Lightweight Risk Management: The Development of Agile Risk Tool Agents

Edzreena Edza Odzaly[1,2], Des Greer[1], Darryl Stewart[1]

[1]Queens University Belfast, University Road, Belfast, Northern Ireland, UNITED KINGDOM
[2]Universiti Teknologi MARA, Faculty of Computer and Mathematical Sciences,
77300 Merlimau, Malacca, MALAYSIA

{eodzaly01|des.greer|dw.stewart}@qub.ac.uk

Abstract. Risk management is an important process in Software Engineering. However, it can be perceived as somewhat contrary to the more lightweight processes used in Agile methods. Thus an appropriate and realistic risk man-agement model is required as well as tool support that minimizes human effort. We propose the use of software agents to carry out risk management tasks and make use of the data collected from the project environment to detect risks. This paper describes the underlying risk management model in an Agile Risk Tool (ART) where software agents are used to support identification, assess-ment and monitoring of risk. It demonstrates the interaction between agents, agents' compliance with the designated rules and how agents react to changes in project environment data. The result shows that agents are of use for detect-ing risk and reacting dynamically to changes in project environment thus, help to minimize the human effort in managing risk.

Keywords: Software Risk Management, Agile Risks, Agile Projects, Software Agents.

1 Introduction

Risk management is recognized as a key process area in the Software Process. Most risk management literature relates to heavyweight plan-driven processes and typically as-sumes that, for example, requirements have been agreed and signed off in advance of development. On the other hand Agile software development uses an iterative approach to software construction, aimed at reducing development time, prioritising value, while improving software quality and inherently reducing risk [1]. This paper intended to demonstrate the idea of software agents to help manage risks in project development. As a start, we highlighted the issues identified in risk management. Later, the proposed Agile Risk Tool (ART) model is discussed which focusing on the development of the tool. This includes how the risk management activities are decomposed into agents as well as the interaction between risk agents. The list of risks triggered in the project is then presented in the risk register and is available for display at the dashboard. This

paper introduced new method where software agents can be used to detect risk and react dynamically to changes in agile project environment.

2 Research Problems

2.1 Traditional Risk Management

Risk management in research articles is always acknowledged as being of utmost im-portance. To determine what is needed we used existing work [2], on an investigation of the barriers to risk management. The results in that investigation concluded:

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

Despite the acceptance that risk management methods enhance system development performance, nonetheless little support is to be found on the provision of these methods [3]. It was argued that the methods of managing risk in software development are not comprehensive as they deal with specific types of risk [4]. Besides, despite many well known risk management approaches having been introduced, risk management was still reported as not being well practiced [5][6]. As reported in [7] discussed the most com-mon risk management approaches found in the literature and highlighted practices such as checklists, analytical frameworks, process models and risk response strategies. Many researchers have conducted research in tailoring risk management, providing various approaches. However, only a few studies have been reported to integrate risk manage-ment with contemporary software development. One study discovered that there was still plenty of work to be done due to the fact that the integration of risk management and software development process was still at its initial stages [8].

2.2 Risk Issues in Agile Software Projects

It is clear that people issues are the most critical in agile projects and that these must be addressed if agile is to be implemented successfully [9]. Indeed, one of the most im-portant success factors in an agile project is individual competency [1]. Additionally, estimation of effort is a consistent challenge in agile development work, especially when it is done for the first time [10] and there are issues with agile skills and personnel turnover, as well as job dissatisfaction [11][12][13]. In Scrum individual motivation is very important and influences how diligent team members are; for example in attending Daily Scrum Meetings [14]. Recognising non-compliance with established practices can provide early signs of risks e.g. low morale expressed during the daily meeting or avoid-ing discussing problems when behind schedule [15].

3 Solution Approach

As a result of the issues identified, there is a strong motivation to improve the manage-ment of risk in agile projects without reducing agility in projects. In reality, contempo-rary risk management should be able to be integrated into the agile process to support decision making.

3.1 The Agile Risk Tool (ART) Model

The development of the ART model started with the establishment of a view of how risk management may apply in an agile environment. Figure 1 below depicts an over-view of the resulting model.

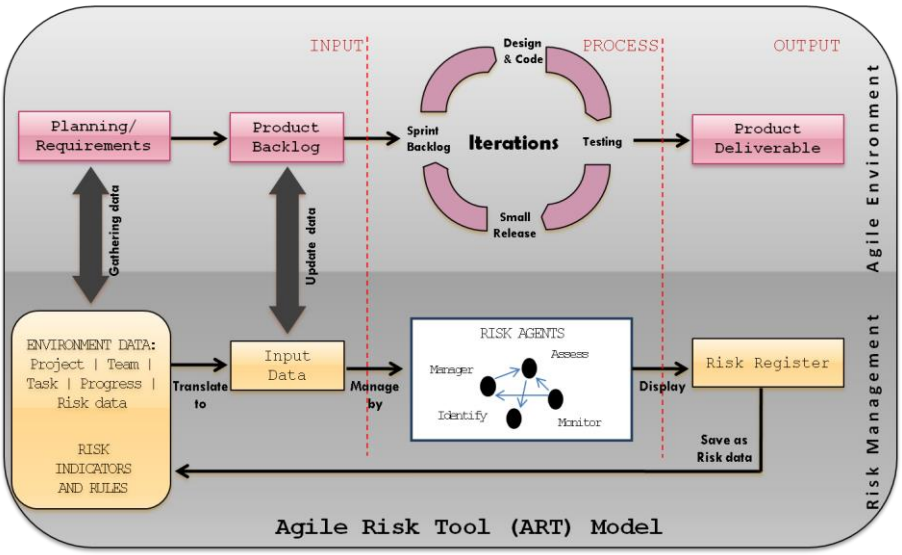

Fig. 1. Agile Risk Tool (ART) Model describing the application of Risk Management in Agile environment

The model represents how risks are gathered and managed throughout the agile project. During the *Input stage*, the agile process begins with planning and requirements gather-ing. At this stage, while preparing the project, at the same time, the gathering of risk data can commence. *Requirements* in agile processes are most often represented as user stories. These are textual descriptions that contain the customer's specification of needs for the required system. A *product backlog* is a subset of these requirements that will be selected from based on priority.

The *environment data* used contains:

- A *project* in this context is a set of user stories, the membership of which is not fixed at any point of its lifetime. Each project relates the unique project name of the pro-ject, a set of goals for the project, when it started and when it ended.
- A *team* is a set of persons where each person consists of a set of attributes describing the person. Each team is working to achieve the goals of the project. For each team member there is specific information, for example on the type of skills that the team member possesses and also their levels of expertise in defined skills, stated as an in-teger;

- User stories are divided into tasks. A *task* refers to a textual description of the task associated with the estimated hours of completion, the name of the person responsi-ble for the task and the progress for the task;
- *Progress* refers to additional information on the progress of a specific task as report-ed by the person responsible for the task. This includes information on attendance of the team member in the Daily Scrum Meeting and whether progress or an impedi-ment is reported for the task;

- *Risk data* represents information on risk captured by the tool. The information in-cludes the name of the risk, its severity, the owner of the risk, location of the risk as well as the date the risk is triggered and resolved.

The *risk indicators and rules* refer to a set of predefined risk factors brainstormed by the team at the early stage of the project and encoded as rules (this will be further discussed in the next subsection). The risk indicators contain a textual description indicating a threshold or state that will trigger the risk. One example might be where a high priority task is selected in the sprint by a developer with too low a predefined skill threshold. *Rules* contain a list of conditions for an event encoded into IF/THEN statements. Later, this information is stored in the rule engine. *Input data* refers to a set of collected data from the environment and translated into a set of templates readable by the tool.

During the *Process stage*, the project proceeds as iterations which include sprint back-logs, design and code, testing and small releases of the product requirement. Iterations contain are time-boxed into fixed length durations of development. *Risk agents* (or ART agents) will manage the risk based on the input data defined earlier. This risk process is autonomous, where software agents; identify, assess and monitor risk based on the input data from the environment. Once any risk is triggered, risk data will be displayed in the Risk register. Any changes or updates to the environment will affect the risk data (whether or not the risk is flagged up).

At the *Output stage*, the final risk data can be obtained after the delivery of the product and during a Sprint review meeting. The risk register provides a view of all identified risk data. At the end, the data displayed in the Risk Register can be recorded and saved in the Risk data repository where this information can be used to plan future projects.

The model has been demonstrated further and used as part of the work in [16]. This is where the ART architecture proposed was demonstrated in order to explore the applica-tion of risk management in agile application. This paper however, focused on the devel-opment of ART agents used at the *Process Stage*.

The development of ART Agents.

One way to move towards automation is to give software 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 de-velopment environment, including the set of tools being used.

In order to reduce barriers in risk management application, a lightweight risk manage-ment approach is needed. The newly proposed approach includes three main steps in risk management; risk identification, risk assessment and risk monitoring. The rationale of doing so was twofold (i) to develop a realistic and acceptable risk management pro-cess that can fit into the agile methods (ii) an empirical study [2] confirmed the most complicated steps in managing risks were risk identification and risk monitoring. In addition, prior to this section evidence is established that contended that risk manage-ment was difficult mainly due to the required human effort. Given this, the aim is to substitute some of the human involvement with autonomous software agents with the goal that these could manage risk and minimize the need for manual input. Automated agents can therefore help ease the work load in managing risk, specifically in identify-ing, assessing and monitoring risk.

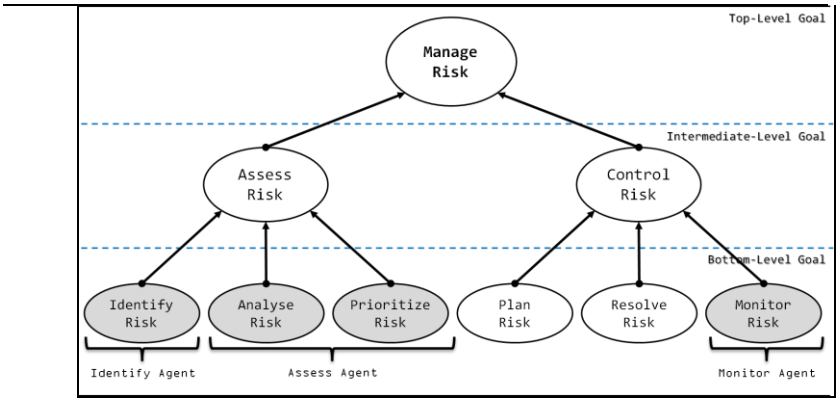

Fig. 2. Risk decomposition graph for the Agile Risk Tool (ART) agents of four risk manage-ment activities

Decomposition of risks into activities is commonplace. One example discussed in [17] used decomposition of risk into conceptual elements like risk factor, risk event, risk outcome, risk reaction, risk effect and utility loss. More recently a top down goal de-composition technique is described in [18] and [19]. Indeed Boehm's tutorial on risk [20] decomposes risk management into activities. In this work the category or type of agents used was derived based on initial agent goal decomposition as shown in Figure 2, based on Boehm's work.

The generic aim of this work is to find ways of lowering the barriers to application of risk management. One of the objectives is to use the agents since agent behaviour is more adaptable and can act on behalf of the project manager of the agile project. In this case, some of the effort of the project manager is replaced by agent execution such that

they will react automatically according to their own goals. In identifying goals for the agents, the top level goal is started in order to apply risk management in software devel-opment project, particularly in agile projects. This goal is further decomposed into two intermediate sub goals; assessing risk and controlling risk. These sub goals are then decomposed into six smaller sub goals; identify, analyse, prioritize, plan, resolve and monitor. As a result of the decomposition of the goal, agents were assigned based on the smallest sub goals which supported the top level goal. Since the most effort intensive steps identified earlier were identification and monitoring, for the meantime, both sub goals were selected in addition to analyse and prioritize goals as highlighted in Figure 2. Note that here that only the bottom level goals are engaged; the assumption being that top and intermediate level goals might have largely a controlling function but nonethe-less have their own goals on how lower level agents should interact.

Further ART agents were developed for this work as four agents; Manager Agent, Iden-tify Agent, Assess Agent (combines analyse and prioritize goals) and Monitor Agent. This is depicted as in Figure 3 that shows the interactions (communicate via passing message) between Manager agent and the Identify, Assess and Monitor agents. Depend-ing on the data from the environment, the agents react to detect risk dynamically through rules execution, where rules are invoked from the rule engine. The ART agents' communication is described further as below.

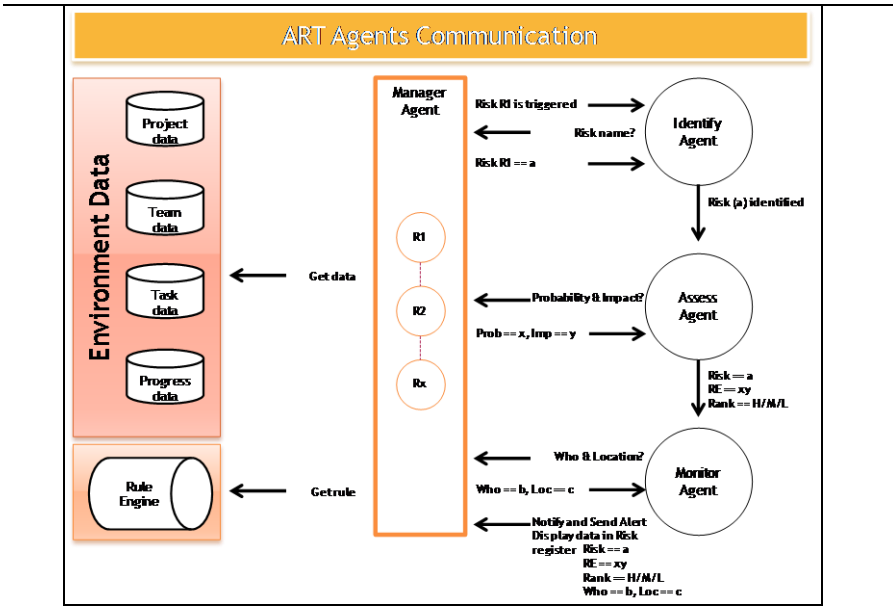

Fig. 3. The communication between the ART agents and how they interact within the environ-ment data and rule engine

There are four ART agents and each of them has a designated goal assigned to them. 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, gets data from the Environment and notifies Identify agent if any risk is triggered.

- *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 which later can be recorded and saved in the Risk data repository.

Process.
At the *Process Stage*, 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. An example of the ART agents' communication was introduced earlier in this chapter (Fig-ure 3).

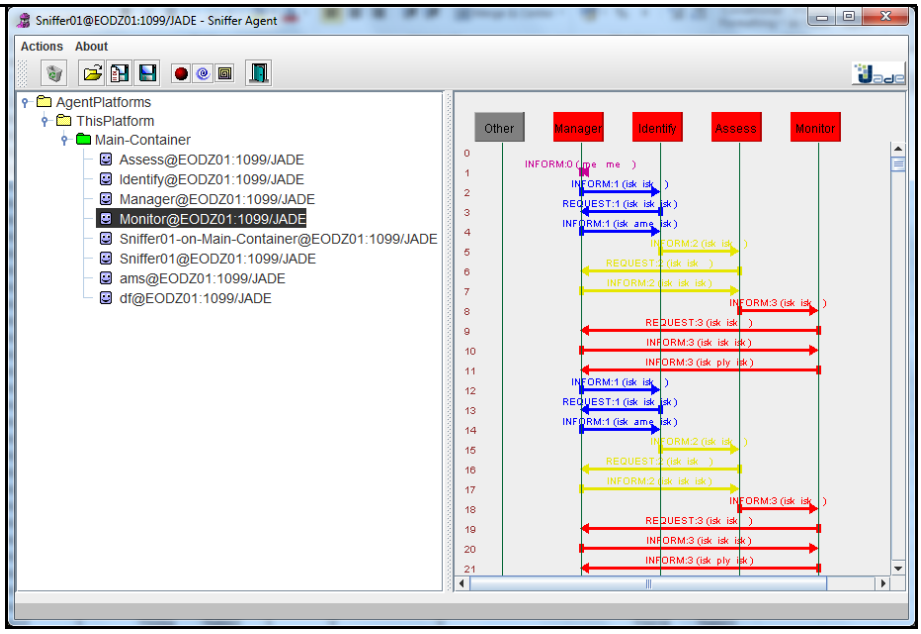

Fig. 4. Sniffer Agent

Figure 4 show an interaction between the ART agents starting when a risk is triggered. The figure shows the agents passing message using Sniffer agent in the JADE platform.

True to its name, sniffer agent is a purely java application that tracks messages in the JADE environment. It is useful when debugging the agent behaviours and for analysing message passing using in the sniffer GUI [21].

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 espe-cially 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.

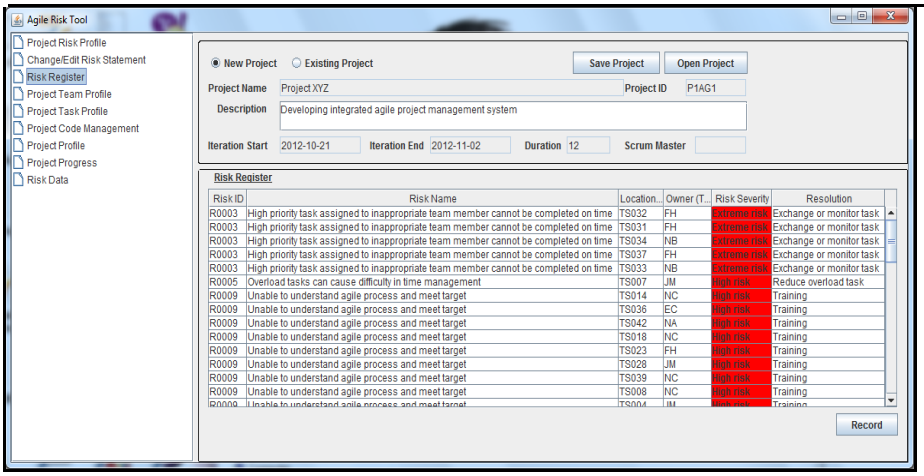

Fig. 5. Agile Risk Tool - Risk Register

Output.

The idea of a Risk register has been defined by [22] who states that "*the risk register has two main roles. The first is that of a repository of a corpus of knowledge… The second role of the risk register is to initiate the analysis and the plans that flow from it*". While [23] reported that very few development and construction of risk registers alt-hough it is commonly used in Risk Management. As such, risk register developed in this work can represent as a risk dashboard in which one can see a list of risks triggered by the ART agents. The Figure 5 shows an example of risk register used as the visualiza-tion of output in this tool.

4 Conclusion

In this paper we presented a novel approach to manage risk in agile projects. This work provides several significant investigations on the problems and issues in risk manage-ment specifically in agile projects. The development of the ART agents has been demonstrated in order to reduce effort in managing risk. The ART model demonstrated in [16] moves the body of knowledge forward via novel contributions towards building a reliable model of risk management. The approach is necessarily supported by a proto-type tool to manage risks in example agile projects.

This approach however, to the authors' knowledge and understanding has never been applied in risk management especially that aimed at reduction of the human effort in risk management and to provide as much autonomy as possible. In addition, the result-ing risk management process is naturally lightweight since each software agent is design to achieve a designated goal i.e. to identify, assess, prioritize or monitor risk. This paper has led to use designated software agents to facilitate the risk management process. Therefore, this work demonstrates the potential of autonomous computing being applied to risk management where software agents have been used to assist the human oriented and complex risk management process. In future, this work aimed to comprehend the physical implementation of the ART model and tool support, where there is a need to integrate this with existing Agile Project Management tools, perhaps as a plug-in, so that automated risk management can be fully realised. This would allow more practical risk management whereby while a project runs in the fore-ground, software agents are in the background ready to manage risks.

5 References

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

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

[3] Ropponen, J. & Lyytinen, K. 1997, "Can software risk management improve system devel-opment: an exploratory study", *European Journal of Information Systems*, vol. 6, no. 1, pp. 41-50.

[4] Bandyopadhyay, K., Mykytyn, P.P. & Mykytyn, K. 1999, "A framework for integrated risk management in information technology", *Management Decision*, vol. 37, no. 5, pp. 437-4location.

[5] Ibbs, C.W. & Kwak, Y.H. 2000, "Assessing project management maturity", *Project Man-agement Journal*, vol. 31, no. 1, pp. 32-43.

[6] Pfleeger, S.L. 2000, "Risky business: what we have yet to learn about risk management", *Journal of Systems and Software*, vol. 53, no. 3, pp. 265-273.

[7] Bannerman, P.L. 2008, "Risk and risk management in software projects: A reassessment", *Journal of Systems and Software*, vol. 81, no. 12, pp. 2118-2133.

[8] Nyfjord, J. & Kajko-Mattsson, M. 2008, "Integrating risk management with software development: State of practice", *Proceedings, IAENG International Conference on Software Engineering, BrownWalker Press, Boca Raton, USA* Citeseer.

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

[10] Deemer, P., Benefield, G., Larman, C. & Vodde, B. 2010, "The scrum primer", accessed 14 March 2014.

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

[12] 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 Soft-ware Engineering Springer, pp. 32-42.

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

[14] 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.

[15] 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 Springer, pp. 197-207.

[16] Odzaly, E.E., Greer, D. and Stewart, D., 2014. Lightweight Risk Management in Agile Projects. In *SEKE* (pp. 576-581).

[17] Kontio, J. 1997, "The RISKIT method for software risk management, version 1.00", *Computer Science Technical Reports, University of Maryland, College Park, MD, USA*.

[18] Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F. & Mylopoulos, J. 2002, "Modeling early requirements in Tropos: a transformation based approach" in *Agent-Oriented Software Engineering II* Springer, pp. 151-168.

[19] Dardenne, A., Van Lamsweerde, A. & Fickas, S. 1993, "Goal-directed requirements acquisition", *Science of computer programming*, vol. 20, no. 1, pp. 3-50.

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

[21] Bellifemine, F.L., Caire, G. & Greenwood, D. 2007, *Developing multi-agent systems with JADE*, Wiley.com.

[22] Willams, T.M. 1994, "Using a risk register to integrate risk management in project definition", *International Journal of Project Management*, vol. 12, no. 1, pp. 17-22.

[23] Patterson, F.D. & Neailey, K. 2002, "A risk register database system to aid the management of project risk", *International Journal of Project Management*, vol. 20, no. 5, pp. 365-374.