DE GRUYTER OPEN

Information Technology and Management Science doi: 10.1515/itms-2015-0010

2015 / 18

Software Delivery Risk Management: Application of Bayesian Networks in Agile Software Development

Ieva Ancveire[1], Ilze Gailite[2], Made Gailite[3], Janis Grabis[4]
[1, 2, 3, 4] *Riga Technical University*

***Abstract* – The information technology industry cannot be imagined without large- or small-scale projects. They are implemented to develop systems enabling key business processes and improving performance and enterprise resource management. However, projects often experience various difficulties during their execution. These problems are usually related to the three objectives of the project – costs, quality and deadline. A way these challenges can be solved is project risk management. However, not always the main problems and their influencing factors can be easily identified. Usually there is a need for a more profound analysis of the problem situation. In this paper, we propose the use of a Bayesian Network concept for quantitative risk management in agile projects. The Bayesian Network is explored using a case study focusing on a project that faces difficulties during the software delivery process. We explain why an agile risk analysis is needed and assess the potential risk factors, which may occur during the project. Thereafter, we design the Bayesian Network to capture the actual problem situation and make suggestions how to improve the delivery process based on the measures to be taken to reduce the occurrence of project risks.**

***Keywords* – Agile software development, Bayesian networks, project risk management, project success, software delivery.**

I. Introduction

Since the advancement of agile methods in software development processes, they have been considerably applied to projects of different application domains. Although their application has become more popular because of the demand for developing flexible information technology solutions faster, there are still several problems to be solved and explored [1], [2].

While traditional software development models are based on precisely and qualitatively defined requirements and subject to strict regulations, agile is not fully defined [3], [4]. Nowadays, there is no definite consensus on the need for risk management within the agile approach. This has led to believe that a software risk analysis is irrelevant using this approach. Many follow the approach to manage risks through the natural sprint progression after they have manifested into issues [5]. Nevertheless, irrespective of the methodology used in software development, there is a possibility for project to fail [6]. If the reasons for unsuccessful project outcome using traditional project management are incomplete and changing requirements, lack of user involvement, resources and executive support caused by communication and interaction issues between developers and stakeholders, then in agile multiple software development problems arise due to incomplete understanding of its principles that include collaboration between business and IT as well as organisation of work. It is noted that an agile approach does not guarantee for project to be successful and risk management should be considered a significant remedy in project management and software development [4]–[7].

Project risks are usually identified maintaining a list; thus, their mutual interaction is rarely investigated. Many different quantitative evaluation methods can be used for research, seeking reasons for project success or failure. One of the methods that should be mentioned is a Bayesian Network (BN). Prosperous project management, progress and closing are important to both business and technology representatives [6]; thus, the purpose of the present research is to investigate agile project risks, their interaction and dependency using BN in order to reveal project success factors.

The reminder of the paper is organised as follows. Section II summarises the related work. Section III describes the method used for further research on agile risk management. Section IV reports a case study and issue identification. Section V presents the results applying the developed model on actual information technology project. In conclusion, Section VI provides with a summary of results obtained and opinion on modelling suitability for agile development.

II. Related Work

Serviceability and high-quality are meaningful characteristics of software. In order to deliver information technology solutions more successfully, many studies and papers have been written, and various models and methods have been developed. Speaking of these different techniques, many have been applied to software project planning, management and development.

BNs have found numerous applications in software development, especially, in relation to iterative development practices. They are viable means for assessment and prediction in the case of uncertainty. Generally speaking, this approach has been applied to fields such as investigation of software influencing factors, management process overview, implementation of project management methodology, release planning, risk assessment, decision-making support, quality assurance and prediction.

The impact factors that affect the agile development cycle should be evaluated constantly in order to ensure high-quality software on time. For these purposes, a model has been introduced for agile software development release planning and project health monitoring [2]. However, these influencing factors can be accompanied by issues. A procedure that

describes how to detect a variety of problems in software development processes through a conceptual network has been explored and discussed in literature as well. This model has been successfully applied to agile project management and planning using Agile-Scrum methodology, managing various processes and achieving the success of the solution [4]. Complementing the above-mentioned considerations, a pattern is developed that assists a project manager to implement this method in the company. It provides the responsible person with information about the potential problems and issues that can arise during the project or through the management process. It helps manage the business and development teams to improve chances of project success [7]. In order to integrate an effective risk management process, a model has also been developed for software process risk assessment using BN. There is a study related to the decision-making in risk management systems and their necessity, showing that they support the process and the occurrence of risks, assessing both complexity and dependence [8]. Despite the importance of software quality, the management of it is difficult. To ease this process, an approach has been introduced in order to assess and predict software quality, using activity-based quality models [12].

Our challenge is to implement and model the software delivery process and consequent risk interdependence in practice, by constructing a BN. The goal is to show how the constructed network could help in specific project risk management. We intend to investigate which risks significantly affect the quality of software delivery, analyse what benefits the model can provide and prove that the application of BN reflects the real situation of the problem domain in projects using agile.

III. Bayesian Networks

A BN also called a belief network is an acyclic graph structure that describes relations between variables, containing different causes and effects to ensure formal description of problem domain [9]. It consists of two major parts: (1) graphics including nodes and values of the occurrence of particular event; (2) conditional probability tables covering dependencies between the graph nodes (see Fig. 1). Probability table is defined and built for each node of the network. It is based on predecessors. Dependency values between nodes could be settled as Boolean, arrangement or integral meaning truthfulness, order and grouping or range [2], [8], [10].

Examining node *E* (see Fig. 1), probability tables for a network are constructed taking into account predecessors of particular node finding whether events are met or not.

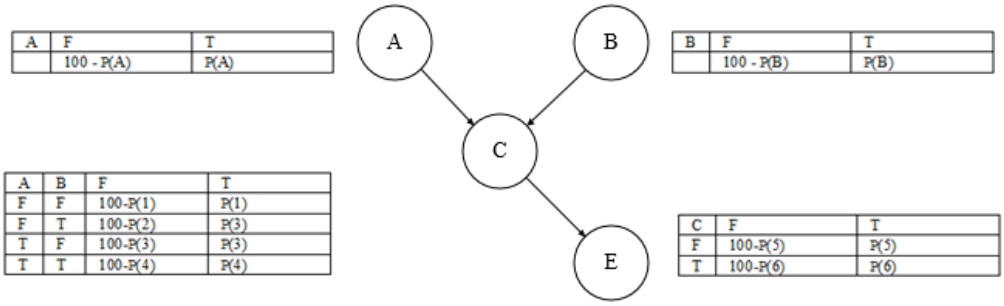

Fig. 1. Expanded Bayesian network model with conditional probability tables.

The probabilities are controlled by the Bayesian formula, where *A* – hypothesis or probability of a particular event, and *B* – evidence or condition of related node [11].

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (1)$$

Let us find the probability for true event *E* to prove Bayesian formula (1), assuming that related events *A* and *B* are also true (*A*, *B* = 1). The probability to find is *P*(*E* = 1, *A*, *B* = 1). There is a necessity to use probability tables for solving this situation. If *A*, *B* of the network are true (see Fig. 1), but *C* is unknown, both scenarios are necessary to be described, meaning that the end probability for node *E* equals the sum of both situations (*C* = 0 *AND C* = 1).

$$P(E = 1|A = B = 1) = P(E = 1, C = 0|A = B = 1) + \quad (2)$$
$$+ P(E = 1, C = 1|A = B = 1)$$
$$P(C = 0|A = B = 1)P(E = 1|C = 0) +$$
$$+ P(C = 1|A = B = 1)P(E = 1|C = 1)$$
$$\big(100 - P(4)\big)P(5) + P(4)P(5)$$

The BN modelling concept includes three node types: behavioural, decision and utility. Behavioural nodes represent events that could be met. Decision nodes describe actions that could be done when a particular case occurs, but utility nodes are related to satisfaction. Belief graphs are typically used in situations where any kind of information is unknown or decision-making support is needed. They can also be used in anticipation of the event probability. Combined with both static and qualitative data, they provide understandable formal characterisation of problem scope, making sensitive information or data analysis possibilities available in case of the occurrence of adverse events. The network also includes event probability tables for the aforementioned nodes, taking into account the number of ancestors. Once nodes and probability tables are defined, the model can be constructed. After construction, the model can be used for performance simulation of various scenarios to test and optimise the model as well as to assess which events affect the results of the particular situation [4], [10]–[13]. We will evaluate software delivery problems in agile development, defining the main

project risks that affect project delivery in each software delivery iteration.

IV. Case Study

In order to demonstrate that the risk analysis is important and substantial in any project affecting its success and failure, BN application possibilities will be justified in a real agile software development situation where the project faces problems associated with low-quality software deliveries due to negligent risk identification. To explore these opportunities, it is important to find answers to the following questions: (1) Does the dependency analysis using BN help reflect the real situation of the software delivery process? (2) How does BN help improve software delivery process? (3) What are the key risks associated with the need to manage and monitor them through the agile project delivery process? (4) Is the BN suitable for agile project risk management? (5) Whether and how does the quantitative risk analysis using BN ensure project success?

A five-step procedure is used to analyse the case. The software delivery process is analysed in the first step. That includes risk identification and reasoning about their dependencies. In the second step, the results of the first step are used to construct a network for quantitative evaluation for each risk factor. The constructed graph structure must be validated and experiments with it must be carried out as well, looking at various situations provided for comparison with the previous software delivery outputs, resulting in those situations where attention is required. Then experimental, forecast and actual results should be compared to obtain conclusions whether this kind of risk identification and analysis is needed to improve the project management, solve problems, meet and satisfy the customer's demands. Finally, opinion on this matter should be provided, explaining how difficult such a risk analysis is carried out and how suitable it is for agile project management.

Justification will be based on a project for the communications and telecommunications industry. The goal is to develop an information system that provides tangible and intangible resource management, implements different kinds of information collection into a single solution, ensures the increase in business value and reduces the paper flow in the company. This solution also provides integration of two databases, eight system modules and six subsystems. Further work is based on the development data of one subsystem.

The success of the project, regardless of the selected design methodology, is described by a number of factors influencing the project. The project under review has seen some problems which are opposite to generally defined agile, including financials, requirements, scope, project team, planning and organisation of work, system architecture and feedback.

Project development is regulated by a fixed price, meaning that all changes are implemented without the involvement of additional funds. Therefore, customer requirements tend to change, causing situations when client and supplier are unable to agree on the necessary functionality. Frequent changes in the project scope are also indicators that can affect a successful go-live of the project. There is a large amount of functionality to be developed and an excessive focus on details of the initial development may result in the loss of target. Although the development team is small, the team members and the client have different levels of expertise, personal interests, commitments and support in problem resolution. Agile also requires good planning and organising work. In this case, time and work planning have not been determined by the project management method, in which the compliance and knowledge of it are required but not observed. There is no complete understanding of the system architecture and technologies used as well. This adversely affects the outcome of the project. Deficiencies arising from servers, applications and database must be taken into account to avoid environmental gaps and standard software vulnerability. Finally, feedback has a significant influence on project progress. If this is missing, then project progress and service cannot be delivered or are at a struggle, taking into account communication problems between team members and the client.

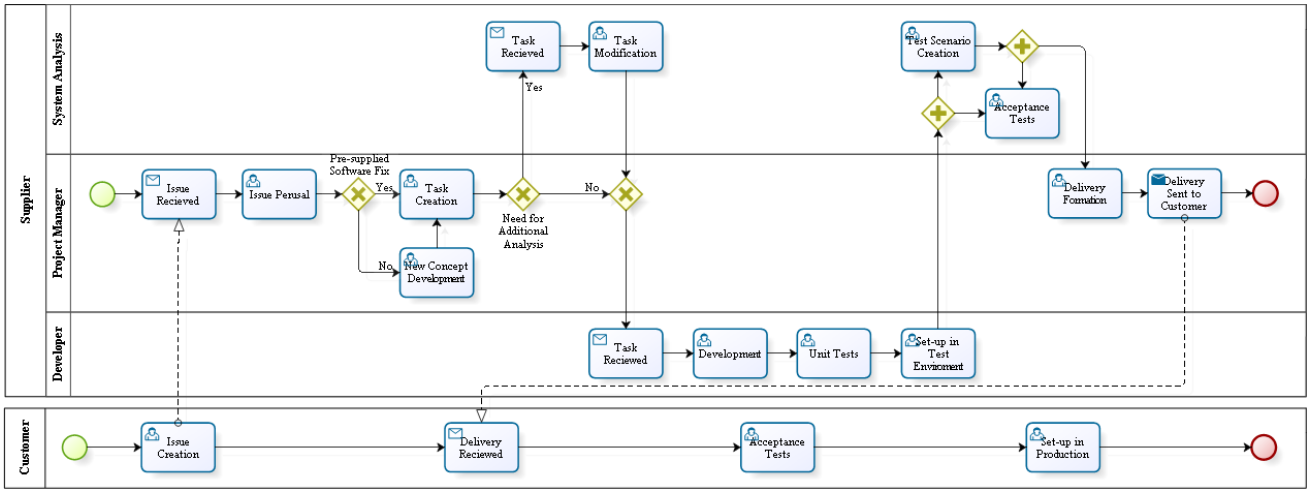

Fig. 2. Software delivery process.

Software delivery process (see Fig. 2) is initialised by the client's request. With the help of messaging, the request reaches the supplier's project manager whose role is to examine the incoming request and decide whether a new conception is needed or it is possible to immediately start the development. Regardless of the decision, the project manager creates an issue and delegates it to a developer or a system analyst for further work. The system analyst is responsible for functional clarification, issue modification and submission to the developer. Next activities involve the development, unit testing and elaboration set-up in the test environment where the system analyst performs acceptance tests. Another task for the system analyst is to create test scenarios for the client. This is always done during the acceptance test phase. When all the aforementioned activities are completed, the project manager forms the delivery and sends it to the customer. In turn, when the delivery is received, the client performs acceptance tests. In case of successful completion of the tests, the client installs the new functionality in the production environment. If the acceptance tests are unsuccessful, the client creates new requests. If the number of returned requests is greater than 50 % of the requests included in the delivery, it is classified as unsuccessful.

Software delivery to the customer consists of the following: (1) Mandatory delivery – the new system functionality and change requests are delivered to the customer once a week; (2) Additional delivery – pre-supplied software fixes are delivered once or multiple times a week, based on issues found during the acceptance tests; (3) Exceptions – if there is no scheduled change or new functionality development, pre-supplied software fixes or high priority fixes can be delivered as part of mandatory delivery.

The amount of functionality, change requests and the number of fixes are variable and can be different for each iteration (see Table I) due to various factors that occur between delivery packages and existing delivered software.

TABLE I

DATA FROM DELIVERY JOURNAL

Month	**Feb.**					**Mar.**					**Apr.**				
Week	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Change Requests														2	2
New Functionality	1	2	4	10		8	10	10	4	9	2	2	3	5	5
Fixes			11	10		10	10	20	10	10	3	10	10	14	19
Returned Requests	1	1	8	9		7	7	12	8	15	3	6	7	15	17
Delivery Type	M	M A	M A	M 2A		M	M	M	M A	M	M	M A	M	M A	M 2A
Month	**May**					**Jun.**					**Jul.**				
Week	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Change Requests							1				1				3
New Functionality						1	1						1		
Fixes		17	13	23	54	16	5	12		11		3		5	4
Returned Requests		10	5	8	20	10	4	7		6	2	1	3	1	1
Delivery Type		M	M	M 2A	M 2A	M 3A	M 2A	M		M	M	M	M	M	M

V. RESULTS

A. Risk Identification

Project risks could be defined as uncertain events or conditions that hold a positive or negative effect on at least one of projects objectives – time, cost, scope or quality. A risk factor is any attribute, characteristic or exposure that increases the likelihood of project to succeed or fail. Even though risk management is not mandatory in agile projects, there are multiple risk factors that could be defined for iterative development projects related to integrated risk management [11], [14], [15]. Risk factors for agile projects usually are unclear objectives, ineffective stand-up meetings, inadequate prioritisation of requirements [14], etc. We use different risk factors for agile software delivery risk identification (see Table II) to formally describe the main problems of the project (see Section IV).

TABLE II

RISK FACTOR ANALYSIS [14]

No.	**Event**	**Predecessor**	**Successor**	**Risk Factor**	**Problem**
1	Project Owner	–	4, 5	Unavailability of Project Owner	Project owner is not available onsite; Communication takes place over the telephone, e-mail or issue tracking system
2	Project Price	–	4, 8	Difficult to Execute Fixed-Priced Project	Changes and fixes are implemented without the involvement of additional funds
3	Requirements	–	5	Requirement Conflict	Developer is unable to agree with the customer on the functionality and its operation within a week
4	Unclear Project Objective	1, 2	6, 7	Lack of Communication Between Team and Client	There is a large amount of functionality to be developed and if it is not possible to clarify the requirements it may result in the loss of target
5	Unclear Requirements	1, 3	6, 7, 22	Lack of Communication Between Team and Client; Lack of Documentation	Without the customer's comments or revisions on time, it is not possible to qualitatively identify the necessary fixes resulting in confusion which might establish a system functionality that does not meet the real requirements

No.	Event	Predecessor	Successor	Risk Factor	Problem
6	Large Amount of New Functionality Development	4, 5	9	Project Scope Definition	Multiple functional applications that should be developed within the scheduled delivery
7	Large Amount of Change Request Development	4, 5	9	Lack of Communication Between Team and Client; Requirement Conflict	Multiple change request applications that should be developed within the scheduled delivery
8	Large Amount of Fixe Development	2	9, 12, 21	Requirement Conflict; Reorganisation of Project Team;	There is a need to solve more bugs than planned within the scheduled or additional delivery
9	Delivery	5, 6, 7	11, 12	Frequent Deliveries	Scheduled and multiple additional deliveries within the week
10	Project Team	–	14	Reorganisation of Project Team	Staff turnover
11	Short Development Time	9	15, 16	Frequent Deliveries	Lack of time to carry out the development
12	Large Amount of Development	9, 8	16, 21	Frequent Deliveries	There is a need to solve more than planned within the delivery
13	Environmental Gaps	–	16, 18	Inappropriate Tool Selection	Supplier’s environments differ from the customer's affecting application software, database and server operation
14	Different Levels of Expertise for Project Team Members	10	19	Growth in Team Size	Staff turnover as well as the knowledge of project team members
15	System Functionality Not Verified	11	20, 21	Project Management and Controlling	There is no time for unit tests
16	Code Integration	11, 12, 13	20	Code Integration	The functionality is disrupted for various reasons, such as time, amount of development or technical issues
17	Test Data	–	22, 24	Test Data Management	There is not a full range of test data
18	Standard Software Vulnerability	13	23	Inappropriate Tool Selection	Errors increase the likelihood that the system will have security holes
19	Mistakes Made Using Semi-automated Delivery	14	23, 25	Inappropriate Tool Selection; Reorganisation of Project Team	Errors resulting from manual operations by the developer
20	Negative Impact on System Functionality	15, 16	24	Project Management and Controlling; Code Integration	Developer does not verify functionality before and after integration
21	Large Amount of Testing	8, 12, 15	24	Frequent Deliveries	There is no time for full system check-up
22	Test Scenario	5, 17	24	Inappropriate Test Scenario; Requirement Conflict	Uncertain requirements affect the quality of test scenario accuracy
23	Release Management and Deployment	18, 19	25	Difficulty in Release Management and Deployment	Not fully developed system parts or incorrectly assembled delivery affects the version and does not meet the expectations
24	Undetected Bugs	17, 20, 21, 22	25	Test Data Management; Code Integration; Inappropriate Test Scenarios	Customer creates new error requests after the delivery about already supplied functionality
25	Unsuccessful Delivery	19, 23, 24	–	Project Management and Controlling	If the number of returned requests is greater than 50 % of the delivery, it is classified as unsuccessful

B. The Construction and Validation of BN

The software delivery model is constructed on theoretical restrictions mentioned in Section III, the software delivery journal containing delivery data for the last six months (see Table I) and risk factor analysis explaining the actual situation of the project (see Table II). In addition, an important step is also to make a quantitative risk analysis. In our case, we use probability matrices for each node of the network taking into account experience and daily observations on the occurrence of risk factors. Fig. 3 represents the constructed network describing software delivery. The formal description of BN results in the probability of each event, taking into account the direct descendants. Bayes Server[1] software is used to evaluate the risk model. After determining probabilities of all events, the probability of unsuccessful delivery P_u was evaluated as P_u = 64.94 % (i.e., project success probability was 35.06 %). To validate the evaluation results, P_u was compared to data recorded in the delivery journal (see Table I).

[1] Bayes Server is an analytics software covering various fields, such as Machine Learning, Data Science, Artificial Intelligence and Time Series Analysis. It is used to make predictions, extract insight, and perform decision support using models built from data and expert opinion.

Additionally, the number of requests returned by the client versus the total number of requests was compared on a weekly basis. Fig. 4 represents the total number of supplier's requests and number of successfully and unsuccessfully delivered requests. Fig. 5 shows unsuccessfully delivered requests as a percentage of number of requests delivered. The percentage of unsuccessfully delivered requests over 27 weeks is 65.35 %. That indicates that the actual percentage of the unsuccessful deliveries is close to the estimated probability P_u and the risk model can be perceived as adequate for the given problem situation.

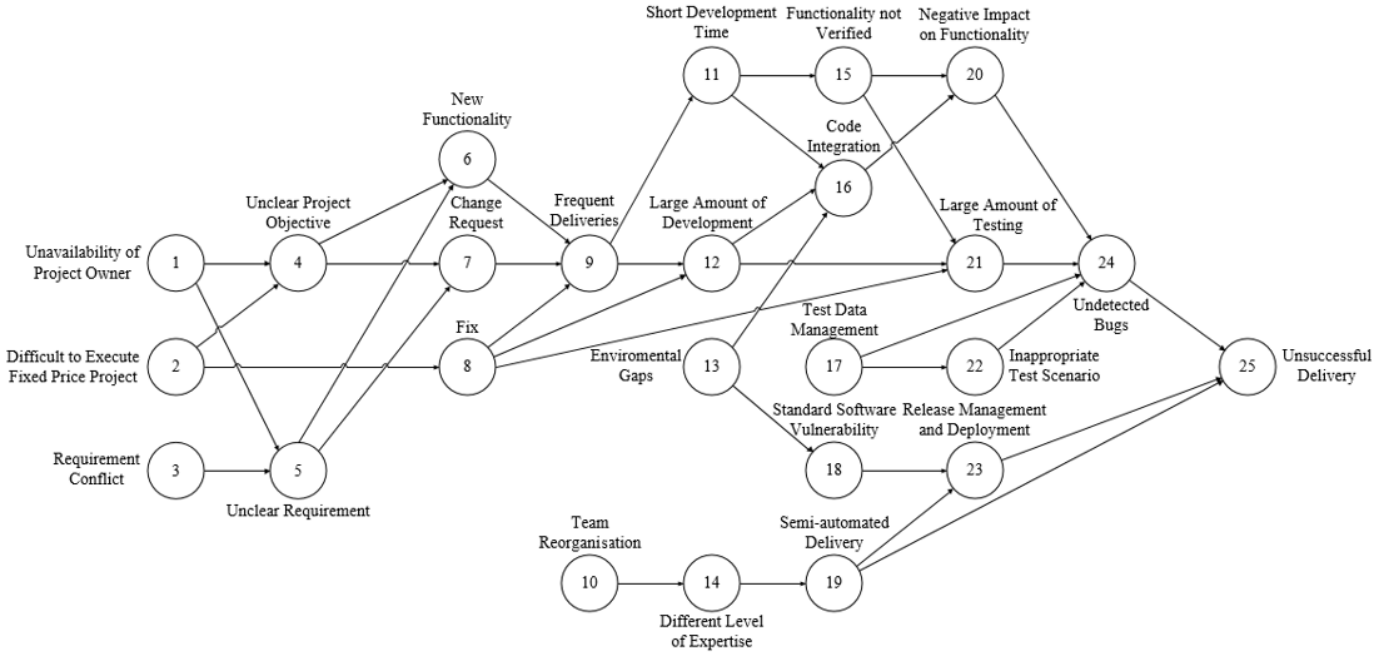

Fig. 3. Risk model of software delivery.

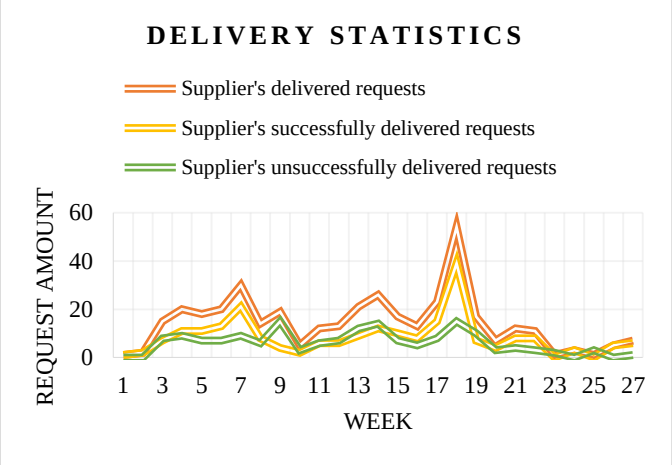

Fig. 4. Delivery statistics.

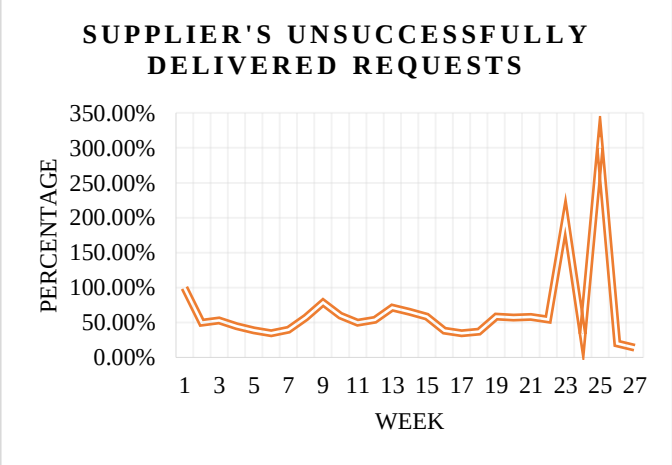

Fig. 5. Unsuccessfully delivered applications.

C. Risk Analysis

To investigate how to reduce the probability of P_u the data of 1st week of August are used as a basis, where three fixes, one change request and five new system functionalities are delivered. To limit the possibility of unsuccessful delivery, there is a need to evaluate the controllable risks of delivery process and the possible prevention methods, which would allow managing software delivery more effectively. By researching the problem situation, one of the risks that can be controlled is unavailability of project owner with the occurrence of 37.56 %. To reduce the influence of this risk on P_u actions that can be done is encouraging Project Owner (further – PO) to be available for team interactions and surrogating PO to interact with PO on the client side [14]. The requirement conflict risk with the occurrence probability of 37.92 % can also be controlled. To control it, such actions as facilitating group discussions, showing demonstrations to the client, recording the minutes of meetings with the customer, performing user acceptance tests after each build and team members visiting end-user location can be done [14]. Software delivery risk can also be reduced by test Data Management control, the occurrence probability of which is 60 %. This risk

has a direct impact on inappropriate test scenario regulation – 53.43 %. Thus, uncertain requirements affect the quality of test scenario accuracy. In cases when full amount of test data is not available, it is needed to automate test data management by using appropriate tools [14]. Test scenario quality can be improved by also having onsite a representative from the vendor at the client location and using exploratory testing as it needs less documentation [14]. In turn, the occurrence probability of reorganisation of project team risk (initially 80.33 %) can be reduced by decreasing employee rotation. It can be controlled with majority of team members being core members and assigning a mentor to new team members to speed them up [14]. Such a solution is also suitable for controlling different levels of expertise risk, thereby reducing the 100 % occurrence probability. In case of risk occurrence there is a need to involve additional human resources, to encourage growth in team size, which is one of the success factors of agile. It determines that relationship is the experience level of the project team. It is recognised that project team members with greater experience and background in project-based work are more adept at completing their assignments, working together collaboratively, and performing tasks efficiently [6]. The most suitable solutions to release management and deployment risk (36 %) reduction are frequent and periodic release dates, keeping infrastructure for component integration ready, keeping functionality flexible if dates are constrained [14]. Standard Software Vulnerability (52 %) and Environmental Gaps (40 %) are controllable risks, but full prevention of these is only possible in the project planning phase. The team should choose the SW and HW tools after careful tool study and refer to existing guidelines for tool selection for engineering activities [14]. During the delivery in August, there was an enhanced control of semi-automated delivery risk by developing guidelines for delivery installation process and enhancing communication with involved parties in organisation communication channels. After executing the experiment, it was recognised that P_u percentage can be reduced to 58.85 % (see Fig. 6). Additionally, it is possible to mathematically calculate the effect of this risk when it occurs. The calculations show that the control of this risk has reduced P_u occurrence by 11.31 %.

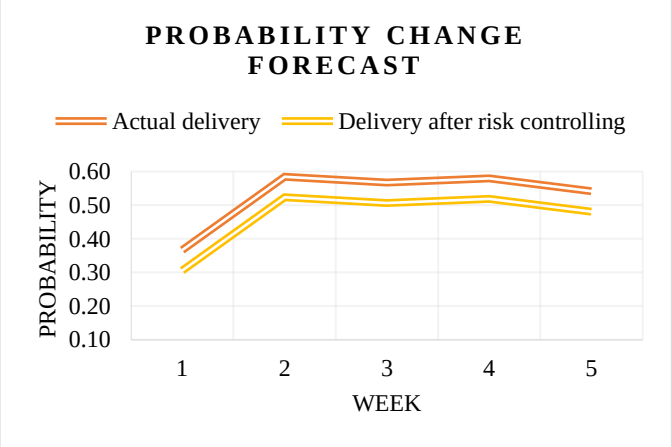

Fig. 6. Probability change after forecast deliveries.

VI. Conclusion

Since the success of any project lies in its management and control, project main objectives can be achieved following the instructions. Although usually a risk analysis is not carried out in agile projects, it may have a significant impact if the project is complex, large-scale and faces numerous problems during the development.

In the present research, we have used a Bayesian Network concept to display an agile project software delivery process in order to seek out whether such an analysis accompanied by suitable actions ensures and increases the service quality. The results have shown that it is possible to limit the failure percentage by monitoring the controllable risks and their impact factors. In addition, direct ancestor controlling constitutes the highest improvement of event under scrutiny.

This kind of risk factor management provides a complete and accurate representation of the actual situation in the project, especially when there is a clear understanding of the key problems associated with risks and their dependencies. Accompanied by statistical and quantitative data, the model is usable for various kinds of experiments related to investigation and forecast purposes. It ensures demonstrative representation of main risk factors and can assist in the decision-making process.

Practicality of this kind of modelling in agile projects is still a moot point. Despite the fact that the graph design process is relatively simple, conditional probabilities are hard to get. It takes time of daily observations and experience for correct probability determination. A knowledge management system, guidelines and a model verification analysis could assist for ensuring the accuracy of data entered. Nonetheless, such models can be used repeatedly describing processes that are standardised, implying that they can be reused within the organisation for projects to come.

References

[1] M. Li, M. Huang, F. Shu and J. Li, "A risk-driven method for eXtreme programming release planning," in *Proc. of the 28th int. conf. on Software engineering*, Shanghai, China, 2006, pp. 423–430. http://dx.doi.org/10.1145/1134285.1134344

[2] A. Nagy, M. Njima and L. Mkrtchyan, "A Bayesian Based Method for Agile Software Development Release Planning and Project Health Monitoring," in *2nd Int. Conf. Intelligent Networking and Collaborative Systems (INCOS)*, Nov. 24–26, 2010, Thessaloniki, Greece. IEEE, 2010. http://dx.doi.org/10.1109/incos.2010.99

[3] S. W. Ambler and M. Lines, *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*, Indianapolis: IBM Press, 2012.

[4] M. Perukusich, G. Soares, H. Almeida and A. Perkusich, "A procedure to detect problems of processes in software development projects using Bayesian networks," *Expert Systems with Applications*, vol. 42, pp. 437–450, Jan. 2015. http://dx.doi.org/10.1016/j.eswa.2014.08.015

[5] S. Mohanarajah and M. A. Jabar, "An Improved Adaptive and Dynamic Hybrid Agile Methodology to Enhance Software Project Success Deliveries," *Journal of Theoretical & Applied Information Technology*, vol. 75, pp. 301–325, May 2015.

[6] P. Serrador and J. K. Pinto, "Does Agile work? – A quantitative analysis of agile project success," *International Journal of Project Management*, vol. 33, pp. 1040–1051, July 2015. http://dx.doi.org/10.1016/j.ijproman.2015.01.006

[7] M. Perukusich, H. Almeida and A. Perkusich, "A model to detect problems on scrum-based software development projects," in *Proceedings of the 28th Annual ACM Symposium on Applied*

Computing, Coimbra, Portugal, 2013, pp. 1037–1042. http://dx.doi.org/10.1145/2480362.2480560

[8] H. Xiaocong and K. Ling, "A Risk Management Decision Support System for Project Management Based on Bayesian Network," *Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference*, April 16–18, 2010, Chengdu, China. IEEE, 2010. http://dx.doi.org/10.1109/ICIME.2010.5478061

[9] X. Bo, L. Xie-lin and Z. Li-guo, "Accident Management Reinforcing with Qualitative and Quantitative Analysis," in *2013 International Conference on Management Science & Engineering (20th)*, July 17–19, 2013, Harbin, P.R.China. IEEE, 2013. http://dx.doi.org/10.1109/ICMSE.2013.6586589

[10] S. Wagner, "A Bayesian network approach to assess and predict software quality using activity-based quality models," *Information and Software Technology*, vol. 52, pp. 1230–1241, Nov. 2010. http://dx.doi.org/10.1016/j.infsof.2010.03.016

[11] J. Dhlamini, I. Nhamu and A. Kachepa, "Intelligent risk management tools for software development," in *Proceedings of the 2009 Annual Conference of the Southern African Computer Lecturers' Association: SACLA '09*, Eastern Cape, South Africa, 2009, pp. 33–40. http://dx.doi.org/10.1145/1562741.1562745

[12] Z. Zhang, G. Rao, J.Cao and L. Zhang, "Software Process Risk Measurement Model based on Bayesian Network," in *2014 5th `IEEE International Conference Software Engineering and Service Science* (ICSESS), June 27–29, 2014, Beijing, China. IEEE, 2014. http://dx.doi.org/10.1109/ICSESS.2014.6933510

[13] I. Khuankrue and W. Rivepiboon, "Model of cross-culture risk prediction base on Bayesian belief networks for software project," in *2012 International Conference on Innovation Management and Technology Research (ICIMTR2012)*, Malacca, Malaysia, 2012, pp. 560–565. http://dx.doi.org/10.1109/ICIMTR.2012.6236458

[14] S. V. Shrivastava and U. Rathod, "Categorization of risk factors for distributed agile projects," *Information and Software Technology*, vol. 58, pp. 373–387, Feb. 2015. http://dx.doi.org/10.1016/j.infsof.2014.07.007

[15] V. G. Venkatesh, S. Rathi and S. Patwa, "Analysis on supply chain risks in Indian apparel retail chains and proposal of risk prioritization model using Interpretive structural modeling," *Journal of Retailing and Consumer Services*, vol. 26, pp. 153–167, Sept. 2015. http://dx.doi.org/10.1016/j.jretconser.2015.06.001

Ieva Ancveire obtained her Bachelor Degree in Computer Science at Riga Technical University, Latvia, in 2013. She works as a Systems Analyst at Autentica, Ltd. Her areas of interest are system analysis and project management.
E-mail: ieva.ancveire@edu.rtu.lv

Ilze Gailite holds a *B. Sc.* (2013) degree in Information Technology from Riga Technical University, Latvia. Since 2015 she has been working as a Junior Software Tester / Systems Analyst at Autentica, Ltd. Her areas of interest are project management, IS testing, test management and system analysis.
E-mail: ilze.gailite@edu.rtu.lv

Made Gailite obtained her Bachelor Degree in Computer Management and Computer Science from Riga Technical University, Latvia, in 2013. Since August 2015, she has been working as a Software Engineering Associate at Accenture Latvia. Her main areas of interests are enterprise resource planning, SAP, software development, data analysis and cloud solutions.
E-mail: made.gailite@edu.rtu.lv

Janis Grabis holds a Doctoral Degree and is a Professor at Riga Technical University (Latvia) and the Director of the Institute of Information Technology. His main research interests lie within the application of mathematical programming methods in information technology, enterprise applications and system integration. He has published more than 80 scientific papers, including a monograph on supply chain configuration. He has led a number of international and national projects and has participated in five projects in collaboration with the University of Michigan-Dearborn (USA) and funded mainly by industrial partners, such as SAP America and Ford Motor Company.
E-mail: grabis@rtu.lv