJ. 2011. Software change management: case studies and practical advice. Redmond: Microsoft Press.

SCHWABER K. 2007. The enterprise and scrum. Redmond: Microsoft Press.

SCHWALBE K. 2010. Managing information technology projects. 6th edition. Toronto: Thomson Course Technology.

SHAKER K. 2010. The seven deadly sins of risk management. [Internet: http://www.projectsmart.co.uk/the-seven-deadly-sins-of-risk-management.html; downloaded on 2012-03-09.]

SHEMER I. 1987. Systems analysis: a systematic analysis of a conceptual model. *Communications of the ACM*, 30(6):506–512.

SLATER P. 2010. Project risk: is it all bad? [Internet: http://www.projectsmart.co.uk/project-risk-is-it-all-bad.html; downloaded on 2012-03-23.]

SMUTS H, VAN DER MERWE A, KOTZ P & LOOCK M. 2010. Critical success factors for information systems outsourcing management: a software development lifecycle view. Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists. Bela Bela: ACM.

SOFTWARE ENGINEERING INSTITUTE. 2001. Carnegie Mellon Software Engineering Institute annual report FY2001. [Internet: http://www.sei.cmu.edu/library/assets/ar2001.pdf; downloaded on 2012-03-16.]

SONG X & OSTERWEIL LJ. 1992. Comparing design methodologies through process modelling. *IEEE Software*, 9(3):43-53.

STANDISH GROUP INTERNATIONAL. 2009. The chaos report. Technical report. West Yarmouth: Standish Group International.

SYMONDS M. 2011. Project risks and how to identify them. [Internet: http://www.projectsmart.co.uk/project-risks-and-how-to-identify-them.html; downloaded on 2012-04-23.]

VAN VLIET H. 2008. Software engineering: principles and practice. 3rd edition. Chichester*:* Wiley and Sons.

VERZUH E. 2011. The fast forward MBA in project management. 4th edition. Hoboken: Wiley.

YEH R. 1991. System development as a wicked problem. *International Journal of Software Engineering and Knowledge Engineering*, 1(2):117-130.

ZIMMERMAN S, KATZMARZIK A & KUNDISCH D. 2012. IT sourcing portfolio management for IT services providers: an approach for using modern portfolio theory to allocate software development projects to available sites. *SIGMIS Database*, 43(1):24-45.

ZOWGHI D & JIN Z. 2010. A framework for the elicitation and analysis of information technology service requirements and their alignment with enterprise business goals. Proceedings of the 2010 34th Annual IEEE Computer Software and Applications Conference Workshops, pp. 269-272.