A Method to Build Bayesian Networks based on Artifacts and Metrics to Assess Agile Projects

Renan Willamy*[1], João Nunes†[1], Mirko Perkusich‡[1], Arthur Freire§[2], Renata Saraiva¶[2], Hyggo Almeida‖[2], and Angelo Perkusich**[2]

[1]Paraíba Federal Institute of Education, Science and Technology, Monteiro, Brazil
[2]Embedded and Pervasive Computing Laboratory, Federal University of Campina Grande, Campina Grande, Brazil

Abstract

Managing software development projects is a complex task because it requires organizing and monitoring several activities. Recently, in order to assist on software projects management, artifact-based models were proposed in the literature. However, the current solutions do not present means to monitor projects health and assist on decisions making. Due to the recent popularization of agile methods, they are the units of study of this research. In this work, we present a method to build artifact and measurement-based models to assess agile projects health. We applied the method to build a generic model based on industrys best practice. We defined the models artifacts and metrics based on findings of a literature review and the assistance of an expert. For each models artifact, we applied the Goal-Question-Metric paradigm to define the metrics. Afterwards, from the GQM meta-model, we constructed a Bayesian network. We validated the model with simulated scenarios. Given the successful results, we concluded that the method and model are promising.

Agile Methods; Bayesian Network; Artifact; Metric; Goal-Question-Metric

*renanwillamy2@gmail.com
†lockenunes@gmail.com
‡mirko.perkusich@ifpb.edu.br
§arthur.freire@embedded.ufcg.edu.br
¶renata.saraiva@embedded.ufcg.edu.br
‖hyggo@embedded.ufcg.edu.br
**perkusic@embedded.ufcg.edu.br

1 Introduction

Managing software development projects is a complex task, because it is necessary to organize and monitor many activities such as requirements, architecture and project definition, coding, testing and implementation of the product. According to Emam and Koru [7], between 46% and 55% of IT projects fail. According to Boehm et al. [4], the top six reasons for this high failure rate are: incomplete requirements, lack of user involvement, lack of resources,unrealistic expectations, lack of executive support and changing requirements and specifications. Most of these reasons are caused by communication and interaction issues between developers and stakeholders.

Recently, agile methods have become largely adopted by software development industry [27]. Agile methods are focused on the improvement of collaboration between developers and stakeholders in order to improve effectiveness when requirements change.

Managing agile projects is a big challenge, specially in organizations with traditional culture [21]. Besides being flexible to changing requirements and adopting iterative and incremental development, the adoption of agile methods is complex given the context of architecture definition, requirements prioritization, and quality assurance. According to VersionOne [26], 44% of respondents pointed to lack of experience with agile methods as the main cause for agile to fail; 42% to company philosophy or culture at odds with core agile values. As barriers to agile adoption, 44% pointed to the ability to change organizational culture, 35% to not enough personnel with the necessary agile experience, 32% to pre-existing waterfall framework. Therefore, there is a need for resources to assist on the adoption and

DOI reference number: 10.18293/SEKE2016-213

continuous improvement of agile methods.

According to Briand et al. [5], it is important to assess a software development process model to define the study objects of a measuring program, and, consequently, the necessary metrics to be collected. In the literature, artifact-based models are proposed to assist on the managing of projects, specially, distributed teams-based projects. Artifacts represent the products related to the software development project such as source code, documentation, specifications e architectural models.

Kuhrmann et al. [16] presented a generic artifact-based model for distributed agile software development teams. Despite the fact that this model is a result of an evolution for centralizing the management of agile distributed teams, some information regarding the project's health, which help on the decision making process, are missing. In our context project health is the current status of the project given its goals (e.g., scope and time restrictions).

These missing information can be collected through software metrics. These metrics allow the measurement, assessment, control and improvement of software artifacts [11]. As a result, it is possible to define a meta-model connecting the artifacts as well as their respective metrics. However, only a meta-model is not enough to assess the artifact according to the collected data.

Recently, Bayesian Networks have become a popular as a mechanism for constructing executable models that are able to deal with uncertainty and assist on the decision making process. It has been applied in several context of software engineering such as risk management [8, 9, 15, 13], quality prediction [1, 14] and process management [19].

In this paper, we aim to complement the work presented by Kuhrmann et al. [16] by presenting a method for constructing artifact-based probabilistic models to assist on the management of agile projects. With the help of a specialist, a generic model (i.e., Bayesian Network) for agile projects was constructed, addressing the main agile projects artifacts and their respective metrics. To define the metrics, we applied the Goal-Question-Metric (GQM) paradigm [2]. Given that the model is a Bayesian Network, its structure facilitates possible modifications to address other artifacts and metrics.

The method and model presented in this paper are limited to the context of projects composed by only one team. To validate them, we used ten simulated scenarios. Based on the results, we concluded that both method and model are promising.

This paper is organized as follows. Section 2 presents an overview of the usage of Bayesian networks in software engineering context. Section 3 presents the method developed to build Bayesian networks based on artifacts and metrics to assess the health of the project. Section 4 presents the generic model build for agile projects. Section 5 the results of the validation Section 6 presents our conclusions, limitations and future works.

2 Overview of Bayesian Networks applied to Software Engineering

Bayesian networks have been applied to many areas in software engineering such as risk management and product quality prediction. Fan and Yu [8], Fenton et al. [9] and Hu et al. [13] modeled software processes to support risk management. Fan and Yu [8] built a model capable of predicting potential risks, identifying the source of risks, and supporting dynamic resource adjustment. [9] showed how to use a Bayesian network to predict software defects and perform "what if" scenarios. Jeet et al.[15] built a model to estimate the impact of low productivity on the schedule of a software development. They used interviews and historical data to build the model. Hu et al. [13] proposed a risk identification framework for outsourced software projects.

Abouelela and Benedicenti [1] and Jeet et al. [14] modeled software processes to support quality management. Abouelela and Benedicenti [1] built a model to predict a releases rate of defects in a XP project and its duration. Jeet et al. [14] built a model to detect the number of defects in a software development project. The rate of defects and the project manager's judgments are used for predictions and support in managing the number of defects.

Settas et al. [22], Stamelos [23], Stamelos et al. [24] and [19] modeled software processes to support other project management activities. Settas et al. [22] and Stamelos [23] used Bayesian networks to help managerial decision making by modeling software project management anti-patterns. Stamelos et al. [24] modeled the uncertainties of factors to estimate software productivity. Perkusich et al. [19] modeled software processes to support continuous improvement.

Hearty et al. [12] and Perkusich et al. [18] applied Bayesian networks to agile context. Hearty et al. [12] used a Learning Dynamic Bayesian network model to predict project velocity in XP. Perkusich et al. [18] modeled the processes of Scrum projects to detect problems in the team and processes. None of them used artifacts as a base to build the Bayesian networks.

3 The Method

The goal of the method is to assist on the construction of a Bayesian Network-model based on artifacts and metrics to assess the health of agile projects, which consists of their status given their goals. This method consists of five sequential steps: (i) measurement goals definition; (ii) artifacts definition; (iii) metrics definition; (iv) Bayesian Net-

work construction; and (v) validation. Despite being sequential, there are feedback loops between the steps. In step (i), the measuring goals are defined according to the needs. In general, they are related to a project restriction, such as schedule, budget or scope.

In step (ii), development processes artifacts that are related to the measurement goals are defined. An artifact might be associated to more than one measurement goals, and a measurement goal might be associated to more than one artifact, which is a many-to-many relationship. In this step, it is preferable to choose artifacts that are already used in the development process to avoid additional effort on the measurement process.

To execute step (iii), it is necessary to apply the GQM paradigm. Since an artifact may be associated to more than one measurement goal, it is possible to have more than one instance for a given artifact. For each instance, according to its measurement goal, it is necessary to define the goal (i.e., level G), a set of questions (i.e., level Q) for this goal, and metrics (i.e., level M) to be collected in order to measure if the goal associated to the artifact is reached.

In step (iv), the meta-model constructed in step (iii) is transformed in a Bayesian Network. To define the DAG (Directed Acyclic Graph), it is necessary to define a node for each artifact. Afterwards, these nodes need to be decomposed until the metrics associated to them are reached. This can be done by following some steps:

1. For each managing area, define a node;
2. For each goal, define a node and connect it to one or more management areas;
3. For each relationship, an edge is added, in which the edge's endpoint points to the management area;
4. For each artifact, a node is defined and, then, connected to one or more goals;
5. For each relationship, an edge is added, in which the edge's endpoint points to the goal;
6. For each question, define a node and connect it to an artifact. For each relationship, an edge is added, in which the edge's endpoint points to the artifact;
7. For each metric, define a node and connects it to one or more questions;
8. For each metric, an edge is added, in which the edge's endpoint points to the question.

After transforming the GQM meta-model into the DAG, it is necessary to define the probability functions by performing "what if" analysis.

In step (v), it is necessary to validate the Bayesian Network. For this purpose, it is possible to use the Brier score [10] and simulated scenarios [18].

4 Generic Model

To build the generic model, we used the artifacts presented in Kuhrmann [16] and elicited knowledge from an expert. In Section 5, we present the results and the profile of the expert. Moreover, we executed the method presented in Section 3.

The goal of this generic model is to represent an agile project, based on agile best practices, to measure its health. Given that the projects run in different contexts, there might be necessary to modify the model to apply it. Usually, In the context of software process modeling, organizations need to adapt methods and models, so they fit their contexts. However, we believe the generic model presented is robust enough to guarantee minimum modification.

The first step for constructing the model is to define the goals of its measuring. We used the Agile Manifesto [3] to define the measuring goals. In addition to that, we tried to categorized the principles of the Agile Manifest based on projects restrictions: scope, quality, schedule, and cost. We introduce some agile principle as follows:

1. The highest priority is to satisfy the customer through early and continuous delivery of valuable software;
2. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale;
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale;
4. Continuous attention to technical excellence and good design enhances agility.

By analysing these principles, it is possible to classify 1, 2, and 4 as scope principles; and 3 as a schedule principle. We considered principles related to product quality connected to the scope principles. We did not consider metrics for cost, because they are not directly related to any agile principle.

Afterwards, the artifacts were defined. As scope artifacts, we considered the *Product Backlog*, the *Product Increment*, and the *Tests Report*. Since the model is limited to the project level and a product can be delivered through more than one project (i.e., releases), we also decided to use the *Release Backlog*, which is a subset of the *Product Backlog*. As schedule artifact, we only considered the *Release Burndown*. For each artifact, the GQM template was applied to describe its the measuring goal.

- G1: Analyse the *Release Backlog* for characterizing it regarding the product quality, from the viewpoint of the PO;

- G2: Analyse the *Product Increment* for characterizing it regarding the product satisfaction, from the viewpoint of the PO;
- G3: Analyse the *Tests Report* for characterizing it regarding the tests completeness, from the viewpoint of the PO;
- G4: Analyse the *Release Burndown* for characterizing it regarding the product progress, from the viewpoint of the PO.

Then, for each artifact, we defined a set of questions and metrics. Due to space limitations, we only present the questions for the *Release Backlog*. To define the questions for this artifact, the expert recommended to use the *DEEP* criteria [20]. Given this, we defined four questions: "Is it correctly detailed?", "Is it correctly estimated?", "Does it evolve correctly?", and "Is it correctly ordered?".

For each question, we derived one metric: *item detail level*, *estimation conformity*, and *dynamics of the estimation*. Since these metrics are all subjective, we decided to represent them in 5-point Likert scale. To collect them, it is necessary to answer, respectively, the following questions: "Do you agree that the next iteration's items satisfy the *INVEST* criteria [6]?"; "Are the next iteration's items small enough so they be done in one iteration?"; and "Does the backlog evolve (i.e., new items added, existing items refined, deleted or reordered) continuously according to the product feedback and changes in the business scenario?"

Since the space is limited, we will only present the assessment method for the metric *item detail level*. Each item allocated for the next iteration needs to be assessed according to the *INVEST* criteria. This criteria assess six factors: (i) independence, (ii) negotiability, (iii) value, (iv) if it is estimable (v) size, and (iv) testability. It i's ideal that all items are independent; their scope had been already discussed between the development team and the client; they have business value; the developers have enough knowledge to estimate them; and they're small enough so they can be done in one iteration, as well as tested considering non-functional requirements (e.g., performance, usability, and security). For this purpose, it is necessary to assess each backlog item according to these criteria in a 5-point scale (i.e., "1" for Very Low, and "5" for Very High). Afterwards, the results need to be aggregated and mapped into the *item detail level* metric. For doing so, we used the WMIN function.

Regarding the question "Is it correctly ordered?", the metrics were defined according to the criteria *DIVE* [20]. These metrics are: *backlog items dependency*, *risks*, *estimation conformity*, and *business value*. Since these metrics are all subjective, we decided to represent them in 5-point Likert scale. To collect these metrics, it is necessary to answer, respectively, the following questions: "Do you consider the backlogs' items technical dependency?", "Do you consider the backlogs' items technical and business risks?", "Are next iteration's items estimated with points or ideal days?", and "Do you consider the business value, in the contexts of the user, schedule, and organization?" For the *business value* metric, it is possible to use the numeric methods presented by Thatte [25]. However, the assessment of their usage wasn't addressed in this study. After defining the set of questions and metrics for each artifact identified, we built a GQM meta-model, which was later transformed into the DAG. We present the DAG in Figure 1.

To calibrate the probability functions, we used the knowledge of one expert, which has over five years of experience as an agile project manager. For each child-node, we defined a questionnaire composed by a set of combinations of its parent nodes. Due to space limitations, we limit ourselves to state that, to generate the probability functions, we applied the approach proposed by Laitila [17].

5 Validation

We validated the model and the procedure in ten simulated scenarios, defined according to the expert availability. To due space limitations, we only present the results of one scenario. This scenario (i) describes an experienced and efficient development team, in which the manager is ineffective and absent.

For each scenario, according to their contexts, the values for the input nodes (i.e., nodes with no parents) were defined by the specialist. Furthermore, for the nodes related to the goals and artifacts, the specialist defined the expected values. For this scenario, the expected results were:

- Scope: central tendency between *Very Low* and *Low*.
- Time: central tendency between *Very Low* and *Low*.
- G1: central tendency between *Very Low* and *Low*.
- G2: central tendency between *Medium* and *Low*.
- G3: central tendency between *Medium* and *Low*.

We present the results for this scenario in Table 1. These results match the expected results we defined before starting the validation process. Given this, we concluded both method and generic model are promising. However, the number of scenarios we defined for validating the method and model are not enough to conclude they would be useful in the context of real projects.

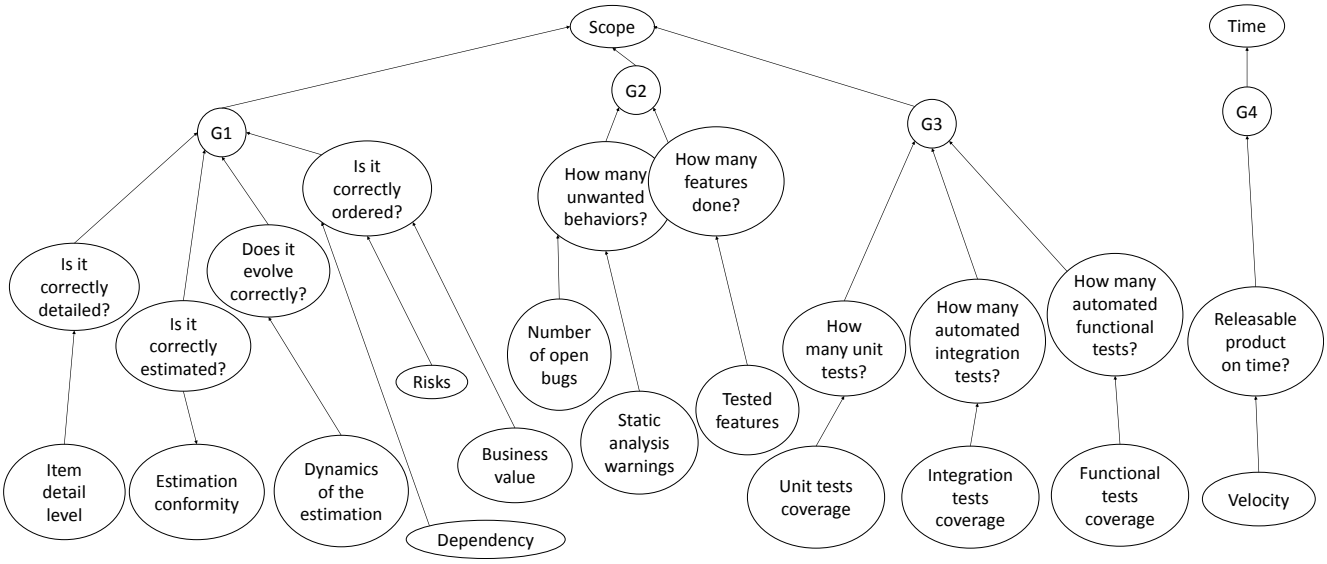

Figure 1. Complete directed acyclic graph for the Bayesian network.

	States				
Node	VL	L	M	H	VH
Scope	56%	44%	0	0	0
Time	0	100%	0	0	0
G1	100%	0	0	0	0
G2	0	61%	39%	0	0
G3	0	70%	30%	0	0

Table 1. Scenario results for highest level nodes in the model

6 Conclusions

In this paper, we presented a method to construct Bayesian networks based in artifacts and metrics to assist in agile project management. The method consists of five steps: (i) measurement goals definition; (ii) artifacts definition; (iii) metrics definition; (iv) Bayesian Network construction; and (v) validation. The purpose of the method is to measure agile projects' health.

The method was applied to construct a generic model considering the most popular artifacts and metrics in the industry. The model's measurement goals were defined according to agile principles, and it was constructed with the help of an expert. We validated the model through two simulated scenarios with positive results. Thus, we concluded that the results are promising.

This study limitations are related to the quantity of simulated scenarios in which we performed the validation. In addition to that, we only collected knowledge from one expert. However, both limitations are due to the unavailability of industry experts to collaborate in the study during the required period.

For future works, we pretend to perform a systematic review to identify metrics, consult other specialists for constructing the model, perform empirical case studies to evaluate the method and the models constructed in industry context, and extend the method to consider projects with multiple distributed teams.

Acknowledgments

This work was financially supported by Paraba Federal Institute of Education, Science and Technology, Campus Monteiro.

References

[1] M. Abouelela and L. Benedicenti. Bayesian network based xp process modelling. *International Journal of Software Engineering and Applications*, 1(3):1–15, 2010.

[2] V. R. Basili and D. M. Weiss. A methodology for collecting valid software engineering data. *IEEE Transactions on Software Engineering*, SE-10(6):728–738, Nov 1984.

[3] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas. Manifesto for agile software development, 2001. Accessed: 29th February 2016.

[4] B. Boehm. Software engineering is a value-based contact sport. *IEEE Softw.*, 19(5):95–96, Sept. 2002.

[5] L. C. Briand, C. M. Differding, and H. D. Rombach. Practical guidelines for measurement-based process improvement. *Software Process: Improvement and Practice*, 2(4):253–280, 1996.

[6] M. Cohn. *Succeeding with Agile: Software Development Using Scrum*. Addison-Wesley Professional, 1st edition, 2009.

[7] K. E. Emam and A. G. Koru. A replicated survey of it software project failures. *IEEE Software*, 25(5):84–90, Sept 2008.

[8] C.-F. Fan and Y.-C. Yu. Bbn-based software project risk management. *Journal of Systems and Software*, 73(2):193–203, Oct. 2004.

[9] N. Fenton, P. Krause, and M. Neil. Software measurement: uncertainty and causal modeling. *Software, IEEE*, 19(4):116–122, 2002.

[10] N. Fenton and M. Neil. *Risk assessment and decision analysis with Bayesian networks*. CRC Press, 2012.

[11] K. A. M. Ferreira, M. A. S. Bigonha, R. S. Bigonha, L. F. O. Mendes, and H. C. Almeida. Identifying thresholds for object-oriented software metrics. *J. Syst. Softw.*, 85(2):244–257, Feb. 2012.

[12] P. Hearty, N. Fenton, D. Marquez, and M. Neil. Predicting project velocity in xp using a learning dynamic bayesian network model. *IEEE Transactions on Software Engineering*, 35(1):124–137, Jan 2009.

[13] Y. Hu, X. Mo, X. Zhang, Y. Zeng, J. Du, and K. Xie. Intelligent analysis model for outsourced software project risk using constraint-based bayesian network. *Journal of Systems and Software*, 7(2):440 – 449, 2012.

[14] K. Jeet, N. Bhatia, and R. S. Minhas. A bayesian network based approach for software defects prediction. *SIGSOFT Softw. Eng. Notes*, 36(4):1–5, Aug. 2011.

[15] K. Jeet, N. Bhatia, and R. S. Minhas. A model for estimating the impact of low productivity on the schedule of a software development project. *SIGSOFT Softw. Eng. Notes*, 36(4):1–6, Aug. 2011.

[16] M. Kuhrmann, D. M. Fernández, and M. Gröber. Towards artifact models as process interfaces in distributed software projects. In *Proceedings of the 2013 IEEE 8th International Conference on Global Software Engineering*, ICGSE '13, pages 11–20, Washington, DC, USA, 2013. IEEE Computer Society.

[17] P. Laitila. Improving the use of ranked nodes in the elicitation of conditional probabilities for bayesian networks. Master's thesis, Aalto University, Finland, 2013.

[18] M. Perkusich, H. O. de 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*, pages 1037–1042. ACM, 2013.

[19] M. Perkusich, 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*, 42(1):437 – 450, 2015.

[20] R. Pichler. *Agile Product Management with Scrum: Creating Products that Customers Love (Adobe Reader)*. Addison-Wesley Professional, 2010.

[21] K. Schwaber and J. Sutherland. Guia do scrum. `http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf`, 2015. Accessed: 17th March 2016.

[22] D. Settas, S. Bibi, P. Sfetsos, I. Stamelos, and V. Gerogiannis. Using bayesian belief networks to model software project management antipatterns. In *Software Engineering Research, Management and Applications, 2006. Fourth International Conference on*, pages 117–124, 2006.

[23] I. Stamelos. Software project management antipatterns. *Journal of Systems and Software*, 83(1):52 – 59, 2010.

[24] I. Stamelos, L. Angelis, P. Dimou, and E. Sakellaris. On the use of bayesian belief networks for the prediction of software productivity. *Information and Software Technology*, 45(1):51 – 60, 2003.

[25] S. Thatte. Agile prioritization: a comprehensive and customizable et simple and practical method. `https://goo.gl/97POHR/`, 2014. Accessed: 17th March 2016.

[26] VersionOne. 9th annual state of agile development survey results. http://www.versionone.com/pdf/state-of-agile-development-survey-ninth.pdf, 2015. Accessed: 2015-05-05.

[27] Y. Zhang and S. Patel. Agile model-driven development in practice. *IEEE Software*, 28(2):84–91, March 2011.