A Framework to Build Bayesian Networks to Assess Scrum-based Software Development Methods

Mirko Perkusich*[1], Kyller Gorgonio†[1], Hyggo Almeida‡[1], and Angelo Perkusich§[1]

[1]Embedded and Pervasive Computing Laboratory, Federal University of Campina Grande, Campina Grande, Brazil

Abstract

Agile software development has been increasingly used to satisfy the need to respond to fast moving market demand and gain market share. Scrum, which is a project management framework, dominates as the most popular agile method. In the literature, there are a number of solutions to customize and assess Scrum-based agile methods, but they are limited to focus only on process factors, assume a predefined set of practices or rely only on subjective evaluation. This paper presents a framework to build a Bayesian Network to assist on the assessment of Scrum-based software development methods. The BN models the main entities of the software development process and can be complemented with software practices and metrics. To evaluate the completeness of our solution, we performed simulations to check if the proposed framework diagnoses 14 known Scrum anti-patterns extracted from the literature. 12 anti-patterns were directly detected, 1 was indirectly detected by the BN and 1 was considered as invalid. We concluded that the proposed solution is complete to detect the major flaws of Scrum-based software development methods and can be used to assist on the configuration, adoption and continuous improvement of Scrum teams.

Agile Methods; Bayesian Network; Software Metrics; Scrum; Method Engineering

*mirko.perkusich@embedded.ufcg.edu.br
†kyller@embedded.ufcg.edu.br
‡hyggo@embedded.ufcg.edu.br
§perkusic@embedded.ufcg.edu.br

1 Introduction

Agile Software Development (ASD) methods have been increasingly used to satisfy the need to respond to fast moving market demands and gain market share. In contrast with traditional plan-driven processes, agile methods focus on people, are communication-oriented, flexible, fast, light, responsive and oriented to learning and continuous improvement [7]. As a consequence, subjective factors such as collaboration, communication and self-organization are key to evaluate the maturity of agile software development.

Scrum dominates as the most popular agile process. It is a software project management framework and must be complemented with technical and managerial practices and processes to define a software development method. If not complemented properly, Scrum might result in reduced productivity and product quality [23, 24].

The method definition depends on customization factors such as team characteristics, internal environment, external environment, project goals, maturity level and previous knowledge [2]. In the context of ASD, most solutions to assist on agile methods configuration are based on Method Engineering [1], in which the team configures a method adapting existing methods (e.g., Rational Unified Process, Scrum and Extreme Programming).

To adopt agile methods, some studies propose combining ASD with CMMI [17]. Others, are specific to ASD [2, 16, 24, 8, 10]. The proposed solutions have one or more of the following limitations: focuses only on the process, assume a predefined set of practices or rely only on subjective data.

Given this, we conclude that there is no consolidated approach to assist on the adoption and continuous improvement of agile methods. In this paper, we present a framework to build a Bayesian Network (BN) to assist on the assessment of Scrum-based software development methods. The BN models the main entities of the software develop-

DOI reference number: 10.18293/SEKE2017-139

ment process and can be complemented with software practices and metrics. With the resulting BN, it is possible to monitor if the agile method is being followed and if it meets the needs of the project to assist on decision-making.

Bayesian networks are probabilistic graph models and used to represent knowledge about an uncertain domain. A Bayesian network, B, is a directed acyclic graph that represents a joint probability distribution over a set of random variables V. The network is defined by the pair $B = \{G, \Theta\}$. G is the directed acyclic graph in which the nodes $X_1, \ldots, X_n$ represent random variables and the arcs represent the direct dependencies between these variables.

The problem of modeling the Scrum process involves causal reasoning and aleatory uncertainty (i.e., intrinsic variability), which can be reduced by gathering more data or eliciting knowledge from experts. Given that the framework must be used by practitioners, the inferences must be clear. Furthermore, the models must be adaptable and assist on decision-making. According to Verbert et al. [20], Bayesian network is a suitable approach for this context.

This paper is organized as follows. Section 2 presents the proposed solution. Section 3 presents the results of the validation; and Section 4 presents our conclusions, limitations and future works.

2 Framework description

The goal of the framework is to build a BN to model a Scrum-based software development method. Scrum Masters should use it to guide the configuration of the agile method and to assess it. As a result, the team has an instrument to support its continuous improvement. As input, the BN receives data collected from the Scrum Master (e.g., through a questionnaire) or automatically through tools. The BN outputs data with probability values that represent the current status of key factors in the software development method. The Scrum team should use the data to, during Sprint Retrospective meetings, diagnose problems and prioritize the areas that must be improved. The data should support discussions regarding action points to be executed to improve the quality of the software development method and, consequently, the project's chances of success. The discussions should be a collaborative activity and involve the stakeholders responsible for decision making such as the Product Owner, Scrum Master and development team. Its usage should be supported by a process such as the one presented in Perkusich et al. [14].

The framework consists of a BN that models the main entities of a software development method based on Scrum. It is composed of two types of fragments: frozen and hot. The frozen fragment is related to official artifacts, resources and practices of Scrum as presented in the Scrum Guide [19]. It should not be modified, unless there are plausible justifications based on domain expert knowledge or historical data. The frozen fragment defines the base BN. The reasoning behind having a frozen fragment is that even though some deviations of the Scrum Guide may be well motivated and reasonable, teams are tempted to adjust Scrum for the company without clearly understanding the consequences of the deviations as evidenced by Eloranta et al. [4]. For instance, not having an ordered Product Backlog might happen in the context of having a Product Owner without authority, a Product Owner that is not part of the Scrum team, lack of feedback loops (e.g. Sprint Review meetings) or an incompetent Product Owner [4]. As a consequence, the team might build wrong features and avoid features which are hard to implement or test, increasing the risk of problems arising in the late stages of development. On the other hand, in the context in which features are fixed and given upfront, it is reasonable to not maintain an ordered backlog. Therefore, given that the modifications on the frozen fragment mean deviation of Scrum recommendation, they must only occur if the consequences are understood and reasonable; and not to adapt Scrum to the company's current *status quo*.

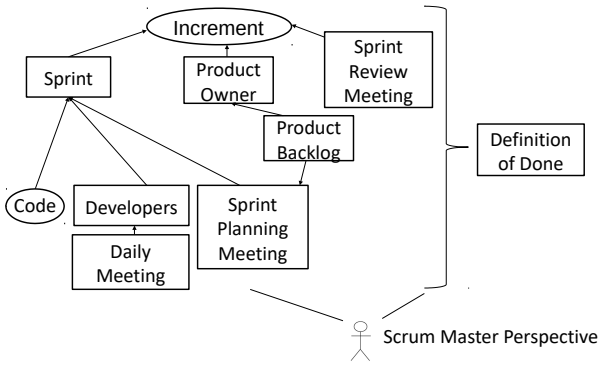

Figure 1. Overview of the framework.

Hot fragments are related with parts of the BN that models practices and processes that should complement the Scrum framework. Furthermore, it is possible to complement the model with fragments representing software metrics-based models (i.e., set of metrics defined for a given purpose). For instance, Quamoco [22], which focuses on measuring product quality, can be associate to the BN. The hot fragments must be defined given the context of the project and must be defined according to the project's Definition of Done.

An overview of the framework is shown in Figure 1, in which the rectangles represent fragments (i.e., set of nodes) and the ellipses represent nodes in the Bayesian network. Notice that the Bayesian network is defined from the viewpoint of the Scrum Master and it must be used to assist during Sprint Retrospective meetings. Therefore, the Scrum Master, a role in Scrum, and the Sprint Retrospective meet-

ing, a Scrum practice, are not represented. Furthermore, the Definition of Done, a Scrum practice, for being an umbrella practice is also not represented.

The problem of building a BN can be divided into: (i) construct the Directed Acyclic Graph (DAG) and (ii) define the Node Probability Tables (NPTs). In Section 2.1, we present details of constructing the DAG of the base BN (i.e. frozen fragment). In Section 2.2, we show the process of complementing the DAG of the base BN with practices and metrics (i.e., hot fragments). The constructed BN is based only on ranked nodes [6], which are based on an ordinal scale. Regarding the definition of the NPTs, this paper is limited to state that collected data from a domain expert and applied the process presented in Fenton et al. [6].

2.1 Base BN construction

The problem of constructing a DAG can be subdivided into: (i) defining the nodes and edges and (ii) defining the scale (i.e., states) of the nodes and the associated probabilities. In the context of our work, a node represents a process entity and there is an edge whenever the entities relate to each other. To solve the second subproblem, we, as other tools to evaluate agile methods such as ComparativeAgility [24], used a five points Likert (i.e., ordinal) scale. Therefore, for each node (i.e., process factor), five possible states were defined. For metrics with numerical scale, thresholds must be defined given the context of the project to map the numerical scale to an ordinal one. The thresholds can be defined using a domain expert knowledge or analysis of historical data.

To solve the first subproblem, we used an incremental approach. Previous versions of the BN were presented in Perkusich et al. [14] and Perkusich et al. [13]. In this paper, we present the most recent version. In comparison with past versions of the base BN, the differences are that (i) we added the concept of frozen and hot fragments, (ii) reduced the size of the base BN from 73 to 44 nodes and (iii) improved the modeling of team (i.e., human) factors.

We followed a top-down approach in which we decomposed the top-level node into factors (i.e., process entities) that we judged to be observable by a Scrum Master or collected through tools. Given that the main goal of agile software development is to satisfy customers with working code and Scrum is an incremental approach, we defined *Increment* as the top-level node. We decomposed it using a well-known DAG idiom: synthesis [5]. Due to space constraints, we only show how we built part of the base BN.

In Scrum, the increment is developed during sprints. The Product Owner is responsible for maximizing the value of the product. Furthermore, the increment must be evaluated during Sprint Review meetings. Therefore, we added the nodes *Sprint*, *Product Owner* and *Review meeting* as parents of the node *Increment*.

Table 1. Examples of practices that can be associated with nodes of the base BN.

Node	Practices
Code	Refactoring, pair programming, Test-Driven Development, Continuous Integration, peer code review and test automation.
Estimation	Planning Poker, big wall, spike [23], story point, ideal hours and T-shirt size.
Product Backlog	Grooming meeting, Release plan [15] and product vision [15].
Ordering	Kano model, Wieger model, OR value, Innovation games, Return of Investment (ROI) and MoSCoW.
Details	User story and Use case
Planning meeting	Sustainable pace [23], Stabilization [23] and Velocity-driven.
Delivery plan	Interaction Room [9].
Monitoring	Burndown and Burnup [19].

During Sprint Review meetings, the product must be inspected and, if necessary, adapted by the customers. As previously explained, not using this practice or not having the customers participating might result in rework and implementation of wrong features. Therefore, we added the nodes *Inspection* and *Adaptation* as parents of the node *Review meeting*. To input data into the model, the Scrum Master must observe the Sprint Review meetings and judge these two factors. As shown in Section 2.2, it is also possible to complement this fragment with metrics to indicate the evidence of these input nodes.

In past evaluations with industry practitioners [12, 14], we noticed that, in practice, it is common for the customer to not be available during all Speint Review meetings. On the other hand, the client evaluated the increment on other informal meetings. Therefore, given that the feedback loop is short and there is a reasonable justification, this is a case in which a frozen spot on the base BN could be modified. An auxiliary node can be added to model this behavior and compensate the lack of participation of the customer during the Sprint Review meetings. The complete DAG for the base BN is shown in Figure 2.

2.2 Complementing the BN with practices and metrics

Given that Scrum must be complemented with practices and processes, the BN can be improved to suit the needs of the given project. Each node in the BN can be mapped to a set of practices. The set of practices is represented as an

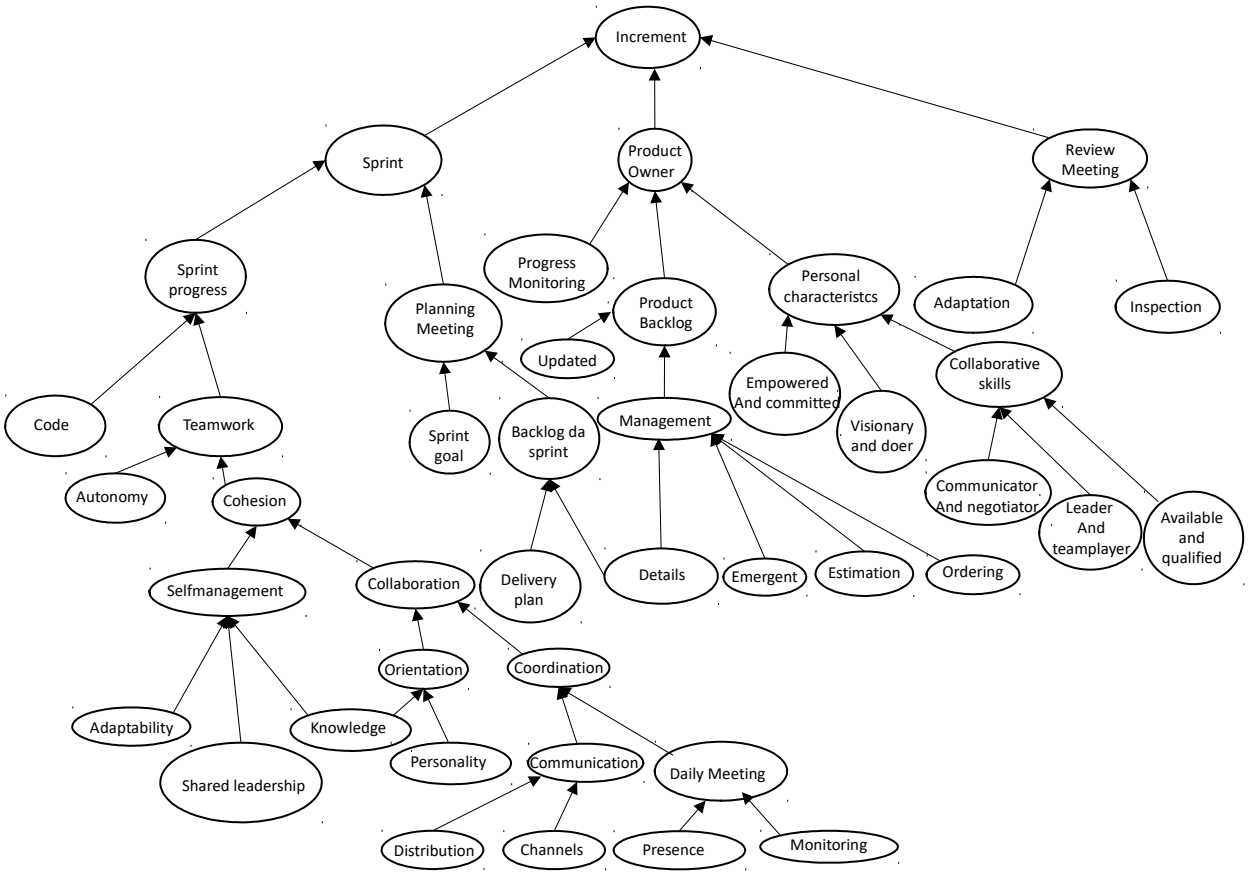

Figure 2. DAG of the base BN.

auxiliary node in the BN. The node is a copy of the given factor. For instance, we can map the node *Code*, to *Node-practices*. Therefore, the evidences on *Code* will indicate the quality of the code production process.

Despite its simplicity, this procedure assists the team on analyzing the BN during the Sprint Retrospective meeting. Given evidences collected during the sprint, the team identifies which processes and practices needs to be modified to increase the chances of building products that satisfy the customers. For this purpose, it is important that the team registers which practices or processes are being used for each factor. In Table 1, we show examples of practices that might be associated with nodes of the base BN.

Another option is to decompose the auxiliary using DAG construction techniques into a causal model representing the given process. Even though it increases the cost of constructing and maintaining the BN, it enables a deeper analysis of the process. For instance, according to Pichler [15], defining a release plan and product vision is a good practice to plan agile projects and influences the quality of the product backlog. The release plan is composed of the release date, description of top features and product vision. The product vision must have a broad and engaging goal, be clear and stable, short and concise and describe the critical attributes to satisfy the needs of the customers. With this information, we can build the causal model shown in Figure 3 and associate it with the node *Product backlog* of the base BN.

Furthermore, it is possible to complement the base BN with metrics. Metrics can be used with two purposes: (i) as indicator of values for input nodes and (ii) warnings. For (i), it is necessary to connect the metrics to the given node and calibrate the NPT. If the metric is automatically collected through tools, this approach can reduce the cost of using the BN. For instance, we could use static analysis metrics to indicate the quality of the code. On the other hand, manually collected metrics can also be added. For instance, to indicate if the Product Backlog is ordered correctly, we can evaluate the technical dependencies, risks and business value [21]. The resulting fragment is shown in Figure 4.

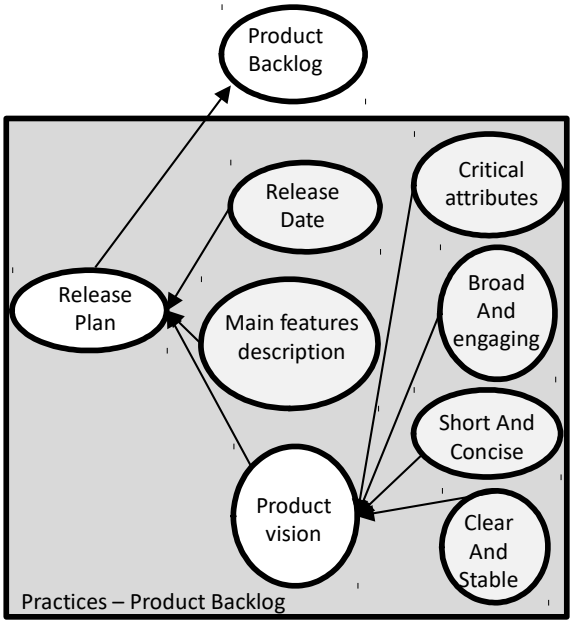

Figure 3. Example of decomposing an auxiliary node.

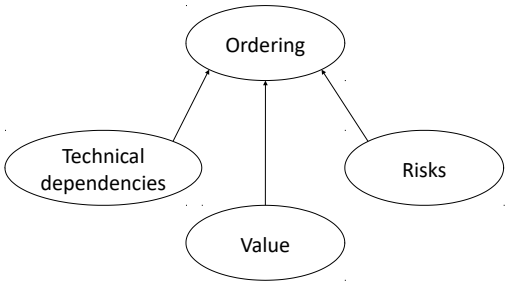

Figure 4. Example of adding metrics to the base BN.

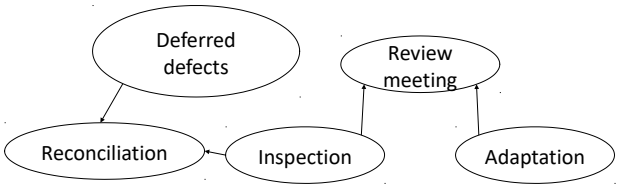

Figure 5. Example of adding warning metric to the base BN.

For (ii), the metric is a warning that indicates that something is wrong with the BN. Furthermore, it helps to identify possible problems in the software development method that the BN failed to identify, which might be a result of problems with the construction, usage or modeling limitation. In this case, the metric should be connected to the base BN using the reconciliation idiom [5].

For instance, we can associate the metric *Deferred defects* [11] to the node *Inspection*. This metric indicates the number of defects that should have been detected during the previous sprint, but were not. Suppose the Scrum Master judges that the Sprint Review meetings for a given project are being executed satisfactorily, but *Deferred defects* were identified. This might raise a warning meaning that increments with defects were approved during Sprint Review meetings, which means that the inspection was not as satisfactorily as judged by the Scrum Master. The resulting fragment for this example is shown in Figure 5. On the other hand, depending on the defects identified, it might not be a problem of inspection during the Sprint Review meeting, but of the testing process. In this case, the metric *Deferred defects* can be associated with the node *Code*. Finally, it might be a false alert and ignored by the team. Therefore, this approach must be used if the team wants to use metrics just as warnings and not as direct input to the BN, which might be the case if there is not much confidence in the causality. In Table 2, we show examples of metrics that can be associated with the base BN.

3 Validation

To validate the framework in terms of completeness, we evaluated the anti-patterns presented in Eloranta et al. [4], in which an empirical study with 11 companies was executed. They identified 14 anti-patterns in adopting Scrum and their consequences, namely: (i) big requirements documentation, (ii) customer Product Owner, (iii) Product Owner without authority, (iv) long or non-existent feedback loops, (v) unordered product backlog, (vi) work estimates given to teams, (vii) hours in progress monitoring, (viii) semi-functional teams, (ix) customer caused disruption, (x) no

Table 2. Examples of metrics that can be associated with the base BN.

Node	Metrics
Code	Static analysis warnings, number of test cases, number of defects and code coverage.
Product Backlog	Percentage of ready items.
Planning meeting	Percentage of completed items.
Details	INVEST [3].
Execution	Sprint velocity.
Sprint	Running Tested Features e velocity.
Increment	Customer satisfaction, delivered value e agileEVM [18].

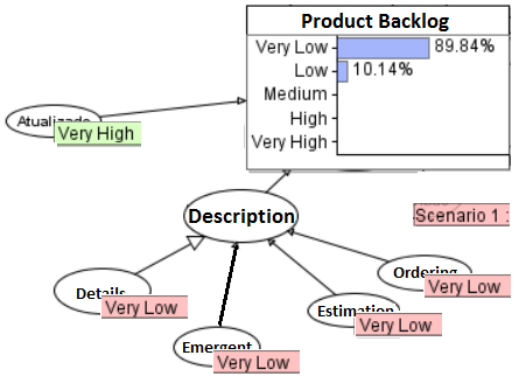

Figure 6. Results of the BN for anti-pattern (i).

sprint retrospective, (xi) invisible progress, (xii) varying sprint length, (xiii) too long sprints and (xiv) testing in next sprint. The goal of our validation was to identify if the proposed solution can identify them. Furthermore, we analyzed if the calculations are in conformance with the consequences of the anti-patterns. Due to space limitations, we only present the analysis of one anti-pattern in this paper.

For anti-pattern (i), the consequences of this pattern are that the team does not understand the requirements and that they are not ordered. It is related to the node *Product Backlog*, which is not an input node (i.e., leaf). Therefore, we analyze this pattern by defining evidences for the nodes *Details*, *Ordering*, *Emergent* and *Estimation*. As previously mentioned, in this context, the requirements are not ordered and emergent. Furthermore, if the team does not understand the requirements, it means that they are not detailed properly, as consequence, not estimated properly. By analyzing the calculated BN shown in Figure 6, we conclude that anti-pattern (i) is detected by the BN.

The anti-pattern (vi) is partially detected by the BN. The consequences of this anti-pattern are unrealistic goals and team demotivation. Even though it is related with the nodes *Estimation* and *Sprint goal*, the model does not present a causality between the quality of the estimations and the quality of the teamwork (i.e., motivation). On the other hand, the BN assists on assessing if the team is motivated in the fragment *Teamwork*. For instance, demotivated teams will probably not collaborate and share leadership.

The anti-pattern (x) is considered invalid because it is related to the Sprint Retrospective meeting, which is not modeled by the BN, as previously explained. The remaining anti-patterns are detected by the need of entering evidence on input nodes or analyzing the BN.

4 Final remarks

In this paper, we presented a framework to build a BN to assist on the assessment of Scrum-based software development methods. The BN models the main entities of the software development process and can be complemented with software practices and metrics. With the resulting BN, it is possible to monitor if the agile method is being followed and if it meets the needs of the project.

We evaluated the completeness of the solution through simulations to check if the proposed framework diagnoses 14 known Scrum anti-patterns (i.e., ScrumBut) presented in Eloranta et al. [4]. From the 14 known Scrum anti-patterns, 1 is invalid, 12 were directly detected and 1 was indirectly detected by the constructed model.

Even though our solution is an evolution of published studies [14, 13], which were evaluated in the industry, the main limitation of this study is the validation process. As future works, we will evaluate the proposed framework through case studies in the industry. To ease the usage on industry context, a tool to support the usage of the proposed framework, hiding the complexity of Bayesian networks from the practitioner, is currently under development.

The main contributions for practitioners are that the solution helps to configure and adopt Scrum-based agile methods and to develop measurement programs to continuously improve the maturity of the team and delivery process. For researchers, it can serve as a basis to configure empirical studies on Scrum-based agile methods and improve the state-of-art of agile methods measurement programs, adoption and maturity.

References

[1] S. Brinkkemper. Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*, 38(4):275 – 280, 1996. Method Engineering and Meta-Modelling.

[2] A. S. Campanelli and F. S. Parreiras. Agile methods tailoring - a systematic literature review. *J. Syst. Softw.*, 110(C):85–100, Dec. 2015.

[3] S. Downey and J. Sutherland. Scrum metrics for hyperproductive teams: How they fly like fighter aircraft. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 4870–4878, Jan 2013.

[4] V.-P. Eloranta, K. Koskimies, and T. Mikkonen. Exploring scrumbutan empirical study of scrum antipatterns. *Information and Software Technology*, 74:194 – 203, 2016.

[5] N. Fenton and M. Neil. *Risk Assessment and Decision Analysis with Bayesian Networks*. CRC Press, 5 edition, 11 2012.

[6] N. E. Fenton, M. Neil, and J. G. Caballero. Using ranked nodes to model qualitative judgments in bayesian networks. *IEEE Trans. on Knowl. and Data Eng.*, 19(10):1420–1432, Oct. 2007.

[7] R. M. Fontana, I. M. Fontana, P. A. da Rosa Garbuio, S. Reinehr, and A. Malucelli. Processes versus people: How should agile software development maturity be defined? *Journal of Systems and Software*, 97:140 – 155, 2014.

[8] R. M. Fontana, V. Meyer, S. Reinehr, and A. Malucelli. Progressive outcomes. *J. Syst. Softw.*, 102(C):88–108, Apr. 2015.

[9] S. Grapenthin, S. Poggel, M. Book, and V. Gruhn. Improving task breakdown comprehensiveness in agile projects with an interaction room. *Information and Software Technology*, 67:254 – 264, 2015.

[10] T. Javdani Gandomani and M. Ziaei Nafchi. An empirically-developed framework for agile transition and adoption. *J. Syst. Softw.*, 107(C):204–219, Sept. 2015.

[11] E. Kupiainen, M. V. Mntyl, and J. Itkonen. Using metrics in agile and lean software development a systematic literature review of industrial studies. *Information and Software Technology*, 62:143 – 163, 2015.

[12] 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*, SAC '13, pages 1037–1042, New York, NY, USA, 2013. ACM.

[13] M. Perkusich, K. Gorgonio, H. Almeida, and A. Perkusich. Assisting the continuous improvement of scrum projects using metrics and bayesian networks. *Journal of Software: Evolution and Process*, 2016. Article in Press.

[14] 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.

[15] R. Pichler. *Agile Product Management with Scrum: Creating Products that Customers Love*. Addison-Wesley Professional, 1 edition, 4 2010.

[16] A. Qumer and B. Henderson-Sellers. A framework to support the evaluation, adoption and improvement of agile methods in practice. *Journal of Systems and Software*, 81(11):1899 – 1919, 2008.

[17] F. S. Silva, F. S. F. Soares, A. L. Peres, I. M. de Azevedo, A. P. L. Vasconcelos, F. K. Kamei, and S. R. de Lemos Meira. Using {CMMI} together with agile software development: A systematic review. *Information and Software Technology*, 58:20 – 43, 2015.

[18] T. Sulaiman, B. Barton, and T. Blackburn. Agileevm - earned value management in scrum projects. In *AGILE 2006 (AGILE'06)*, pages 10 pp.–16, July 2006.

[19] J. Sutherland and K. Schwaber. The scrum guide. http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf, 2017. Accessed in: 03-10-2017.

[20] K. Verbert, R. Babuka, and B. D. Schutter. Bayesian and dempstershafer reasoning for knowledge-based fault diagnosisa comparative study. *Engineering Applications of Artificial Intelligence*, 60:136 – 150, 2017.

[21] J. Vlietland, R. van Solingen, and H. van Vliet. Aligning codependent scrum teams to enable fast business value delivery: A governance framework and set of intervention actions. *Journal of Systems and Software*, 113:418 – 429, 2016.

[22] S. Wagner, A. Goeb, L. Heinemann, M. Kls, C. Lampasona, K. Lochmann, A. Mayr, R. Plsch, A. Seidl, J. Streit, and A. Trendowicz. Operationalised product quality models and assessment: The quamoco approach. *Information and Software Technology*, 62:101 – 123, 2015.

[23] L. Williams. What agile teams think of agile principles. *Commun. ACM*, 55(4):71–76, Apr. 2012.

[24] L. Williams, K. Rubin, and M. Cohn. Driving process improvement via comparative agility assessment. In *Proceedings of the 2010 Agile Conference*, AGILE '10, pages 3–10, Washington, DC, USA, 2010. IEEE Computer Society.