Effort Prediction in Agile Software Development with Bayesian Networks

Laura-Diana Radu[a]
Department of Accounting, Business Information Systems and Statistics, Faculty of Economics and Business Administration, Alexandru Ioan Cuza University of Iasi, Blvd. Carol I, Iasi, Romania

Keywords: Agile Methodologies, Effort Estimation, Teamwork Quality, User Stories Characteristics.

Abstract: The success rate of software projects has been increased since agile methodologies were adopted by many companies. Due their flexibility and continuous communication with clients, the main reason for the failure has shifted from the formulation and understanding of the requirements to inaccurate effort estimation. In recent years, several researchers and practitioners have proposed different estimation techniques. However, some projects are still failing because the budget and/or schedule are not accurately estimated since there still are numerous uncertain variables in software development process. Previous team collaborations, expertise and experience of team members, frequency of changing requirements or priorities are just a few examples. To improve the accuracy of effort estimation, this research proposes a model for agile software development project prediction using Bayesian networks. Based on literature review and practitioners' knowledge, we identified two major categories of factors that influence effort needed: teamwork quality and user stories characteristics. We identified the sub-factors for each category and inter-dependencies between them. In our model, these factors are the nodes of the directed acyclic graph. The model can help agile teams to obtain a better software effort estimation.

1 INTRODUCTION

Agile software development consists of frequent or continuous delivery of software. The predictability and stability of traditional methods were replaced with flexibility and, consequently, agility, to generate value as quickly as possible. The previous studies indicate that the adoption of agile methodologies is beneficial in project implementation. According to a report published by PwC (2017), agile projects are 28% more successful than traditional. The authors of Agile Manifesto consider that the following values are more important: "individual and interactions over process and tools", "working software over comprehensive documentation", "customer collaboration over contract negotiation" and "responding to change over following a plan" (Beck et al., 2001). These characteristics are suitable in the current context, extremely dynamic, characterized by rapid changes in the market, technology and business environment (Freire et al., 2018). There are many agile frameworks, including Scrum, eXtreme Programming (XP), Agile Unified Process (AUP), Dynamic System Development Methodology (DSDM), Feature-Driven Development (FDD), Adaptive Software Development (ASD), Crystal and Lean Software Development (LSD). All have the following common characteristics: support continuous delivery of valuable software, strong collaboration between team members and co-location, and adaptability of the process during the entire lifecycle of the project (Raslan and Darwish, 2018). These frameworks proved that have the potential to reduce software development time and costs due to minimization of the amount of information transferred between the client and the development team and the elimination of the intermediate steps between the adoption of the decisions and their application (Weflen, 2018).

Agile principles don't guarantee the success of the projects. According to the studies published the rate of projects failure (Standish Group, 2014), including those that are using agile processes (Standish Group, 2015), is quite high. The most important causes are

[a] https://orcid.org/0000-0002-1463-8369

Radu, L.
Effort Prediction in Agile Software Development with Bayesian Networks.
DOI: 10.5220/0007842802380245
In *Proceedings of the 14th International Conference on Software Technologies (ICSOFT 2019)*, pages 238-245
ISBN: 978-989-758-379-7

incomplete requirements, lack of user involvement, resources or executive support and changing requirements and specifications. Realistic requirements and estimation of the proper time horizon for their implementation and delivery are challenges for many teams, including those that apply agile principles. These are important conditions for the avoidance of budget overruns and delayed dates of delivery (López-Martínez, 2017a).

Many uncertainties affect the planning process. Effort estimation is one of them. According to Trendowicz and Jeffery (2014), it is performed to manage and reduce project risks.

There are many practices of effort estimation, such as Planning Poker, T-Shirt, bucket systems, etc. They require the participation of all team members. Each of them has the right to express an opinion on the necessary effort to perform a task. An expected result of this practice is a greater commitment to the team. Story point is used to estimate the effort by many teams. This is an estimation of the overall effort needed to implement completely a piece of work, including the complexity and the risk associated with its implementation. The concept of velocity is used to measure the team's progress rate during a single sprint (Scott and Pfahl, 2018). This is the sum of the story-point estimates of the issues that the team resolved during an iteration and it used to predict when a release or a software should be completed (Choetkiertikul et al., 2018). The velocity depends by team expertise and experience, product complexity, the quality of requirements, communication and collaboration between team members and clients, etc.

Effort estimation is subjective in nature. It depends on developers' expertise and previous experiences with similar projects and tasks. According to previous studies, the average error in effort estimation ranges between 20% and 30% (Schweighofer, 2016). As a result, a method to improve the accuracy of effort estimation and to reduce the number of errors is welcome. A BN model can help the team assure the consistency of effort estimation especially in the case of complex projects with many issues.

This paper is organized as follows. Section 2 presents related work. Section 3 presents the proposed model and Section 4 presents our conclusions, limitations and future works.

2 RELATED WORKS

Agile frameworks assume incremental software development in small iterations or continuous delivery. Effort estimation needed to implement the functionalities will be achieved progressively. The following techniques can be used, either individually or aggregated, expert judgement, analogy with similar functionalities developed in previous projects or disaggregation.

Planning Poker is one of the most commonly used methods of project estimation. It combines elements of all three techniques (Usman, 2014). Each of them has a certain level of uncertainty as they are largely based on the professional experience and opinions of the experts in order to minimize the uncertainty and to improve the accuracy of the estimates. According to Zare et al. (2016), effort estimation methods are divided into two categories: model-based and expert-based methods. The first category contains methods that use statistical tools, such as statistical regression model (López-Martín, 2015), and methods that use artificial intelligence. The latter uses the following types of machine learning (ML) techniques: Case-Based Reasoning (CBR), Artificial Neural Networks (ANN), Decision Trees (DT), Bayesian Networks (BN), Support Vector Regression (SVR), Genetic Algorithms (GA), Genetic Programming (GP) and Association Rules (AR) (Wen et al., 2012). Based on ML, predictive models were developed to estimate the effort needed in solving issues (Porru et al.) or to assign story points (Choetkiertikul et al., 2018).

Several authors have proposed BN models for effort estimation in software development with agile methods. For instance, Dragicevic et al. (2017) use the following set of elements: working hours, requirements complexity, developer skills, type of task (new or recurrent), form complexity, function complexity, report complexity, and specification quality, to develop a BN model for task effort prediction. Karna and Gotovac (2015) consider three major entities involved in the estimation process: project, work items and estimators. They identify sets of attributes for each entity based on data collected from projects executed within Croatian software companies. Hearty et al. (2009) combine sparse data, prior assumption, and expert judgement into a BN model for predicting project velocity in extreme programming environment. Zare et al. (2016) present a three-level BN based on COCOMO components to estimate the effort needed in Man-Month for the software development. The Magnitude of Relative Error (MRE), the Mean Magnitude of Relative Error (MMRE), and the Prediction at Level m (Pred. (m)) are the most widely used metrics to assess the average estimation accuracy (Dantas et al., 2018; Port and Korte, 2018). Zahraoui et al. (2015) adjusted story point calculation of Scrum projects using three

elements: priority, size and complexity. They proposed to calculate an adjusted velocity with a new adjusted story point measure. They didn't decompose the complexity in sub-factors. Trendowicz and Jeffery (2014) divided the effort estimation factors in context and scale factors:

- Context factors include the programming language, application domain of the software, development type (e.g. new development, enhancement, or maintenance) and development life cycle (e.g., waterfall, incremental, or scrum).
- Scale factors include software size, complexity of project, management activities, complexity of system interfaces and integration, project maturity, project duration, team size and structure, project novelty and the project budget.

Based on previous studies, same authors classified the most relevant effort driver into four groups presented in Table 1.

Table 1: Effort drivers (Trendowicz and Jeffery, 2014; Schweighofer et al., 2016).

Drivers	Definition
Personnel factors	related to characteristics of human resources involved in the software development project (expertise, abilities and motivation of stakeholders, analysts, designers, programmers, project managers, users, maintainers, etc.).
Process factors	related to characteristics of software processes – methods, tools, and technologies used during software development.
Product factors	related to characteristics of all artifacts created throughout entire life cycle (requirements, software code and documentation).
Project factors	related to characteristics of project management (the use of agile practices) and organization, resource management, development constraints, and working conditions.

Curcio et al. (2018) identified three groups of obstacles for agile requirements engineering:

- Related to the environment that include difficulties with communication in distributed teams;
- Related to the people that include difficulties on finding specific and specialized skills, minimal and up to date documentation, customer availability, limited customer knowledge regarding requirements definition and his inability in terms of decision making, accuracy of estimations.
- Related to resources: lack of specialized tool, lack of documentation, inappropriate architecture, growing technical debt and imprecise cost and schedule estimation.

It is obvious that better teamwork creates better software (Freire et al., 2015), but better user stories will help team members to understand correctly the client needs and requirements and will reduce the time needed for additional explanations. Both aspects will favourable influence the accuracy of effort estimation and team's velocity.

Based on related works, the model presented in this article combines team-specific quality with user stories-specific characteristics to increase the accuracy of effort estimation.

3 BAYESIAN NETWORK MODEL

According to Tanveer et al. (2016), developers' experience and knowledge together with the impact and complexity of the system's changes influence the accuracy of estimations. They conducted a case study research with three agile development teams in a German multinational software company. The accurate estimation of costs and time is important for contract negation, human resources allocation, prioritizing the tasks, etc. In the case of underestimation, lack of necessary budget or failure to complete projects in time may occur.

3.1 Key Factors of Effort Prediction

In this paper, we propose a BN-based model to assess the effort prediction in the case of developing software with agile methodologies. Researchers have applied such a model in software engineering for process improvement, productivity prediction of the teams that use XP, risk evaluation and product quality prediction (Freire et al., 2018; López-Martínez et al., 2017a; Karna and Gotovac, 2015; Hearty et al., 2009; Lai, 2017). Our model is generic, since there are many different agile frameworks and it is impossible to build a universal model that matches for all these frameworks. Also, the differences between projects, between teams' size, organisational culture, the variety of enterprises business goals, etc. make it difficult, if not impossible, to use a universal model.

Based on the above-mentioned literature the factors that influence effort needed in software project development can be classified into two categories: teamwork quality and user stories characteristics, summarized in Table 2.

Table 2: Effort prediction key factors.

Categories	Key factors
Teamwork quality (Freire, 2018; Scott and Pfahl, 2018; Dragicevic, 2017; Hoegl et al., 2001; Weimar et al., 2013; Moe et al., 2009)	Autonomy
	Collaboration
	Self-management
	Trust
	Backup
	Feedback
	Communication
	Team learning
	Expertise
	Shared leadership
User stories characteristics (Dragicevic et al., 2017; Karna and Gotovac, 2015; Hearty et al., 2009; López-Martínez et al., 2017a; López-Martínez, 2017b; Chow and Cao, 2008; Lai, 2017)	Stability of requirements
	Integration
	Clarity
	Ordering
	Quality of specifications
	Complexity
	Size
	Completeness
	Consistency
	Discussable
	Understandable

3.2 Bayesian Networks

BN combines principles from graph theory, probability theory, computer science and statistics to represent knowledge about uncertain domains (Ben-Gal, 2008). A BN is a directed acyclic graph (DAG) that represents "a joint probability distribution over a set of random variables" (Freire et al., 2018; Friedman et al., 1997). DAG is defined by two sets: the set of nodes which represent random variables and the set of directed edges which represent direct dependence among variables, V (Ben-Gal, 2008). The BN is defined by a pair B = {G, Θ}, where G is DAG whose nodes X_1, X_2, …., X_n correspond to the random variables and whose edges represent the direct dependences between variable. Θ is the set of probability functions and contains the parameter $\theta_{x_i|\pi_i} = P_B(x_i|\pi_i)$ for each x_i in X_i conditioned by π_i the set of parameters X_i in G (Freire et al., 2018; Friedman et al., 1997). The joint probability distribution of B over variables V (Freire et al., 2018) is presented in eq. 1.

$$P_B(X_i, \ldots . X_n) = \prod_{i=1}^{n} P_B(x_i|\pi_i) = \prod_{i=1}^{n} \theta_{x_i|\pi_i} \quad (1)$$

According to Perkusich et al. (2015), the problem of BN building can be divided in two stages: the DAG construction and the Node Probability Tables (NPT) definition.

3.3 Bayesian Networks Construction

Based on literature and discussions with practitioners, we identified the relationship between key factors. A top-down approach is used to decompose the top-level node into key factors that can be observed by experts or can be collected and analysed with specific tools. The complete DAG is shown in Figure 1. Next, in this section, we discuss in more details the key factors mentioned in previous section.

The success of agile methodologies depends of the quality of team members and their ability to auto-organize and the project requirements definition. The main goal is to satisfy customers with working functionalities. During the development phase or at the end of each sprint, the product should be inspected and adapted by customers. The changes, although welcome, can lead to significant delays in product completion. As a result, it is necessary to find an equilibrium between the value that a feature will bring and its influence on the development of the project.

The completeness, consistency and understandability of project requirements can reduce time between their formulation and delivery. Therefore, we added the nodes *complete*, *consistence* and *understandable* as parents of the node *Clarity*.

Further, clear and simple requirements will help the team to increase the velocity. This can be difficult for client, but team members will not need additional information for each functionality. Also, they should be ordered correctly in Product Backlog with respect to technical dependencies and agile principles that promote the development of functionalities based on their business value. Therefore, we added the nodes *Ordering*, *Clarity* and *Stability* as parents of the node *Quality*, in the case of user stories characteristics.

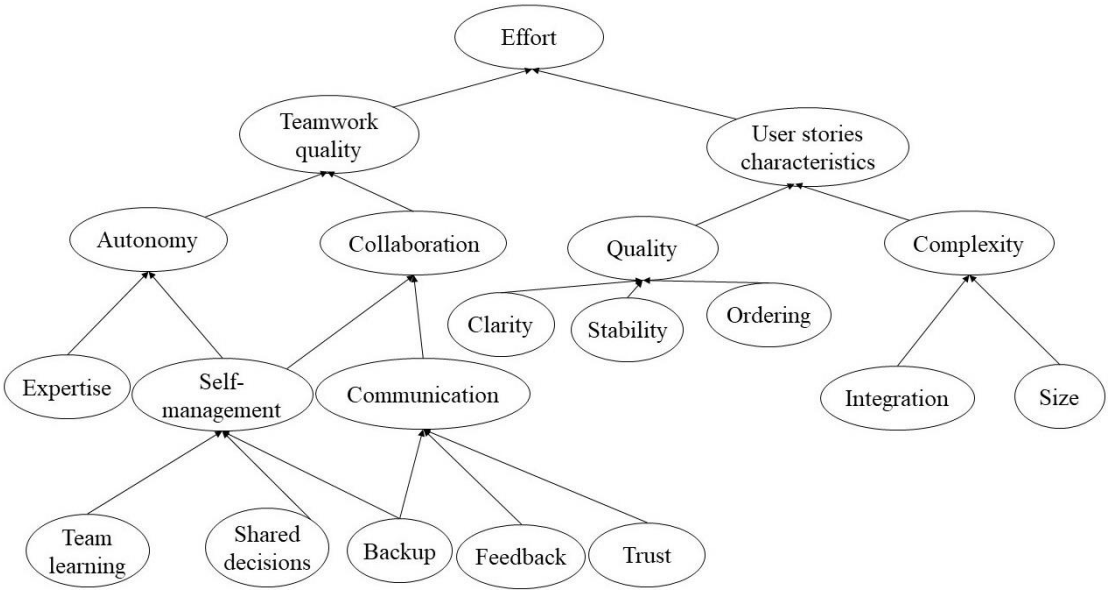

Figure 1: DAG of the BN.

Project functionalities *integration* and its *size* evaluated as source lines of code reflect the difficulty of project implementation. We added these nodes as parents of the node *Complexity*.

Team learning is defined as the ability to identify team members' changes and the adaptation of the strategies according to them. *Shared decisions* process is described as the sharing of decision authority and leadership. *Backup* represents the understanding of the other members' tasks and the availably to assist them. These three characteristics are important for *self-management teams* (Freire el., 2018; Moe et al., 2009) since a benefit of agile adoption is the reduction of time between a decision and the outcome of that decision (Weflen et al., 2018). This feature, along with *Expertise* of the team members, are added to the model as parent nodes of *Autonomy*.

Feedback involves the exchange of information among team members and it is parent node for *Communication* along with *Backup* and *Trust*. According to previous research, the lack of trust limits the share of information between team members, because it creates the fear of being perceived as incompetent (Moe et al., 2009; Perkusich et al., 2015).

The ideal agile team is small, co-located and communicates face-to-face daily. All members work together to achieve some common goals. *Collaboration* is a strong commitment towards the team and the team members to achieve the common goals. For the team to collaborate, they must communicate and must be able to self-manage. We added these nodes as parents of the node Collaboration.

Team autonomy is an important key factor for teamwork quality. According to Moe et al. (2009), this is the ability of the team to regulate their boundary conditions in regards with the influence of management or other individuals on its activities. We added team autonomy and Collaboration as parents of the node *Teamwork Quality*.

According to Perkusich et al. (2013) each key factor represents a set of tuples $N=\{(s_1,p_1), \ldots,(s_{|N|},p_{|N|})\}$, where s_i is a possible state and p_i is the probability of each state i to appear. For identifying the elements of N, each node can be mapped to a set of practices and is composed of a 5-point Likert scale: Very Low (VL), Low (L), Medium (M), High (H) and Very High (VH) (Freire et al., 2018; López-Martínez et al., 2017a, 2017b). During the project's lifecycles, the team should register the processes and practices used for each key factor. Furthermore, the base BN can be complemented with metrics. Some metrics, as size can be automatically collected. Other metrics, such as ordering, clarity, communication, expertise or shared decisions, are manually collected. To identify the elements of N, we defined that each node is composed of 5-point Likert scale mentioned above. Each state reflects to the quality of the node. Based on agile experts' answers, we defined the NPT. This paper represents research-in-action and, as such, the validation of rank nodes is not complete and will be improved in future research. Table 3 illustrates the calculated values of user stories characteristics in the case of excellent formulated requirements and a

complex project (VL is very low, L is low, M is medium, H is high and VH is very high). The user stories are clear, stable and ordered. We used WMEAN function (eq. 2) (Dimitrova and Petkova, 2015) with different weights for each attribute. The variance is 0.0005, the lower bound is 0 and the upper bound is 1.

$$WMEAN = \frac{\sum_{i=1\ldots n} w_i X_i}{n} \quad (2)$$

In formula (2), $w_i \geq 0$ are weight, and *n* is number of parent node. The syntax of this function in AgenaRisk is: wmean(w_1, parent$_1$, w_2, parent$_2$,…w_n, parent$_n$).

Table 3: The calculated values for user stories characteristics.

	State				
	VL	L	M	H	VH
User stories characteristics	0	0	0.31	0.69	0
Quality	0	0	0	0.03	0.97
Complexity	0	0	0	0.04	0.96

Table 4 illustrates the calculated values of teamwork quality for an ideal team. It has adequate experience with agile practices, team's members collaborate properly, and the team is autonomous. We use the same function, variance and bounds.

Table 4: The calculated values for teamwork quality.

	State				
	VL	L	M	H	VH
Teamwork quality	0	0	0	0.11	0.89
Autonomy	0	0	0	0.02	0.98
Collaboration	0	0	0	0.14	0.86

In our example, the effort calculated based on previous values of the nodes, will be: Medium – 0.03, Low 0.65 and Very low – 0.32. If the quality of team is very high, user stories are well defined and the project is complex, the effort needed to implement the requirements is low. Team's members will be able to develop complex functionalities since they communicate and collaborate, have the abilities and the right to adopt and apply decisions, they understand properly the requirements since these are clear, ordered and stable.

To validate the completeness of the model, Eloranta et al., (2016) identified 14 anti-patterns in adopting Scrum based on a study expected in 11 companies: (1) big requirements documentation, (2) customer product owner, (3) product owner without authority, (4) long or non-existent feedback loops, (5) unordered product backlog, (6) work estimates given to teams, (7) hours in progress monitoring, (8) semifunctional teams, (9) customer caused disruption, (10) no sprint retrospective, (11) invisible progress, (12) varying sprint length, (13) too long sprints and (14) testing in next sprint. We only considered the anti-patterns that are valid for our BN.

We present only the analysis for one anti-pattern in this section. For big requirements document, the consequence is the ambiguity. The requirements are stable, but they are not ordered, and are unclear. This is related to Quality, which is not an input node. We define evidences for the nodes *Complete*, *Consistent*, *Discussable* and *Understandable* to analyse this pattern. The result is presented in Figure 2.

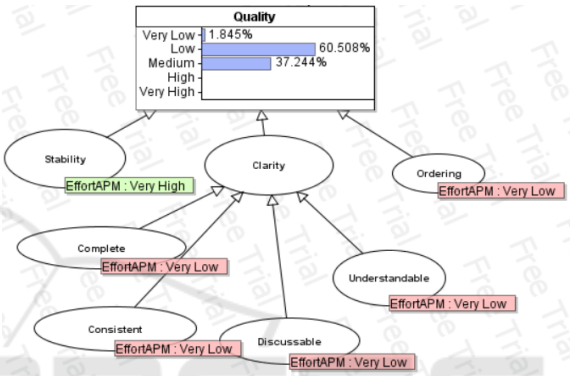

Figure 2: BN for anti-pattern "big requirements document".

The requirements are incomplete and inconsistent. Furthermore, the client is not available to discuss them and, as consequence, the team doesn't understand the functionalities that has to be developed. The calculated BN shown in Figure 2 is shown that antipattern (1) is detected by the BN. Some anti-pattern related to sprint length (13), testing (14) and agile ceremonies (10) are considered invalid. They are not modelled by this BN.

4 CONCLUSIONS

This paper proposes a model of BN to assist the agile teams to estimate the effort needed in the case of software projects. To build the model, we performed a literature review and discussed with experts to identify the main factors that influence the effort needed. We identified two categories of factors: teamwork quality and user stories characteristics. In the case of teamwork quality, the key factors are autonomy, collaboration, self-management, trust, backup, feedback, communication, team learning, expertise and shared leadership. User stories

characteristics that influence the effort are: stability of requirements, integration, clarity, stability of specifications, ordering, complexity, completeness, consistency, discussable and understandable size.

The main limitation of the model is the validation. For future research, we intend to validate the model in two stages: NPT validation and model validation. We will define more scenarios and will compare, in collaboration with experts, the expected output with actual results to conclude if they are acceptable. We will complete the model and the final version will be evaluated it through case studies in the software companies that use agile methodologies. The main contribution of the study is the integration of teamwork quality and user stories characteristics into the same model for estimating efforts needed for developing functionalities in software projects.

ACKNOWLEDGEMENTS

This project is funded by the Ministry of Research and Innovation within Program 1 – Development of the national RD system, Subprogram 1.2 – Institutional Performance – RDI excellence funding projects, Contract no.34PFE/19.10.2018.

REFERENCES

Agena Ltd. (2018), *Bayesian Network Software for Risk Analysis and Decision Making*, www.agenarisk.com.

Al-Rouson, T., Sulaimin, S., and Salam, R. A. (2009). Supporting architectural design decision through risk identification architecture pattern (RIAP) model. *WSEAS transactions on information science and applications*, 6(4): 611-620.

Beck, K. and several authors (2001). Agile Manifesto. Website. http://www.agilemanifesto.org.

Ben- Gal, I. (2008). Bayesian networks. In *Encyclopedia of Statistics in Quality and Reliability*, pages 1-6. John Wiley & Sons, Ltd., New York.

Choetkiertikul, M., Dam, H. K., Tran, T., Pham, T. T., Ghose, A., and Menzies, T.(2018). A deep learning model for estimating story points. *IEEE Transactions on Software Engineering*, pages 1-21, IEEE Comput. Soc.

Chow, T., and Cao, D. B. 2008. A survey study of critical success factors in agile software projects. *Journal of systems and software*, 81(6): 961-971.

Curcio, K., Navarro, T., Malucelli, A., & Reinehr, S. (2018). Requirements engineering: A systematic mapping study in agile software development. *Journal of Systems and Software*, 139, 32-50.

Dantas, E., Perkusich, M., Dilorenzo, E., Perkusich, A., de Almeida, H., and Santos, S. (2018). Effort Estimation in Agile Software Development: an Updated Review. In *Proceedings of the 30th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pages 1-10, ACM Press.

Dimitrova, L., and Petkova, K. (2015). Analysis and assessment of injury risk in female gymnastics: Bayesian Network approach. *TEM Journal*, 4(1): 83-95.

Dragicevic, S., Celar, S., and Turic, M. (2017). Bayesian network model for task effort estimation in agile software development. *Journal of Systems and Software*, 127: 109-119.

Eloranta, V. P., Koskimies, K., and Mikkonen, T. (2016). Exploring ScrumBut—An empirical study of Scrum anti-patterns. *Information and Software Technology*, 74: 194-203.

Freire, A. S., da Silva, R. M., Perkusich, M., Almeida, H., and Perkusich, A. (2015). A Bayesian Network Model to Assess Agile Teams' Teamwork Quality. In *Proceedings of the 29th Brazilian Symposium on Software Engineering*, pages 191-196. IEEE.

Freire, A. S., Perkusich, M., Saraiva, R., Almeida, H., and Perkusich, A. (2018). A Bayesian networks-based approach to assess and improve the teamwork quality of agile teams. *Information and Software Technology*, 100: 119-132.

Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, 29: 131-163.

Hearty, P., Fenton, N., Marquez, D., and Neil, M. (2009). Predicting Project Velocity in XP Using a Learning Dynamic Bayesian Network Model. *IEEE Transactions on Software Engineering*, 35(1): 124-137.

Hoegl, M., and Gemuenden, H. G. (2001). Teamwork quality and the success of innovative projects: A theoretical concept and empirical evidence. *Organization science*, 12(4): 435-449.

Karna, H., and Gotovac, S. (2015). Estimating software development effort using Bayesian networks. In *Proceedings of the 23rd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 229-233, IEEE Comput. Soc.

Lai, S. (2017). A User Story Quality Measurement Model for Reducing Agile Software Development Risk. *International Journal of Software Engineering & Applications*, 8(2): 75-86.

López-Martín, C. 2015. Analyzing and handling local bias for calibrating parametric cost estimation models. *Applied Soft Computing*, 27: 434-449.

López-Martínez, J., Ramírez-Noriega, A., Juárez-Ramírez, R., Licea, G., and Jiménez, S. (2017a). User stories complexity estimation using Bayesian networks for inexperienced developers. *Cluster Computing*, 21(1): 1-14.

López-Martínez, J., Juárez-Ramírez, R., Ramírez-Noriega, A., Licea, G., and Navarro-Almanza, R. (2017b). Estimating user stories' complexity and importance in Scrum with Bayesian networks. In *Proceedings of the World Conference on Information Systems and*

Technologies, pages 205-214, Springer.

Moe, N. B., Dingsøyr, T., and Røyrvik, E. A. (2009). Putting agile teamwork to the test–an preliminary instrument for empirically assessing and improving agile software development. In *Proceedings of nternational Conference on Agile Processes and Extreme Programming in Software Engineering*, pages 114-123, Springer.

Perkusich, M., Soares, G., Almeida, H., and Perkusich, A. (2015). A procedure to detect problems of processes in software development projects using Bayesian networks. *Expert Systems with Applications*, 42(1): 437-450.

Perkusich, P., de Almeida, H., and Perkusich, A. (2013). A Framework to Build Bayesian Networks to Assess Scrum-based Software. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Applied Computing*, pages 1037-1042, ACM Press.

Porru, S., Murgia, A., Demeyer, S., Marchesi, M., and Tonelli, R. (2016). Estimating story points from issue reports. In *Proceedings of the 12th International Conference on Predictive Models and Data Analytics in Software Engineering*, article no. 2, ACM Press.

Port, D., and Korte, M. (2018). Comparative studies of the model evaluation criterions mmre and pred in software cost estimation research. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 51-60, ACM Press.

PwC (2017), *Agile Project Delivery Confidence. Mitigate project risks and deliver value to your business*. Website https://www.pwc.com/gx/en/services/audit-assurance/risk-assurance/agile-project-delivery-confidence.html

Raslan, A. T., and Darwish, N. R. (2018). An Enhanced Framework for Effort Estimation of Agile Projects. *International Journal of Intelligent Engineering and Systems*, 11(3): 205-214.

Schweighofer, T., Kline, A., Pavlic, L., and Hericko, M. (2016). How is Effort Estimated in Agile Software Development Projects? In *Proceedings of the SQAMIA 2016: 5th Workshop of Software Quality, Analysis*. pages 73-80, CEUR-WS.

Scott, E., and Pfahl, D. (2018). Using Developers' Features to Estimate Story Points. In *Proceedings of the 2018 International Conference on Software and System Process*, pages 106-110. Gothenburg, Sweden: ACM Press.

Standish Group (2014). *Chaos Report 2014.* Website https://www.projectsmart.co.uk/white-papers/chaos-report.pdf.

Standish Group (2015). *Chaos Report 2015.* Website https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf.

Tanveer, B., Guzman, L., and Engel, U. (2016). Understanding and improving effort estimation in agile software development – an industrial case study. In *Proceedings of the International Conference on Software and System Processes*, pages 41-49, ACM Press.

Trendowicz, A., Jeffery, R. 2014. *Software Project Effort Estimation. Foundations and Best Practice Guidelines for Success*. Springer, Cham.

Usman, M., Mendes, E., Weidt, F., and Britto, R. 2014. Effort Estimation in Agile Software Development: A Systematic Literature Review. In *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*, pages 82-91, ACM Press.

Weflen, E., Korniejczuk, K., Lau, S., Kryk, S., MacKenzie, C. A., and Rivero, I. V. (2018). Application of Bayesian Belief Network for Agile Kanban Backlog Estimation. In *Proceedings of the 2018 IISE Annual Conference*, Paper 118.

Weimar, E., Nugroho, A., Visser, J., and Plaat, A. (2013). Towards high performance software teamwork. In *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, pages 212-215, ACM Press.

Wen, J., Li, S., Lin, Z., Hu, Y., and Huang, C. (2012). Systematic literature review of machine learning based software development effort estimation models. *Information and Software Technology*, 54(1): 41-59.

Zahraoui, H., & Idrissi, M. A. J. (2015). Adjusting story points calculation in scrum effort & time estimation. In *Proceedings of the 10th International Conference on Intelligent Systems: Theories and Applications (SITA*). pages 1-8. IEEE.

Zare, F., Zare, H., and Fallahnezhad, M. (2016). Software effort estimation based on the optimal Bayesian belief network. *Applied Soft Computing*, 49: 968–980.