Using Bayesian Networks to estimate Strategic Indicators in the context of Rapid Software Development

Martí Manzano
Universitat Politècnica de Catalunya
Spain
mmanzano@essi.upc.edu

Emilia Mendes
University of Oulu
Finland
Emilia.Mendes@oulu.fi

Cristina Gómez
Universitat Politècnica de Catalunya
Spain
cristina@essi.upc.edu

Claudia Ayala
Universitat Politècnica de Catalunya
Spain
cayala@essi.upc.edu

Xavier Franch
Universitat Politècnica de Catalunya
Spain
franch@essi.upc.edu

ABSTRACT

Background: During Rapid Software Development, a large amount of project and development data can be collected from different and heterogeneous data sources. *Aims:* Design a methodology to process these data and turn it into relevant strategic indicators to help companies make meaningful decisions. *Method:* We adapt an existing methodology to create and estimate strategic indicators using Bayesian Networks in the context of Rapid Software Development, and applied it to a use case. *Results:* Applying the methodology in the use case, we create a model to predict product quality based on software factors and metrics, using companies' business knowledge and collected data. *Conclusions:* We proved the methodology's feasibility and obtained positive feedback from the company's use case.

CCS CONCEPTS

• **Software and its engineering → Software creation and management**

KEYWORDS

Strategic indicator, Bayesian Network, Decision Making, Rapid software development

1 Introduction

Rapid Software Development (RSD) is the organizational capability to develop, release, and learn from software in rapid cycles without compromising its quality [1]. During RSD, a large amount of project and development data (e.g., number of to do issues, acceptance-testing time) is available in several data sources (e.g., JIRA, Git). These data may be processed and analyzed with the purpose of turning into meaningful and relevant Strategic Indicators (SIs) to inform decision-makers about how the software development is progressing in each iteration, what is especially valuable in RSD. Examples of SIs are customer satisfaction, estimated effort and, in general, any aspect that a company considers relevant for strategic decision-making.

In this context, defining SIs becomes a challenge. On the one hand, the aggregation of simple data into SIs has not been much investigated. On the other hand, this aggregation cannot be realistically expected to be universal, but instead each company may have its own intricacies yielding to different definitions. For satisfying both requirements, Bayesian Networks (BNs) emerge as a promising approach, because their definition does not require any existing framework to exist but only the availability of data and experts.

The two goals of this paper are: 1) To adapt an existing methodology for the construction and validation of SI prediction models using Bayesian Networks (BNs) in the context of RSD, and 2) To apply the methodology within the context of a use case at a company that uses RSD, to prove its feasibility.

As far as we know, no previous study has proposed a general methodology for defining SIs prediction models within the context of RSD [2]. However, there exist several studies proposing individual SIs using BNs in the area of software engineering, specifically, to estimate "teamwork quality" [3], to model quality in software projects [4], to estimate effort in software development [5], for requirements engineering [6], to predict software defects [7], and to estimate value and help decision making [8,9]. The difference between these works and ours is that we build a SI BN prediction model based upon companies' business knowledge and data automatically measured from heterogeneous data sources, and within the context of RSD.

The remaining of this paper is organized as follows: Section 2 introduces the context of this work. Section 3 describes the methodology used to build and validate SI prediction models and its application to a real scenario is presented in Section 4. Finally, conclusions, comments and future work are given in Section 5.

2 The Q-Rapids Approach

This work is carried out in the context of the Q-Rapids project[1]. Q-Rapids is a data-driven, quality-aware RSD tooled method in which quality requirements are identified from available data and evaluated with respect to some selected SIs [10]. Q-Rapids aims at increasing software quality and improving the development

[1] www.q-rapids.eu

process through: 1) Gathering and analyzing data from project management tools, software repositories, quality of service and system usage to continuously assess software quality; 2) Providing decision makers with a strategic dashboard to help them make requirements-related strategic decisions; 3) Extending the RSD process in a way that favors software quality, by considering the integration of quality and functional requirements, as well as their management.

In the Q-Rapids approach, the strategic dashboard visualizes information about a set of selected SIs. A SI (e.g., product quality, time to market) is defined as an aspect that the company considers relevant for the decision-making process [10]. In order to define SIs, the Q-Rapids Quality Model [11], links the data gathered from some data sources to the SIs rendered in the dashboard. Concretely, *Metrics* are computed from data gathered via *Data Sources* relating to a software product and development process. The gathering and computation is achieved using software data collectors, which periodically gather heterogeneous raw data from different data sources' repositories (e.g., SonarQube and Jira) and aggregate them into *Metrics* measured within the [0, 1] range. The *Metrics* are then combined into *Quality Factors* (QFs) via custom linear aggregation functions, and are measured within the same range. These QFs can be related to the product being developed (*Product Factors*) or to the software development process itself (*Process Factors*). Finally, QFs are aggregated into *Strategic Indicators*. More details on the Q-Rapids Quality Model and Q-Rapids approach can be found in [12,13].

3 The Methodology

In this section, we introduce the methodology used to build a SI BN prediction model in the context of RSD. It is based on the EKEBN (Expert-based Knowledge Engineering of Bayesian Networks) methodology [14] and the Weighted Sum Algorithm (WSA) [8], adapted to the Q-Rapids context. BNs allow incorporating company business knowledge and representing uncertainty that is inherent in a complex domain. Such representation and quantification use probabilities, and model the cause and effect relationships between the variables of a BN model [8]. BN models can be created using expert knowledge, pure data-driven approaches or a combination of both. Next sections describe the steps of our hybrid approach.

3.1 Structure Building

This step consists in the construction of the directed acyclic graph (DAG) that represents the BN structure. Such DAG includes, within the context of this work, two types of nodes: product/process QFs and a SI. Further, such DAG also includes arrows between nodes, which represent their cause and effect relationships. Herein a SI node is always the destination of arrows from QF nodes, which means that QFs are SI's predictors. While building a DAG, a company also needs to decide upon: i) the QFs that will be used to predict the chosen SI; ii) the cause and effect relationships between nodes; and iii) categorical scales to measure each of the nodes. In the case of QFs, it is also important to decide how each scale point matches the [0, 1] interval. For instance, a given QF node may have three categories: *Low, Medium,* and *High,* corresponding respectively to ranges [0 - 0.6), [0.6 - 0.85) and [0.85 - 1].

3.2 Uncertainty Quantification

The uncertainty quantification step represents the probabilistic quantification of every node's Conditional Probability Table (CPT). These probabilities are entered in order to define and quantify the relationships between these nodes, to reflect the knowledge of the expert stakeholders on the cause-effect relations between the selected QFs, and between those and the SI in the model being built. In our case, for parent nodes, i.e., nodes that do not have arrows pointing to them, probabilities are computed by the frequency quantification of the historical data gathered for the QFs associated with these nodes, using the corresponding scale types (categories).

To quantify the probabilities of the child nodes a semi-automatic approach is used based on the Weighted Sum Algorithm (WSA) [15]. The WSA uses expert knowledge to identify the probabilities of the most compatible states for a child node's parent nodes, and to assign weights to the parent nodes, representing their importance when computing the child's probabilities. A detailed explanation of the WSA algorithm is given in [8]. Such solution reduces the number of probabilities to be elicited. After finishing this step, the actual BN model is ready for its validation.

3.3 Model Validation

This step aims to verify the accuracy of the model and recalibrate it, if necessary. Generally, two validation methods are used in EKEBN: Model Walkthrough to assess the reliability of the model in terms of subjective accuracy [16,14] and Outcome Adequacy [17], to evaluate the model using real past scenarios. Within the context of this work, for Model Walkthrough, stakeholders have to come up with hypothetical scenarios, which represent hypothetical states for the QFs, and the expected SI state that should present the highest probability. The scenarios are entered in the BN model as evidence, and if the SI's state with the highest probability does not match to the experts' expectation, the model is recalibrated until it matches.

The Outcome Adequacy uses real past data to assess if the model is accurate. Therefore, it is necessary to dispose of past statuses of QFs and the resulting value of the SI computed using custom functions prior to the BN creation. If the SI's computed does not match to the output obtained by the BN, the model is recalibrated.

4 Pilot Use Case

A real scenario was considered in a company working in the telecommunications domain to create a company-specific SI BN prediction model. For this purpose, a workshop to apply the methodology introduced in Section 3 was conducted, with the help of three stakeholders: a software engineer with more than 4 years of experience, a project manager with almost 2 years of experience, and a software development specialist with more than

10 years of experience. The workshop lasted for 4 hours, divided into 2 sessions of 2 hours each. The first one covered the structure building and uncertainty quantification and the second one covered the model validation. The last participant only attended the second session. As an introduction, in the first session an example presenting the rationale of BNs was shown, and concepts like probabilities and causal relationships were clarified, along with the objectives and the results to emerge from the workshop.

4.1 Company Description

The company providing the use case develops distributed systems for telecommunication networks using a release-based development process based on agile and lean principles. They manage multiple product lines each one having their own products, combining hardware and software components. In relation to quality, they are interested in defining methodological support to manage quality requirements that are common to their product lines. To this aim, they want to define and manipulate appropriate SIs visualized through a strategic dashboard. The workshop reported next is the first step in such definition.

4.2 Structure Building

In this part of the workshop, the stakeholders decided which of the QFs being computed in their premises would take part in the SI assessment model. The SI selected was *Product Quality*, based on the *Code Quality* and *Software Stability* QFs, which had been computed and stored for six months in a software-testing environment. The first factor, *Code Quality*, is computed from several software metrics related to the software complexity, ratio of comments and percentage of duplicated code. The second factor, *Software Stability,* is computed from ratio of bugs' related metrics. These factors are aggregated from metrics coming out of several data sources like *SonarQube, Jira* and *Jenkins.* Fig. 1. shows the cause and effect relationships between QFs and the SI.

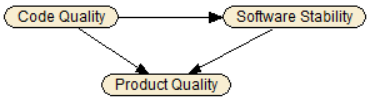

Figure 1: Product Quality BN model

The two domain experts defined the following scale types to measure every BN node: *Very Low, Low, Medium, High* and *Very High.* The ranges for each scale type are as follows:

Code Quality: [0 – 0.805), [0.805 – 0.807), [0.807 – 0.808), [0.808 – 0.9), [0.9 – 1).

Software Stability/Product Quality: [0 – 0.7), [0.7 – 0.85), [0.85 – 0.95), [0.95 – 0.99), [0.99 – 1).

Ranges provided for *Code Quality* node are narrow due to the low variance of its historical data. This part of the workshop, along with the introduction, lasted for 1.5 hours.

4.3 Uncertainty Quantification

The probabilities elicitation process for each node was the following:

4.3.1 Code Quality. This is a parent node; thus, its probability quantification was done using historical data of the computed QFs over a period of 6 months, discretized according to scale types and their correspondence in the [0, 1] range. The probabilities for this node's CPT are shown in Fig. 2 (a).

4.3.2 Software Stability. The probabilities for this node (see Table 1) were directly elicited from the stakeholders, conditionally on each of its parent node's states. For example, when *Code Quality* is 'Very High', there is a 70% probability of *Software Stability* being also 'Very High', a 20% of being 'High' and a 10% of being 'Medium'. Note that the WSA was not used here because it can only be used when a child node has at least two parent nodes.

Table 1: Elicited CPT for the node Software Stability

	Software Stability (%)				
Code Quality	**Very Low**	**Low**	**Medium**	**High**	**Very High**
Very Low	70	20	10	0	0
Low	65	15	20	0	0
Medium	20	30	40	10	0
High	0	20	30	45	5
Very High	0	0	10	20	70

Table 2: Elicited partial CPT for the node Product Quality

Parent nodes		Product Quality (%)				
Code Quality (w = 30%)	**Software Stability (w = 70%)**	**Very Low**	**Low**	**Medium**	**High**	**Very High**
Very Low	Very Low	100	0	0	0	0
Low	Very Low	95	5	0	0	0
Medium	Medium	0	50	50	0	0
High	High	0	0	20	75	5
Very High	Very High	0	0	0	10	90
Very Low	**Very Low**	100	0	0	0	0
Medium	**Low**	40	55	5	0	0
Medium	**Medium**	0	50	50	0	0
High	**High**	0	0	20	75	5
Very High	**Very High**	0	0	0	10	90

4.3.3 Product Quality. We used the semi-automatic technique WSA to ease the elicitation task. As both its parent nodes and the SI node have five scale types, a manual elicitation of all probabilities would require eliciting 5^3 probabilities. By using the WSA, the number of probabilities to be elicited decreased from 125 to 50 (i.e. 25 rows of 5 probabilities to 10 rows of 5 probabilities, one per SI's category, respectively). For example, when *Code Quality* is 'Very Low', the elicited most compatible state for the parent node *Software Stability* is also 'Very Low', and given such combination, the probability of *Product Quality* being 'Very Low' is 100%, and 0% for the other 4 states. The WSA required weights for the parental nodes were also elicited from the two stakeholders, and are shown in Table 2, along with the partial CPT for *Product Quality* node. These data was entered into the WSA to infer the full CPT. The uncertainty quantification step lasted for 30 minutes. Fig. 2 (a) shows the resulting BN.

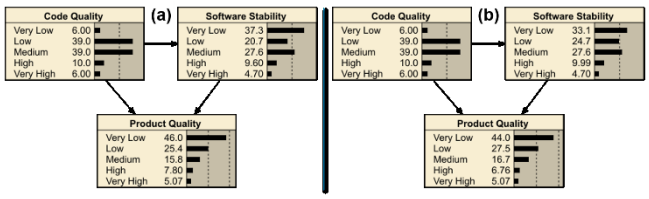

Figure 2: (a) BN obtained from steps 4.2 and 4.3. (b) BN obtained from step 4.4

4.4 Model Validation

In the two-hour second workshop session, we carried out the Model Walkthrough validation with the three stakeholders. Having the third stakeholder only in the validation step allowed achieving a more robust validation [18]. It is part of our future work to perform the Outcome Adequacy validation when building larger SI BN models for which SIs had been automatically measured during a past period of time, with a formula defined by the company's domain experts. As this requirement was not met in this use case, only Model Walkthrough validation was carried out. Stakeholders were asked to prepare jointly 10 hypothetical scenarios (see Table 3) for the SI BN model shown in Fig. 2 (a). Such scenarios were entered in the SI BN model one at a time, and were used to check whether the model provided the highest probability to the same state in the SI node corresponding to the stakeholders' expectation. In 4 out of the 10 scenarios, the SI node's CPT needed to be recalibrated (see Table 3). Recalibration represented the change of probabilities in order to match stakeholders' expectations. Fig. 2 (b) shows the validated BN.

Table 3: Scenarios used for model validation

Code Quality	Software Stability	Product Quality	Required calibration
Low	**Low**	**Low**	Yes
Low	**High**	**Medium**	Yes
Medium	**High**	**Medium**	Yes
High	**Medium**	**Medium**	No
High	**High**	**High**	No
Very Low	**Very Low**	**Very Low**	No
Very Low	**Medium**	**Low**	No
Very Low	**Low**	**Very Low**	Yes
Very High	**Very High**	**Very High**	No
Very High	**High**	**High**	No

5 Conclusions And Future Work

This work adapts an existing methodology to build SI prediction models for RSD, and presents its application to a use case using a combination of real data from a company, along with the expertise of the company's stakeholders. Our methodology is especially useful for RSD because it permits to recalibrate SI prediction models in an iterative and incremental way. After applying the methodology to the use case, the stakeholders claimed that this kind of SI prediction models could be useful to support their software quality decision-making processes.

The work introduced in this paper presents some limitations to be addressed as future work. Specifically, it is clear that the created model is simplistic in terms of the number of QFs included in the model. The small number of domain experts who provided the business knowledge during the workshop can also have an effect on the model's accuracy, causing it to be biased. Moreover, the Outcome Adequacy validation was not conducted since product quality was not being computed prior to the BN construction. Additionally, we need to compare our methodology with others using different models, as for instance the GSRM model to predict release dates in RSD [19].

As ongoing work, and to show the scalability of our proposal, we are now working on defining more complex SIs for software products and APIs of other companies.

ACKNOWLEDGMENTS

This work is a result of the Q-Rapids project, which has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement N° 732253. Also, this work is partially funded by the Spanish project GENESIS (TIN2016-79269-R). We thank to the company participating in the workshop for defining a prediction model for a SI.

REFERENCES

[1] L. Guzmán *et al.* "How Can Quality Awareness Support Rapid Software Development? - A Research Preview". REFSQ 2017.
[2] A. Tosun, A. B. Bener, and S. Akbarinasaji. "A systematic literature review on the applications of Bayesian networks to predict software quality". *SQJ 25(*1), 2017.
[3] A. Freire, M. Perkusich, R. Saraiva, H. Almeida, and A. Perkusich. "A Bayesian networks-based approach to assess and improve the teamwork quality of agile teams". IST 100, 2018.
[4] N. E. Fenton, P. Hearty, M. Neil, and Ł. Radliński. "Software Project and Quality Modelling Using Bayesian Networks". *Artif. Intell. Appl. Improv. Softw. Eng. Dev. New Prospect.*, 2009.
[5] L. Radlinski. "A survey of bayesian net models for software development effort prediction," *Int. J. Softw. Eng. Comput.* 2(2), 2010.
[6] I. M. del Águila and J. del Sagrado. "Bayesian networks for enhancement of requirements engineering: a literature review". *Requir. Eng.*, 21(4), 2016.
[7] A. Okutan, Olcay, T. Yıldız, A. Okutan, and O. T. Yıldız, "Software defect prediction using Bayesian networks". *Empir Softw. Eng*, 2014.
[8] E. Mendes, P. Rodriguez, V. Freitas, S. Baker, and M. A. Atoui. "Towards improving decision making and estimating the value of decisions in value-based software engineering: the VALUE framework". *SQJ 26(2), 2017*.
[9] A. T. Misirli and A. B. Bener. "Bayesian networks for evidence-based decision-making in software engineering". *IEEE Trans. Softw. Eng.*, 2014.
[10] C. Gómez *et al.* "Towards an Ontology for Strategic Decision Making: The Case of Quality in Rapid Software Development Projects". MREBA@ER 2017.
[11] S. Martínez-Fernández, A. Jedlitschka, L. Guzmán, A. M. Vollmer. "A Quality Model for Actionable Analytics in Rapid Software Development". CoRR abs/1803.09670, 2018.
[12] X. Franch *et al.* "Data-driven requirements engineering in agile projects: the Q-rapids approach". *REW 2017*.
[13] X. Franch *et al.* "Data-Driven Elicitation, Assessment and Documentation of Quality Requirements in Agile Software Development". *CAiSE* 2018.
[14] E. Mendes. "Using knowledge elicitation to improve Web effort estimation: Lessons from six industrial case studies". *ICSE 2012.*
[15] B. Das. "Generating Conditional Probabilities for Bayesian Networks: Easing the Knowledge Acquisition Problem," *CoRR*, 2004.
[16] E. A. Drost. "Validity and Reliability in Social Science Research". *Educ. Res. Perspect.*, 2011.
[17] E. Mendes, Practitioner's knowledge representation: A pathway to improve software effort estimation. Springer, 2014.
[18] J. Pitchforth and K. Mengersen, "A proposed validation framework for expert elicited Bayesian Networks," *Expert Syst. Appl.*, 40(1), 2013.
[19] H. Washizaki and K. Honda Fukazawa. "Predicting release time for open source software based on the generalized software reliability model". AGILE, 2015.