Lightweight Risk Management in Agile Projects

Edzreena Edza Odzaly[1,2], Des Greer[1], Darryl Stewart[1]
{eodzaly01|des.greer|dw.stewart}@qub.ac.uk

[1]Queens University Belfast
University Road
Belfast, Northern Ireland
UNITED KINGDOM

[2]Faculty of Information Technology and Quantitative Science (FTMSK)
Universiti Teknologi MARA, Shah Alam
MALAYSIA

Abstract— **Risk management in software engineering has become a recognized project management practice but it seems that not all companies are systematically applying it. At the same time, agile methods have become popular, partly because proponents claim that agile methods implicitly reduce risks due to for example, more frequent and earlier feedback, shorter periods of development time and easier prediction of cost. Therefore, there is a need to investigate how risk management can be usable in iterative and evolutionary software development processes. This paper investigates the gathering of empirical data on risk management from the project environment and presents a novel approach to manage risk in agile projects. Our approach is based on a prototype tool, Agile Risk Tool (ART). This tool reduces human effort in risk management by using software agents to identify, assess and monitor risk, based on input and data collected from the project environment and by applying some designated rules. As validation, groups of student project data were used to provide evidence of the efficacy of this approach. We demonstrate the approach and the feasibility of using a lightweight risk management tool to alert, assess and monitor risk with reduced human effort.**

Keywords- software risk, risk management, agile projects.

I. INTRODUCTION

The Oxford dictionary defines 'risk' as an exposure to danger, harm or loss. Risk can also be seen as an event with a negative impact that may or may not occur in future. On the other hand, risk can also be positive and invites opportunities [29]. In estimating and measuring risk, Boehm [1] defines Risk Exposure as a fundamental concept that can be used to quantify risk: Risk Exposure (RE) = Prob(UO) x Loss(UO) where Prob (UO) refers to the probability of an unsatisfactory outcome and Loss (UO) refers to the impact of the unsatisfactory outcome. A lower risk exposure can be obtained by reducing the probability or by reducing the associated loss. Risk management is a process that involves identifying risk, assessing and prioritizing risk, as well as monitoring and controlling risk. Risk is a necessary evil in the software processes, even those that are claimed to inherently reduce risk, such as in agile methods [5].

II. RESEARCH PROBLEMS

A. Traditional Risk Management

More than a decade ago it was recognized that there should be a change and new risk management discipline developed [2]. This was around the same time as agile methods started to become popular. Little work has been done to date on the role of risk management in agile methods. This may be due to a common theme in research on risk management e.g. Higuera and Haimes (1996) [3] stated that risk management is difficult to implement and is complex. The implication made is that existing heavyweight risk management is contrary and to the philosophy of agile. Thus, a lightweight or improved risk management method would be more likely to be adopted.

In a survey done in 2009 [4], we gathered responses from companies in Northern Ireland as part of an investigation of the barriers to risk management. The results concluded:

- There is no standard or commonly adopted risk management process and/or tool being used in every software development situations.
- Risk Identification was the most effort intensive process and 30% agreed that Risk Monitoring is most difficult and needs more effort.
- The most recognized barrier was that where visible (and tangible) development costs get more attention than intangibles like loss of net profit and downstream liability.

Overall, traditional risk management processes is multifaceted, complex and traditionally a heavyweight process.

B. Risk Issues in Agile Software Projects

Due to the fact that agile methods depends a lot on the credibility of the people involved in the projects [5][6] as well as their motivation in applying the agile practices [7][8], most issues encountered relate to the people and the practices involved. This echoes one of the values in agile manifesto i.e. "*individuals and interactions over processes and tools*" (Agile Manifesto). This implies that not having the right people doing the right process will be a source of risk.

Cho [9] developed some research work on issues and challenges of agile software development with Scrum, among which the following points are discussed: (i) forming a Scrum team with relevant skills; (ii) one individual with multiple responsibilities and overloaded tasks; (iii) lack of accountability where team members do not take responsibility for delayed tasks, coupled with a lack of supervision and (iv) Daily scrum meeting and monthly sprint planning are considered to be a waste of time or taking too much time.

Cockburn and Highsmith [5] highlighted that one of the most important success factors in a project is individual competency emphasizing the qualities of the people involved in the project. This is also supported by Boehm and Turner

[10] where people issues are the most critical and it is very important to address them before adopting and integrating agile practices into a project. Deemer and Benefield [11], discuss challenges of the ability of a team to provide estimation of effort in their development work especially when it is done for the first time. There are also many studies addressing issues with agile skills and personnel turnover as well as job dissatisfaction [12][13][14]. Individual motivation is important in Scrum as this leads to the team adhering to agile process practices, for example attending Daily Scrum Meetings [16]. Team member behavior, where teams fail to comply with practices, can provide early sign of risks e.g. low morale expressed during the daily meeting or avoiding discussing problems when behind schedule [17].

III. SOLUTION APPROACH

Based on the research problems discussed in the previous section, there is a strong motivation to improve the management of risk in agile projects without unduly threatening the agility of projects. In reality, contemporary risk management should be looked as an integral part of the agile process and decision making. This includes taking into account human factors such as developer skills and ability as well as their behavior in performing tasks. In the following subsections, we will explain the architecture of the Agile Risk Tool (ART) and the process flow of our approach.

A. The architecture of Agile Risk Tool(ART) Prototype

The ART model can be described in terms of two main architectural components:

(i) The *Agile Risk Tool* refers to the main engine of the tool which consists of the graphical user interface (GUI) for the Input and Output, the Rule engine and the ART agents. It interacts with the ART template, which is a template that is used to define the environment data. Once the ART template file that contains environment data is uploaded, this data can be modified using a GUI. Further explanation on ART template is discussed later.

(ii) *Environment Data* refers to the data from the project environment. Changes in this data stimulate dynamic reaction from the ART agents. To achieve this, an ART template must be created for the project environment and data from the real project environment must be translated into this template. The categories of data used for this work were 'Project', 'Team', 'Task' and 'Progress'. Any risks triggered will be stored in a risk data repository so that this data can be used in future to support risk decisions.

1) Agile Risk Tool elements

The architecture of the main engine of the Agile Risk Tool (ART) prototype has three main elements; the Input/Output, the ART agents, and the Rule engine.

a) Input/Output. Previously, issues in agile projects were discussed (Section B) and it was found that most of the problems related to the people involved and their motivation and skills in software development. Indeed Agile relies heavily on the competency of the people involved. Therefore we converted these issues to risk factors i.e. situations or events that may cause a loss to occur and therefore that we need to monitor in a project.

Table 1: Mapping Problem Identified to the Sprint Goal of the Project

Sprint Goal	Problems Identified	Identifier
In Sprint X, Task Y should be assigned to appropriate number of developers once Sprint X is started	• Pair programming • No accountability or ownership • Collective code ownership	G1: Task ownership
In Sprint X, Task Y should be assigned to a proper skilled team member	• Not enough people skilled in agile / forming Scrum team with relevant skills • Insufficient agile training	G2: Skills and Experience
In Sprint X, developer should focus on one role and one project at a time	• Multiple responsibilities	G3: Resources
In Sprint X, developer should attend Daily Scrum Meeting and provide task Y progress	• Personnel turnover • Daily meetings in Scrum ceremonies – inefficient meeting, waste of time because sometimes involves more time than usual	G4: Progress

Further, there is a need to specify the input for the project, which consists of the type of risks and the risk indicators as well as the environment data which can be used to identify the risks for the project. Thus the issues discussed earlier are transformed into a set of sprint goals (Table 1). These will later be used to define the risks and their indicators thus allowing risks to be monitored continuously.

In order to identify the sprint goal for this project, we grouped the list of issues found earlier and assigned an appropriate goal for each item. An identifier was assigned for each goal and this identifier is used throughout this work (Table 1). Sprint goal rather than project goal was used to allow each sprint to have different goals and different risks associated to each goal. For this project, we adopted and reused the first three layers of the GSRM model [18] for translating the project goals into problem scenarios (Figure 1).

At the top layer (Goal/Sub-goal layer) of the model, we developed a set of sprint goals. For this work, we proposed four sprint goals based on the problems identified (Table 1) and mapped this to the problem scenario and risks (Risk-Obstacle layer) that could possibly threaten the sprint goal. Further, we mapped accordingly between the risk and possible indicators (Assessment layer) that could later provide an alert which will trigger a risk.

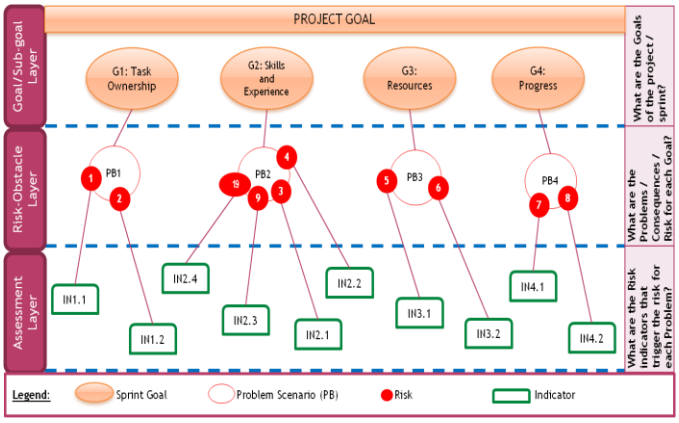


Figure 1: The Agile Risk Tool (ART) GSRM Model

In a real world situation, each project might have different goals and risks associated to the goals. However, we limited the set of goals proposed for this work to the problems identified in the literature. An example of how the ART GSRM model is applied is shown below.

Goal 1 (G1): *In Sprint X, Task Y should be done in pairs*
Problem Scenario (PB1): *During the sprint, the developer does not practice pair programming*
Risk 1: *Pair programming is not applied, single expert risk occurs*
Indicator 1.1: *When the sprint cycle is started, a task being selected in the sprint and the selected task has no pair with another developer*

b) ART agents. As discussed in the previous section, a lightweight risk management approach was introduced to reduce barriers to risk management application. One way to move towards automation is to give agents responsibility to identify, assess and monitor risk. These agents ideally should be able to autonomously react to environmental changes, where the environment in this case is the software development environment, including the set of tools being used.

There are four ART agents: Manager Agent, Identify Agent, Assess Agent and Monitor Agent. The goal and purpose of each of these is discussed below.

- *Manager Agent* acts as an intermediary between the other three agents. It manages and executes rules, notifies the Identify agent if any risk is triggered, gets data from the Environment and passes data requested from the agents.
- *Identify agent* is notified if any risk is triggered. It requests from the Manager agent what risk has been identified and notifies the Assess agent.
- *Assess agent* is invoked by the Identify agent and its goal is to estimate the Risk Exposure (RE) of the identified risk where RE = Probability (P) x Impact (I). The identified risk will then be ranked as High, Medium or Low and the Monitor agent is notified to take subsequent action.
- *Monitor agent* is invoked by the Assess agent with some data: RE and rank of the identified risk. The Monitor agent will establish the location of the identified risk along with the owner of the risk. These data are then displayed in the *Risk Register*.

The Risk Register acts as a screen to display all identified risk data. Data displayed in the Risk Register can be recorded and saved in the Risk data repository. The documented risk data can be used in future to plan and mitigate risks for the future projects.

Table 2: Rule template

Goal	The sprint goal that needs to be defined before the project starts
Problem scenario	A possible risk event that associated with the sprint goal
Consequences	The penalty if the risk is occurred
Indicators	The events or measures that forewarn of the risk event and their values
Repository/ Data	The list of repository of environment data involve in this risk event
Rule(s)	The list of rule(s) that trigger the risk event
Risk name	The unique name for the risk

c) Rule engine. In this subsection, we use two methods: risk drivers and generated rules to identify input for the Rule engine.

Earlier, we summarized problems and issues and mapped these to a set of sprint goals. In the ART GSRM model presented earlier, we derived a set of indicators for the identified problems and risks. As such, we used the indicators to generate rules to then develop inputs for the rule engine. The indicators are determined beforehand using the Rule template. Table 2 shows the template of a rule and risk drivers (indicators) for a problem scenario. Each problem scenario proposed one or more possible risk event that is associated with a sprint goal for the project. Sprint goal is important in the sense that using sprint goals to identify how this environment data could be used as indicators of threats to the goals and trigger the risks. Therefore, we generated the rules that use the indicators / risk drivers to identify events that cause loss (delay/extra cost/loss of value) i.e. risks. The problem scenario was derived from the research problem discussed in previous Section B and summarized in Table 1.

This also represents a standard template that allows the manager to define the Sprint Goal, Problem Scenario, Consequences, Indicators, Repository, Rules and Risk Name before the project starts. Each problem scenario proposes a possible risk event that is associated with a sprint goal for the project. Defining a sprint goal is important in that it allows us to identify how data from the development environment can serve as an indicator to a risk event in the project. Hence, the rules were generated using the indicators to identify risk based on the objects defined in the environment data. For this work we started by employing details from the ART GSRM model which consists of sprint goal, problem scenario, risks and indicators and further defined the consequence of each problem if the risk where to occur. Subsequently, the rule is constructed to produce an alert for the risk.

2) The Environment data

As far as the agile development process is concerned, XP and Scrum are the most widely used agile methodologies [19][20].

Table 3: List of selected data used in this work

Data/ Objects	List of Attributes
Product / Project*	Project ID / Name Project Start date / End date User Story ID / Name User Story Estimation User Story Priority User Story Owner ID / Name Task ID / Name Task Estimation Task Priority Task Owner ID / Name Task Status Task Paired By (Pair Programming)
People**	Team Member ID / Name Total no. of Role in a Project Total no. of Project Involved Programming Skill Level Agile Experience Level
Progress**	Attendance in Daily Standup Meeting Daily Progress Report on Assigned Task

*Refers to data available in both EM and RS
**Additional data collection that is required in this work

Further an agile development survey [20], stated that agile companies use a wide variety of agile tools and 60% of them are currently using an agile project management tool. The tool choice might be different ranging from a simple spreadsheet to commercial tools such as VersionOne. Tools help the project manager to have better visualization in delivering the project and in more interactive manner. Therefore, we studied two agile project management tools, Rally software (RS) and Extreme manager (EM), considering what is accessible in terms of configuring their product features and data design . In both cases information was gathered on the environment data model to include their process model, the objects used, and the attributes and values for those objects. The outcome from this was a more generalized definition of the available data.

Since the list of data available was massive, we have screened it and show only the data that are used, as shown in Table 3. Both tools when compared, have almost the same objects and attributes, although the naming convention used might be slightly different. These data were then translated into the ART template in form of objects and attributes, where the value of each attribute can be collected from the real project. The ART template represents as data within the project environment in which changes in the project environment are captured dynamically by the ART agents.

B. ART Process

In order to support the ART process the ART prototype tool was developed and used to demonstrate the reactions of the ART agents towards the changes in the environment data following the execution of the set of rules built from consideration of goals. The ART process flow is depicted in Figure 2.

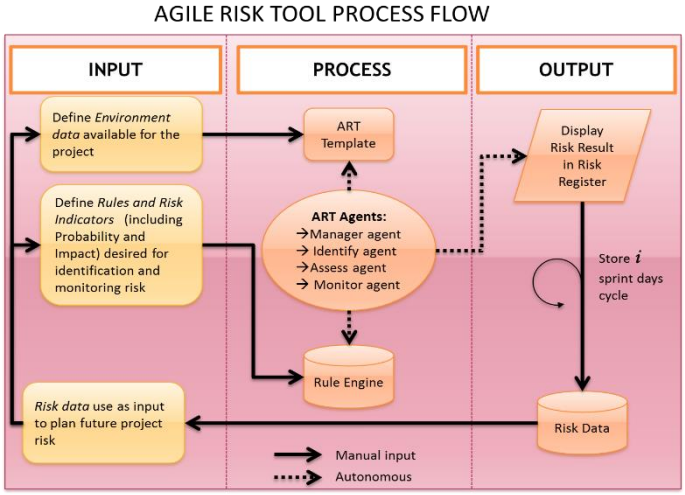


Figure 2: ART Process describing the flow or steps starting from defining the Input and storing the Output in the Risk data file

1) Input. This process is started at the Input stage. At this stage, there are two main inputs needed: defining the environment data available for the project and defining the rules and risk indicators to monitor risks.

a) Defining the environment data available for the project

Firstly, one should define what environment data is available in the project. Since we are constrained by the modern Software Engineering environment and the need for agility, we are limited to data can be easily collected. An example of the form of environment data in a project includes user stories name and id, task name and id, task priority, task owner and so on. The collected data are then translated into the ART template. The ART template consists of objects and attributes used in this work as shown previously in Table 3. It contains data and values collected from the project environment.

b) Defining the rules and risk indicators

The definition of rules and risk indicators for the project allows one to identify what risks to monitor for this project and from which data or indicator that the risks could possibly be triggered. At this stage, one can either add new rule or add an existing rule where the rule was created from a previous project.

2) Process

At the Process level, the ART agents will monitor the risk by acknowledging any rules or risk indicators triggered as informed by the ART template. The ART agents will initiate communication between them. Messages are passed according to request and each agent will notify another agent in prompting any further action to be taken.

Rules and the environment data are dynamically editable. In the event where changes need to be made, one can modify the environment data (which has been translated into the ART template earlier) as well as the risk rules and indicators using the provided main screen area. On the other hand, when developing possible risks associated with rules and risk indicators, one might find the environment data used to be insufficient to detect certain risks. In some cases, a small change in collection of the environment data would allow defining or detecting more risks. For example, adding the information on developer's skill will allow monitoring the developer's programming capability especially in completing high priority task. An example of a rule syntax that can be used is, *"IF the developer skill level is 'Low' AND the developer involved with a 'High' priority task, THEN there is probability a risk of the task cannot be completed on time because of the developer's poor programming skill".*

ART agents will react dynamically to input data, process the input by assessing any risk triggered and produce a risk result in the Risk Register.

3) Output. At the Output stage, the risk data are stored in the Risk data repository. The risk data also can be captured daily up to the *i* no. of days in a sprint and the data will be saved in spreadsheet format. Later, one can use this risk data, analyse the risk according to project and use the analysed data as an input for identifying future project risk as shown earlier in Figure 2.

This application tries to support Continuous Risk Management (CRM) [21]. Applying the ART process accompanied with the designated tool will help the project manager to manage risk continuously. This is where, when changes take place in the environment data, these are captured by the ART agents who constantly run updates on the risk data and display the results in the risk register. As far as the CRM is concerned, manual implementation of this technique can be minimized and the monitoring is moved towards being autonomous.

IV. CASE STUDIES

Two case studies were used to validate the approach and tool support. The first case study was developed in 2011 involving 38 undergraduate students, assigned into 6 teams with 6 or 7 developers each. The second case study was developed in 2012 and involved a total of 56 undergraduate students with eight groups and each group consisting of 5 to 8 developers. The case studies were used to demonstrate the ART Process Flow (Figure 2) and the tool. All groups were required to develop the same product requirements. Likewise, all groups were given the same product requirements, in both case studies. In the theoretical part of the course, students received lectures on general agile development practices with an emphasis on Scrum. During the course, students were required to build a large software artefact using Microsoft.NET technologies using an industrial strength environment adopting both agile project and software engineering practices. This includes applying important Agile project management practices such as Pair Programming, Test Driven Development, Release and Iteration Planning and Refactoring in their software project.

The most vital part of the process is to determine its environment data and risk rules for the project. The sources used in this case study are as shown in Table 3, the actual physical sources being:

- *Hartmann-Orona Spreadsheet [15]* – a widely used spreadsheet tool, here providing data to capture Sprint Backlog Information on tasks, their estimates and progress status. It also provides team member information such as working time and activity details;
- *Sprint Backlog* – a document containing a list of user stories, story points and dependencies;
- *Scrum Minutes of Meeting (Daily)* – teams were required to keep data on ScrumMaster and work progress. In Agile Life Cycle tools this data would likely be available from the tool;
- *SVN [25] Repositories* - Directory and Files Versioning, Commit Files/Code, Details of commits and changes;
- *Source Code (C#) & Resharper [26]* quality metrics.

A small sample of the risk rules defined on the data using the ART prototype is shown in Figure 3.

Risk Name	Rules [Object.Attribute] == [Value]	Prob Score	Imp Score
Pair programming	PROJECT.PROJECTSTATUS = Completed, TASK.PAIREDBY = ""	3	5
Task ownership	PROJECT.PROJECTSTATUS = Completed, TASK.TOTALOWNED > 2	1	1
High priority task assigned to inappropriate team member cannot be completed on time	TASK.PRIORITY = High, TEAM.SKILLLEVEL = -1	5	4
Overload tasks can cause difficulty in time management	PROJECT.PROJECTSTATUS = Completed, TEAM.TOTALNOROLE > 1	5	1
Developer absent in meeting possible of employee turnover	PROJECT.PROJECTSTATUS = Completed, PROGRESS.DAILYMEETING = N	1	3
No progress report	PROJECT.PROJECTSTATUS = Completed, PROGRESS.IMPEDIMENTDATA = N	1	3
Unable to understand agile process and meet the target	PROJECT.PROJECTSTATUS = Completed, TEAM.AGILE = false	3	3
Unable to comply with the agile process	TEAM.AGILE = true TEAM.AGILE_EXPERIENCE=Very Poor	1	1

Figure 3: Sample of Risk Rules used in Case study

Running the tool will then generate a risk repository at any point. Figure 4 shows a sample screen shot of the risk register showing risks that have been identified at the end of a sprint.

Given the novelty of this study, it was expected that there would be some issues raised while performing it. The first issue that we found at this stage was the difficulty in matching the data from the studied tools with the data available in the student project artefacts. The archived artefacts available however, did not provide as much data as the studied tools. Nevertheless, we found that the archived artefacts contained enough useful information, particularly related to the sprint backlog and the user stories, breakdown of the tasks, details of the developer responsible for a task and so on. In addition, the goal of the study was to demonstrate the approach and tool support, not applicability to every data item collected in mainstream tools.

Based on the lessons learnt from the first case study, some improvements were made to data collection but also rules were modified. One rule on pair programming rule was modified dues to the observation that not using pair programming in very small tasks did not constitute a significant risk. The ability to modify the rules as needed demonstrates that the solution approach and tool support can dynamically respond to changes, as is required in agile projects.

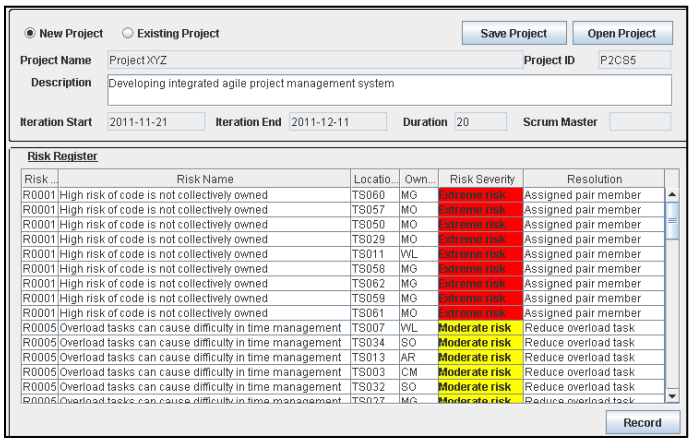
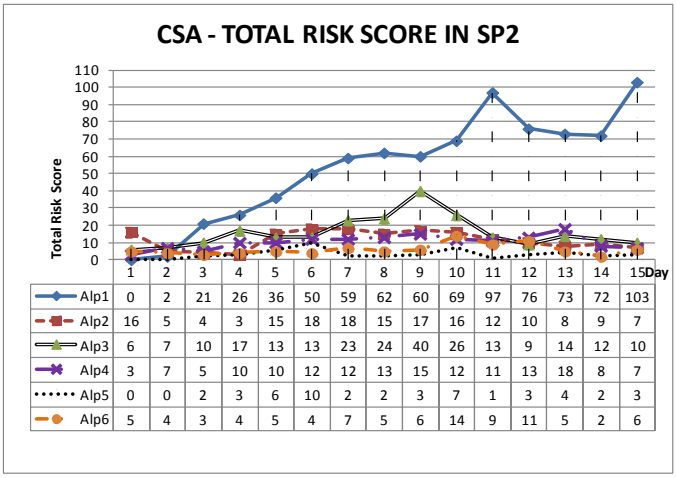


Figure 4: Risk Register

Risk data derived using the tool displayed in a Risk Register and can also be recorded and saved in the Risk Data Repository. A further output from the tool is a means of assessing the total risk in the project at any point or in a post sprint review as shown in Figure 5. This includes information on the breakdown of risk identified each day. Total Risk Score is a metric based on the generic severity score of a risk item and size of the task it is related to. It provides results on counting of risk daily and cumulative risk counting in a sprint.

Normally one would expect risk should decrease (burn down) over a sprint, but using Figure 5 a sprint review for Team Alp1 would demonstrate clearly that that particular sprint for that team was problematic. Further, the reports provide useful insights in correlating identified risks to agile practices.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Alp1	0	2	21	26	36	50	59	62	60	69	97	76	73	72	103
Alp2	16	5	4	3	15	18	18	15	17	16	12	10	8	9	7
Alp3	6	7	10	17	13	13	23	24	40	26	13	9	14	12	10
Alp4	3	7	5	10	10	12	12	13	15	12	11	13	18	8	7
Alp5	0	0	2	3	6	10	2	2	3	7	1	3	4	2	3
Alp6	5	4	3	4	5	4	7	5	6	14	9	11	5	2	6

Figure 5: Total Risk Score graph of Case Study Alpha (Sprint 2)

V. STUDY VALIDITY

Since the study presented introduces a new approach in managing risk in agile project, the main issues are focused on the internal threats. The first internal threat is in terms of the accuracy of the measured data, especially because the data used was based on historical artefacts. Further, confirmation of this data was not possible as the project had already been completed at the time of analysis. Secondly, the approach used entailed manual collection and translation of data from archived artefacts into the ART tool. This human effort was required before the ART agents could begin reacting towards environment data as they were designed to work in. This effort could be minimized by selecting a proper individual in the team to conduct this process, for example the Scrum Master in a Scrum project. One step taken to ensure the quality of the study was that cross checking was done from time to time with the Product owner to confirm perception based on his observation. Considering external validity threats, the risk management approach and tool supports were designed to be as general as possible so that this is applicable in general to agile project environments. This includes taking into account two popular agile project management tools studied for this work so that the approach is as applicable as possible to other contexts but also lightweight and unobtrusive to the team daily activities. Nonetheless, no claim can be made of good fit with tools not studied. Additionally, the study used student project data rather than industrial data. Hence, there will be arguments whether this is applicable to a real world environment.

VI. CONCLUSION

In this paper we presented a novel approach to manage risk in agile projects. We provide a contemporary and lightweight risk management approach which consists of the ART prototype and process flow. We validated the approach by using student project artefacts. An interesting by product of the research is a series of observations on how non-compliance with agile principles can increase risk to a project and also negatively affect the quality of the software product. This will be a subject of further analysis. In future, we plan to improve this approach by adding knowledge therefore helping in automatic learning and, decision support regarding risks, as well extending to other risk management steps.

REFERENCES

[1] Boehm, B.W. 1989, Tutorial: Software risk management, IEEE Computer Society Press.

[2] Kontio, J. 2002, ""Risk Management: What went wrong and what is the new agenda?," in Kontio, J. and Conradi, R. (Eds.) Software Quality - ECSQ 2002, Quality Connection - 7th European Conference on Software Quality" .

[3] Higuera, R.P. & Haimes, Y.Y. 1996, Software Risk Management.

[4] Odzaly, E.E., Greer, D. & Sage, P. "Software risk management barriers: An empirical study", Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd Interlocation.

[5] Cockburn, A. & Highsmith, J. 2001, "Agile software development, the people factor", Computer, vol. 34, no. 11, pp. 131-133.

[6] Nerur, S., Mahapatra, R. & Mangalaraj, G. 2005, "Challenges of migrating to agile methodologies", Communications of the ACM, vol. 48, no. 5, pp. 72-78.

[7] Layman, L., Williams, L. & Cunningham, L. 2006, "Motivations and measurements in an agile case study", Journal of Systems Architecture, vol. 52, no. 11, pp. 654-667.

[8] Conboy, K., Coyle, S., Wang, X. & Pikkarainen, M. 2010, "People over process: key people challenges in agile development", IEEE Software.

[9] Cho, J. 2008, "Issues and Challenges of agile software development with SCRUM", Issues in Information System, vol. 9, no. 2.

[10] Boehm, B. & Turner, R. 2005, "Management challenges to implementing agile processes in traditional development organizations", Software, IEEE, vol. 22, no. 5, pp. 30-39.

[11] Deemer, P., Benefield, G., Larman, C. & Vodde, B. 2010, "The scrum primer", http://assets.scrumtraininginstitute.com/downloads/1/scrumprimer121.pdf, vol. 1285931497 accessed 14 March 2014.

[12] Boehm, B. & Turner, R. 2003, "Using risk to balance agile and plan-driven methods", Computer, vol. 36, no. 6, pp. 57-66.

[13] Melnik, G. & Maurer, F. 2006, "Comparative analysis of job satisfaction in agile and non-agile software development teams" in Extreme Programming and Agile Processes in Software Engineering Springer, pp. 32-42.

[14] Melo, C., Cruzes, D.S., Kon, F. & Conradi, R. 2011, "Agile team perceptions of productivity factors", Agile Conference (AGILE), 2011 IEEE, pp. 57.

[15] Hartmann-Orona Scrum Spreadsheet, www.bryanavery.co.uk/file.axd?file=2009%2F5%2FSprint+Backlog.xls, accessed 14 March 2014.

[16] Hossain, E., Babar, M.A., Paik, H. & Verner, J. 2009, "Risk identification and mitigation processes for using Scrum in global software development: A conceptual framework", *Software Engineering Conference, 2009. APSEC'09, Asia-Pacific*, IEEE, pp. 457.

[17] Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., Tesoriero, R., Williams, L. & Zelkowitz, M. 2002, "Empirical findings in agile methods" in *Extreme Programming and Agile Methods—XP/Agile Universe 2002* Springer, pp. 197-207.

[18] Islam, S. 2009, "Software development risk management model: a goal driven approach", *Proceedings of the doctoral symposium for ESEC/FSE on Doctoral symposium*, ACM, pp. 5.

[19] Ambler, S. 2006, Results from Scott Ambler's 2006 Agile Adoption Rate Survey, www.ambysoft.com,accessed 14 March 2014.

[20] VersionOne 2013, *"7th Annual State of Agile Development Survey"*, www.versionone.com, accessed 14 March 2014.

[21] Dorofee, A.J., Walker, J.A., Alberts, C.J., Higuera, R.P. & Murphy, R.L. 1996, *Continuous Risk Management Guidebook*.

[22] Runeson, P. & Höst, M. 2009, "Guidelines for conducting and reporting case study research in software engineering", *Empirical Software Engineering*, vol. 14, no. 2, pp. 131-164.

[23] Wohlin, C., Host, M., Henningsson, K., 2003, "Empirical Research Methods in Software Engineering," *Empirical Methods and Studies in Software Engineering*, R. Conradi and A. Wang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, pp 7–23.

[24] Yin, R. K., 2009, "Case study research: Design and methods", vol. 5. SAGE Publications.

[25] Subversion, http://subversion.apache.org/, accessed 14 March 2014.

[26] JetBrains, Resharper, http://www.jetbrains.com/resharper/, accessed 14 March 2014.

[27] Rally, http://www.rallydev.com, accessed 14 March 2014

[28] Hindsa, Extreme Manager, http://www.hindsa.com, accessed 14 March 2014

[29] Snyder, C. S., 2013, A User's Manual to the PMBOK Guide. John Wiley & Sons.