A Method to Estimate Software Strategic Indicators in Software Development: An Industrial Application

Martí Manzano[a,*], Claudia Ayala[a], Cristina Gómez[a], Antonin Abherve[b], Xavier Franch[a], Emilia Mendes[c]

[a] Universitat Politècnica de Catalunya, Barcelona, Spain
[b] Softeam Group, Paris, France
[c] Blekinge Institute of Technology, Karlskrona, Sweden

Abstract

Context: Exploiting software development related data from software-development intensive organizations to support tactical and strategic decision making is a challenge. Combining data-driven approaches with expert knowledge has been highlighted as a sensible approach for leading software-development intensive organizations to rightful decision-making improvements. However, most of the existing proposals lack of important aspects that hinders their industrial uptake such as: customization guidelines to fit the proposals to other contexts and/or automatic or semi-automatic data collection support for putting them forward in a real organization. As a result, existing proposals are rarely used in the industrial context.
Objective: Support software-development intensive organizations with guidance and tools for exploiting software development related data and expert knowledge to improve their decision making.
Method: We have developed a novel method called SESSI (Specification and Estimation of Software Strategic Indicators) that was articulated from industrial experiences with Nokia, Bittium, Softeam and iTTi in the context of Q-Rapids European project following a design science approach. As part of the industrial summative evaluation, we performed the first case study focused on the application of the method.
Results: We detail the phases and steps of the SESSI method and illustrate its application in the development of ModelioNG, a software product of Modeliosoft development firm.
Conclusion: The application of the SESSI method in the context of ModelioNG case study has provided us with useful feedback to improve the method and has evidenced that applying the method was feasible in this context.

Keywords: Software Strategic Indicator, Decision-making support, Estimation model, Design Science, Case Study

* Contact author: mmanzano@essi.upc.edu

1. Introduction

Software-development intensive organizations (which we define as public or private organizations extensively developing software) produce large amounts of data related to their processes and products from the use of their corporate tools (e.g., continuous inspection tools, continuous integration tools, project management tools, and issue trackers). Although considerable efforts have been done to exploit software development related data for decision-making support, further research is required for automating and generalizing actionable analytics from such data to procure meaningful information [1].

Decision-making processes in software-development intensive organizations range from strategic and tactical to operational decisions [2,3]. Operational decisions are taken by software development teams on a daily basis to deal with features completion [3]. Strategic and tactical decisions are taken by management-related roles such as product owners or project managers to reach the organizations' business goals and objectives or to manage project resources, respectively [2,3].

Project and product management tools (e.g., SonarQube[1], Kiuwan[2]) are commonly used by development teams to exploit software development data to support operational decisions, and in some cases, tactical decisions [4,5]. For example, SonarQube can provide continuous code quality assessment based on static code analysis and software metrics (e.g. code smells, number of bugs and vulnerabilities). General approaches and business intelligence solutions (e.g., Tableau[3], Power BI[4]) exist to extract insights from corporate data for supporting strategic and tactical decisions. However, specific support for assisting tactical and strategic decisions in connection with operational ones in software-development intensive organizations is still scarce, as highlighted by previous works [5,6].

There have been several proposals contributing to the specification and assessment of software-related indicators [7–10] as a mechanism to fill in this gap. However, there are several aspects that still remain open. Although data-driven approaches to support decision making have been put forward in Software Engineering [1], they are endangered by the fact that the use of inappropriate solutions or tools might inundate them with irrelevant information that can negatively influence decisions [2,6]. From the practical point of view, combining data-driven approaches with expert knowledge has been highlighted as a sensible approach for leading software-development intensive organizations to rightful decision-making improvements [6,11]. Although there exist data- and expert-driven proposals, most of them hardly consider their customization to other contexts as well as the automatic or semi-automatic data collection support for putting them forward in an organization. As a result, existing proposals are rarely used in the industrial practice.

[1] https://www.sonarqube.org
[2] https://www.kiuwan.com
[3] https://www.tableau.com
[4] https://powerbi.microsoft.com

In this context, we present a novel method called SESSI (Specification and Estimation of Software Strategic Indicators) that was articulated from industrial experiences with Nokia, Bittium, Softeam and iTTi in the context of Q-Rapids European project following a design science approach. The goal of the method is to support software-development intensive organizations with guidance and tools for exploiting software development related data and expert knowledge. Specifically, the method supports the specification and assessment of software strategic indicators (SSIs). SSIs refer to measurable aspects, such as development process performance, software quality and on-time delivery, that a software-development intensive organization considers important for its strategic and tactical decision-making processes.

The method has two phases: a) SSI specification and development of data collectors for feeding such SSI with data automatically collected from available organizational repositories. b) Construction of the SSI estimation model based on Bayesian networks [12]. The resulting estimation model allows monitoring the progress of software development (what is especially valuable in agile software development iterations) and gives support to make the right decisions for improving the development processes or products.

The contribution of this paper is twofold: 1) To provide an overview of the SESSI method. 2) To illustrate the application of the method in the context of ModelioNG, a software product of Modeliosoft, a software development firm.

The remainder of this paper is organized as follows: Section 2 describes the research context in which this work has been carried out. Section 3 provides an overview of the related work. Section 4 gives a brief background on Bayesian networks, as they are used to build the estimation models proposed by the SESSI method. Section 5 provides an overview of the SESSI method detailing its phases and steps. Section 6 illustrates the application of the method in Modeliosoft, tackled as a case study and discusses relevant threats to validity that promote the correct interpretation of the case study results. Finally, Section 7 summarizes the lessons learnt on the application of the method and future work.

2. Research Context

The SESSI method was devised in the context of Q-Rapids [13], a joint European project composed by a multidisciplinary academic team from three different institutions and four industrial partners with different profiles and sizes: Nokia, Bittium, Softeam and iTTi. The method was articulated from industrial experiences with these industrial partners in a design science fashion, following the design cycle described by Wieringa [14]. Figure 1 shows an overview of the design cycles followed.

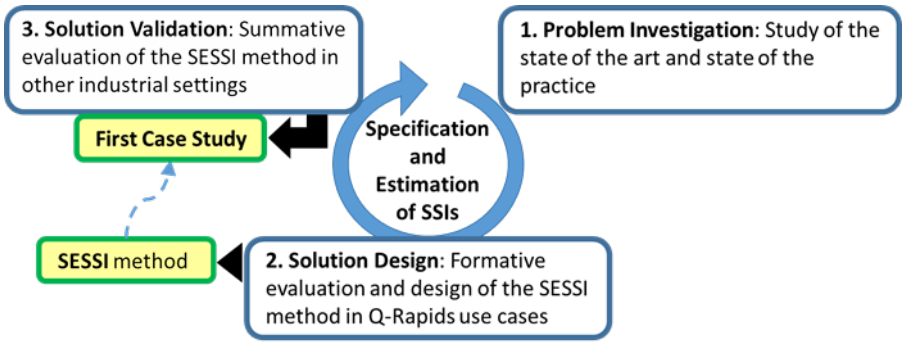


Figure 1 Stages of the design science cycle, inspired on [14]**, applied to conceive the SESSI method**

The stages of the design science cycle applied to conceive the SESSI method were the following:

- **Stage 1. Problem Investigation** aimed to understand decision-making problems and needs related to software-development intensive organizations from the literature and the industrial practice.
- **Stage 2. Solution Design** focused on devising the SESSI method following an action-research like approach [15] in the context of the four Q-Rapids industrial partners. We applied our solution attempts in these partners, and we shaped the SESSI method based on the lessons learnt and insights gained from them. Preliminary formative results of the application of our solution attempts have been published elsewhere [16,17].
- **Stage 3. Solution Validation** refers to the summative evaluation of the SESSI method in other industrial settings than those it was conceived. The goal of the industrial summative evaluation plan of SESSI is: to gain insights on the application of the method in different industrial contexts and to assess its potential worthiness.

In this paper we present a summary of the results obtained from Stage 1 and focus on detailing the SESSI method resulting from the Stage 2 and the initial results of the Stage 3. Specifically, we report the phases and steps of the SESSI method and illustrate its application in the context of ModelioNG case study, a software development project of Modeliosoft.

3. Related Work

One of the conclusions we drew from the Problem Investigation stage (see Figure 1) is that software-related indicators are a mechanism to support decision making in software-development intensive organizations. Although there exists a plethora of works presenting different types of software-related indicators based on low-level software metrics and measures (e.g. [18–22], see Table 1), indicators supporting

strategic and tactical decision-making in Software Engineering (SE) are still scarce [6].

On the one hand, specifying software-related indicators is not an easy task considering their contextual nature [23] and the non-deterministic nature of decision-making in software development [2,3]. On the other hand, assessing software-related indicators to enable monitoring and further data analytics (such as what-if analysis and prediction) requires the elaboration of assessment models. Thus, we characterized the literature according to some relevant aspects related to the specification and assessment of software-related indicators: 1) Customization to different contexts and/or organizations, 2) Data/Expert-Driven approaches to build assessment models for software-related indicators, 3) Tool support for assessment model creation and deployment.

Table I summarizes the most relevant aspects of the reviewed works and classifies them according to the indicator name and the characterization explained above.

Table 1 Related work Summarization

Ref.	Indicator name	Characterization aspects		
		Customization[1]	Data/Expert-Driven[2]	Tool support[3]
[24]	Risky areas of software code in Agile/Lean software development	SC	E	G*
[25]	Cost, schedule deviation, cost, satisfaction, productivity	S	E	N
[26]	Product quality, team Productivity in Continuous Integration	S	D	N
[18]	Pre-release defect density	S	D	N
[19]	Maintenance inflow, lead-time, workload	C	ND	N
[27]	Release readiness	SC	E	G*
[28]	Software readiness factors	SC	E	N
[20]	Flow measures in lean software development	C	B	N
[29]	(Method) Example indicator: Probability of residual defects	C	B	N
[30]	Software early defects	SC	E	N
[31]	(Framework for measurement systems) Example indicator: Project status	SC	E	B*
[32]	Bottlenecks in Agile and Lean software development	S	E	G*
[10]	Teamwork quality of agile teams	C	E	N
[9]	Process problems in software development	C	E	M
[33]	(Framework) Scrum-based processes	C	E	N
[34,35]	Value of decisions in software development	C	B	M
[7]	Delivery capability	SC	D	M
[36]	Cloud services quality aspects	SC	E	N
[37]	Maintainability	S	E	N
[38]	Maintainability, other quality characteristics	SC	E	N
[22]	Software reliability	C	B	N
[21]	Software quality	C	E	B
[39]	Software quality	C	E	N

[40]	Software quality	SC	E	B
[41]	Software quality	S	D	N
[42]	Software quality	SC	D	M*
[43]	(Method). Enterprise architecture models	C	E	M
[44]	(Method). Example indicator: enterprise system modifiability	C	E	M
[45]	Enterprise system modifiability	C	E	M
[46]	(Method). Framework for enterprise architecture analysis	C	E	M
[47]	Service response time for service-oriented architectures	C	E	M
[48]	Application usage	C	E	M

[1] **S**: Specific for an organization, customization not considered; **SC:** Specific for an organization, customization planned but not detailed; **C**: Customizable; **ND**: Not detailed

[2] **D**: Data-driven; **E**: Expert-driven; **B**: Both; **ND**: Not detailed

[3] **M**: Building assessment model tool support; **G**: Data collection tool support; **B**: Both, **N**: None; *: Not shared

Customization to different contexts and/or organizations. There is plenty of literature on indicators and metrics aimed to specify meaningful aspects of software development products and processes.

One specific type of instrument that has been widely used in the literature to specify and create assessment models for software-related indicators refers to Quality Models (QMs) [39–42]. Although useful for specification, most of these works are academic and provide too abstract procedures to be operational [49]. Yan et al.'s [50] systematic mapping study provides an overview of works proposing assessment models based on QMs.

We have found studies proposing ad-hoc methods for specifying and assessing software-related indicators. However, they do not provide details on how to adapt their approach for other contexts or other indicators. For instance, Choetkiertikul et al. [7] propose a predefined indicator for *delivery capability* using data from open source projects. Bakota et al. [38] introduce a *maintainability* indicator; Staron et al. [27] propose a *release readiness* indicator; Fenton et al. [30] present a *software early defects* indicator; or the *relative risk* indicator defined by Antinyan et al. [24].

Even though these proposed indicators may be suitable for the specific organizations or contexts they were built on, their specification and assessment cannot be realistically expected to be universal and reusable, since each organization may have its own intricacies yielding to different definitions [23]. Hence, the lack of smooth customization of the indicators proposed by the mentioned studies can restrain their adoption in other organizations. Only few works provide guidance support for this customization, for instance [9,10,34,39].

It is worth mentioning other works in other higher-level domains than software development such as enterprise architectures that also deal with the assessment of some indicators. For instance, Johnson et al. [43] propose a framework and a formal language to create assessment models to analyze several properties/indicators of enterprise architectures, as for instance *information security* and *interoperability*. This framework has been adapted and instantiated to assess aspects of enterprise systems such as *security* [51], *performance* [47], *usage* [48] and *modifiability* [45]. Although these works offer an interesting background for specifying and assessing software-related indicators, the amount and nature of their input data is different than in software development.

Data/Expert-driven approaches to build assessment models for software-related indicators. The creation of suitable assessment models for software-related indicators requires input data from software development related repositories (collected through software development related tools) and/or expert knowledge. Some of the reviewed works only rely on software development collected data. For example, Choetkiertikul et al. [7] collect software development data for building a estimation model for the *delivery capability* indicator and Vasilescu et al. [26] exploit continuous integration data to assess *software quality* and *productivity*.

Other works consider the participation of experts to build assessment models for software-related indicators, that is, they are based on expert-driven approaches. However, the majority of these works are either simplistic, limiting the experts' participation to providing weights for weighted averaged indicators, e.g. [40]; or overwhelming, requiring large amounts of parameters, e.g. [21,36]; or even too complex, requiring non-trivial statistical knowledge, such as [10]. All these aspects can hinder the adoption of these proposals by managerial decision makers.

Only few works consider the combination of data- and expert-driven approaches to build assessment models for software-related indicators, for example Mendes et al. framework [34] for *value* assessment, or the approach to define a *software reliability* indicator by Fenton et al. [22].

From the practical point of view, combining data-driven approaches with expert knowledge has been highlighted as a sensible approach for leading software-development intensive organizations to rightful decision-making improvements [6,11].

Tool support for assessment model creation and deployment. Building assessment models for software-related indicators is a demanding process that requires supporting tools. On the one hand, we found that most proposals do not provide tools for supporting the assessment model construction process. On the other hand, most proposals also fail on providing data collectors that allow feeding the assessment model for enabling its monitoring capabilities. This makes the industrial uptake of these proposals hard.

Some works, such as [7,9,21,34,46], provide software tools to assist some specific tasks for the assessment model creation.

Some popular software tools such as source code management (e.g. Git, Subversion), source code analysis (e.g. Sonar, StyleCop) or project management (e.g. Redmine, Jira), among others, automatically collect data to provide specific metrics for software development lifecycle. These metrics can be visually consulted by their included dashboards (e.g. SonarCloud, Jira). However, these tools do not support the connection between low-level metrics and assessment models. Very few works have developed specific software artifacts to support such connection, for instance the works proposed by Wagner et al. [21] and Staron et al. [27,31,32]. Most of these proposals do not further detail nor share their developed software artifacts. A relevant exception is [21], that provides reusable software artifacts to enable the *software quality* indicator monitoring.

To get insights about the state of the practice in this area, we performed a survey in the context of the industrial partners of the Q-Rapids project. See further information about such survey in [52]. The main observations were:

- None of the organizations used any method or approach to assist them with the specification of software-related indicators to support their decision-making processes. Instead, they stated that such indicators were implicit in the head of the decision makers.
- Decision makers confirmed that they did not explicitly specify such indicators but took their decisions based on the information provided from their tools in use (e.g., Mantis, SonarQube, Redmine or Jenkins) and from their own experience and intuition. A relevant problem that some of them emphasized regarding this approach was the dependency on the decision maker's experience. That is, if the person in charge of assessing the indicators is not available, the indicators can be incorrectly assessed or not assessed at all.
- Regarding the aggregation capabilities provided by some of their tools in use, they stated that such functionalities were not capable enough, as they lacked a mechanism to aggregate the heterogeneous information coming from their tools in use to SSIs, usable for decision-making endeavors.

All in all, we found that most of the existing proposals do not fully address the relevant aspects related to the specification and assessment of software-related indicators. Therefore, we propose a tooled and guided approach that allows software-development intensive organizations to specify SSIs and build estimation models for their own needs, considering expert knowledge and data-driven capabilities to support strategic and tactical decision-making.

4. Background: Bayesian networks

Bayesian networks use probability theory and graph theory to construct probabilistic inference and reasoning models. The obtained models are graphical and represent cause-effect relationships. Bayesian networks have been used across diverse domains such as weather forecasting, medical diagnosis [53] and software engineering [29] as the basis for constructing assessment models capable of estimating the value of variables or events through probabilistic inference and different kinds of simulation, also known as "what-if analysis".

The application of Bayesian networks in software engineering has been extensive, ranging from performing data analysis in empirical research [54], estimating defects [30], reliability issues [22], software quality [55], software development effort [56,57], and the assessment of several properties for enterprise architectures [43]. In a similar way to these works, the method proposed in this paper also uses Bayesian networks as the basis of the SSI estimation model construction as it will be detailed in Section 5.

Bayesian networks perform inferences using Bayesian probability [42] and are represented by Directed Acyclic Graphs (DAG), whose nodes represent variables or events that can be continuous or discrete, and for which their potential values are known or hypothesized. Edges represent conditional dependencies. Nodes that are not connected (i.e., no path connects one node to another in either way in the DAG) represent variables that are conditionally independent of each other. Each node of the DAG has associated a probability distribution, conditionally dependent on the

impacting nodes, when any. For discrete nodes, probability distributions are represented by Conditional Probability Tables (CPTs). CPTs specify the probabilities of each node being in each of its specified discrete states for all the combinations of the states of the parent nodes (if any), hence allowing quantifying the causal relations and the uncertainty among the nodes. Figure 2 shows a well-known example of a simple Bayesian network with variables modelling the "wet grass" event and its causes [58]. The figure contains the DAG of the Bayesian network (nodes and edges), and its CPTs, representing the probabilities of each node being in each of its *T* (for True) or *F* (for False) states (presence or absence), conditionally on the states of its parent nodes, when any.

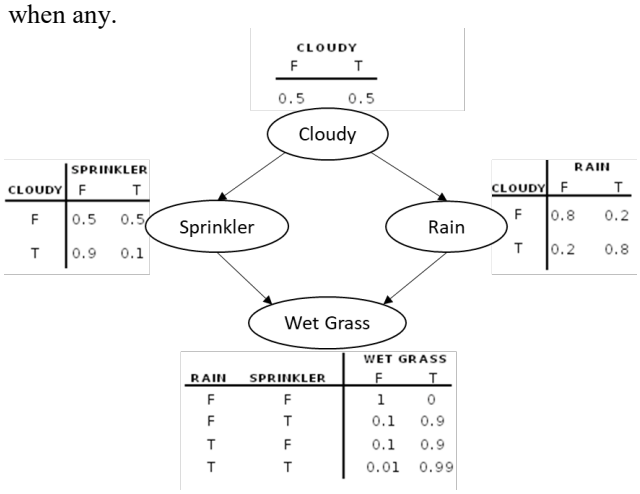


Figure 2 Example of a *Wet Grass* Bayesian network model. From [58]

For simple cases, Bayesian networks can be created and filled by a domain expert. However, there are situations in which building Bayesian networks is too complex or cumbersome for humans, since the manual creation of CPTs can be prohibitive in terms of time, as domain experts would have to provide hundreds or thousands of probabilities. In these situations, exploiting available historical data and/or using several techniques such as Weighted Sum Algorithm (WSA), proposed by Das [59] or the Ranked Nodes, originally proposed by Fenton et al. [60], can reduce the effort of filling in the CPTs and thus the model building time.

5. SESSI: A Method to Specify and Estimate SSIs in Software Development

In this section we detail the SESSI (Specification and Estimation of Software Strategic Indicators) method. As mentioned before, SESSI was devised from an action-research like cycle in the context of the industrial partners of the Q-Rapids project.

SESSI is aimed to support software-development intensive organizations to specify SSIs based on their available data and to construct SSIs estimation models for supporting decision-making with SSIs monitoring and what-if analysis.

SESSI comprises two main phases involving one or more domain experts (i.e., people from the organization who are familiar with their data sources and their decision-making processes). The first phase provides support to specify an SSI and its associated data collectors (i.e., software artifacts allowing automatic data collection from the data sources of the organization). The second phase aims to build the SSI estimation model using a combination of data collected through data collectors and expert knowledge. The resulting model provides SSI estimations, thus enabling SSI monitoring and supporting decision-making.

As mentioned in section 2, the SESSI method was conceived from dealing with the industrial needs of the Q-Rapids industrial partners and devising solution attempts in their contexts. Preliminary formative results of the specification and estimation model construction phases are provided in [16] and [17] respectively. The method presented here extends these preliminary results by consolidating both phases and providing not just results in the context of a single partner but consolidating the experiences from all of them, broadening the applicability of the method. Furthermore, it extends the method with supporting tools and additional techniques to support the estimation model creation.

In the following subsections, we detail each phase of the method.

5.1 Phase 1: SSI Specification

The specification of an SSI reconciles the informational needs for decision making of the interested organization and the software-related data available in its repositories (e.g., continuous inspection tools, continuous integration tools, project management tools and issue trackers). To enable the automatic collection of these data as input to estimate the SSI, we suggest the development of data collectors.

To determine the information required for the specification of an SSI, several elicitation activities can be performed by the domain experts of the organization. Although SESSI does not propose specific techniques for this purpose, some of the techniques we have used in previous industrial cases may serve as a useful reference [16].

For illustrative purposes, in the remaining section, we will use as a reference example the *Product Quality* SSI from one of the Q-Rapids industrial partners. Figure 3 shows the complete specification of the *Product Quality* SSI. The data collectors developed for this use case are available at GitHub [61].

To operationalize the specification of an SSI, the SESSI method suggests the gathering of the following assets:

- **Textual definition.** Captures the SSI rationale according to the strategic needs and the available data from the organization. For the *Product Quality* indicator, its textual description is "*Degree of fulfilment of the collection of features and characteristics of a product according to the given requirements"* (see Figure 3).

- **Hierarchical decomposition for specifying the SSI meaning**. SSIs should be elaborated using a hierarchical approach based on the QM structure proposed in [23,62,63]. We chose such structure as it has proved to contribute to ease the specification and understandability of software development related concepts. The three levels of the hierarchy are as follows[5]:
    - **Metrics** refer to specific attributes of an entity or a process that may be measured. For example, *Code Complexity* (see Figure 3) is a metric that refers to the complexity attribute of the code entity. Metrics are directly computed from data available in the repositories.
    - **Factors** refer to an aggregation of metrics that constitute a property, which is present in a software product or process, representing the intermediate level of the hierarchy. For example, *Code Quality* (see Figure 3) is a factor that aggregates *Code Complexity* and *Percentage of Duplicate Code* metrics.
    - **SSIs** refer to aspects or characteristics related to software products and/or development processes that an organization considers important for its tactical and strategic decision-making processes. They are aggregated from factors. For example, an organization may define its *Product Quality* SSI (see Figure 3) as a key aspect of a software product to be considered for decision making.
- **Data collectors** refer to software artifacts that should be developed to automatically collect data from organizational repositories to compute the metrics. Each organization can develop its own data collectors, reuse or customize them from other projects. Some open source data collectors for tools like Jira, OpenProject or SonarQube were developed by the partners of the Q-Rapids project as modular software artifacts integrated in a Java tool available in [61].

[5] The layout is aligned with the usual Bayesian network DAGs layouts in which the response variable/s yield at the bottom part and probabilities propagate top-bottom.

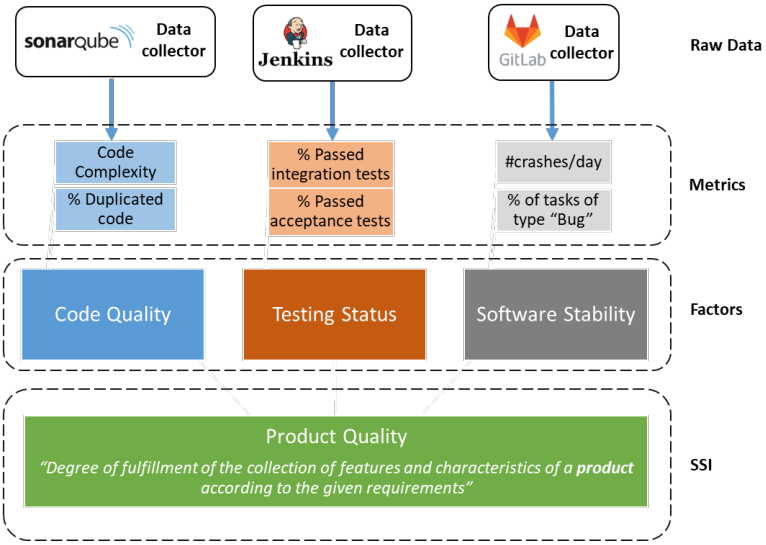


Figure 3 Example of a *Product Quality* SSI specification

5.2 Phase 2: SSI Estimation Model Building

The purpose of this phase is to build an estimation model for the SSI specified in the previous phase. It is done by exploiting expert knowledge and historical data gathered through data collectors.

The estimation model is built based on Bayesian networks. We chose Bayesian networks as they are a flexible and well-known instrument highly used in software engineering to build estimation models under uncertainty conditions [34,35,54]. The flexible nature of Bayesian networks and the extensive theory and software tools available for their construction and management were important factors that we considered positive to contribute to the industrial uptake of the SESSI method. The availability of several software tools (some of them as open source software) such as Netica® [64] or unBBayes [65] have allowed us the creation and management of the estimation models through graphical interfaces.

The steps to build the SSI estimation model are (see Figure 4):

1) **Data Splitting:** Generation of training and validation sets from the historical data.
2) **DAG Specification:** Definition of the DAG representing the Bayesian network structure of the SSI.
3) **CPTs Specification:** Specification of the CPT for each node of the previously defined DAG.
4) **Estimation Model Generation:** Generation of the complete Bayesian network model for the SSI based on the output of the two previous steps.
5) **Estimation Model Validation:** Validation/recalibration of the resulting SSI estimation model.

6) **Deployment and use of the SSI Estimation Model:** Deployment of the model to enable SSI monitoring.

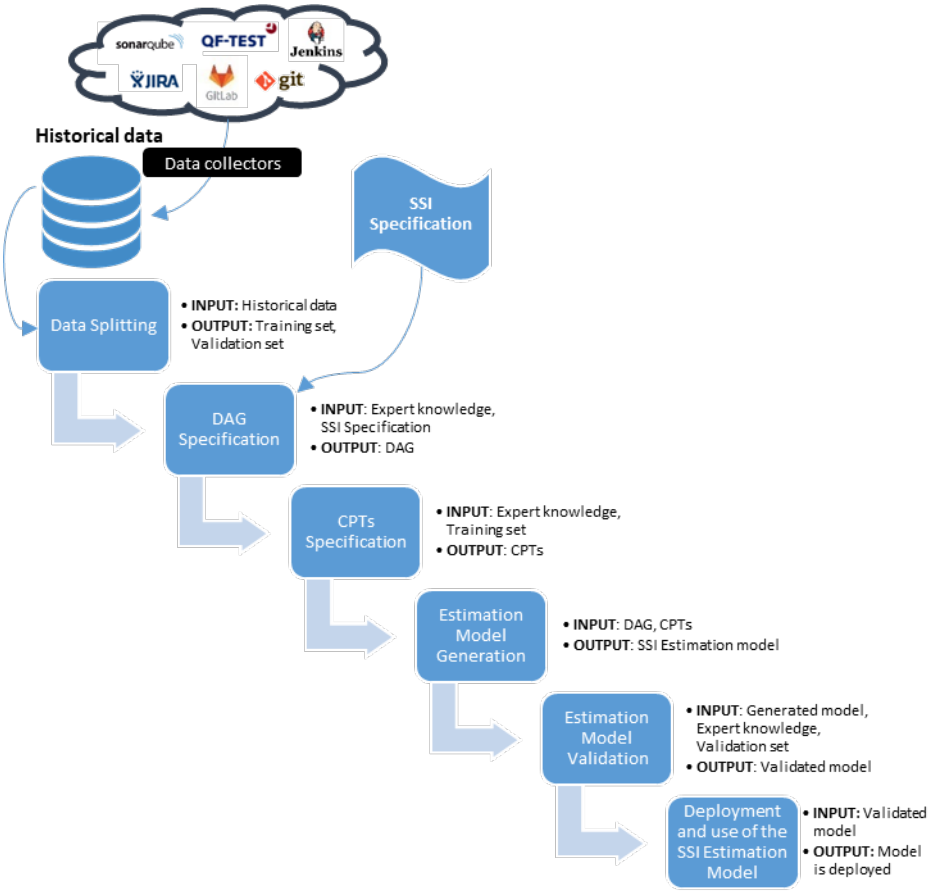


Figure 4 Overview of SESSI steps to build the SSI estimation model

Each of these steps is detailed as follows.

Step 1-Data Splitting

This step has the objective of generating training and validation sets from historical data. Splitting the historical data allows using a subset of that data to build the model (as training set) and the remaining data for validation (as validation set). Common training/validation splits are 70%-30% or 80%-20% [66]. It is important to remark that the accuracy of the resulting model depends on the amount and quality of the available data.

Step 2-DAG Specification

The initial input of this step is the hierarchical structure of the SSI based on factors and metrics obtained in the specification phase, which is used for defining the DAG. Thus, the DAG is composed by the SSI node, stated as the leaf node and being the factors its parent nodes, and the metrics, the root nodes. Figure 5 shows the DAG of the Bayesian network built for the *Product Quality* SSI introduced in Figure 3. Metrics have directed edges to factors and factors at their turn to the SSI node.

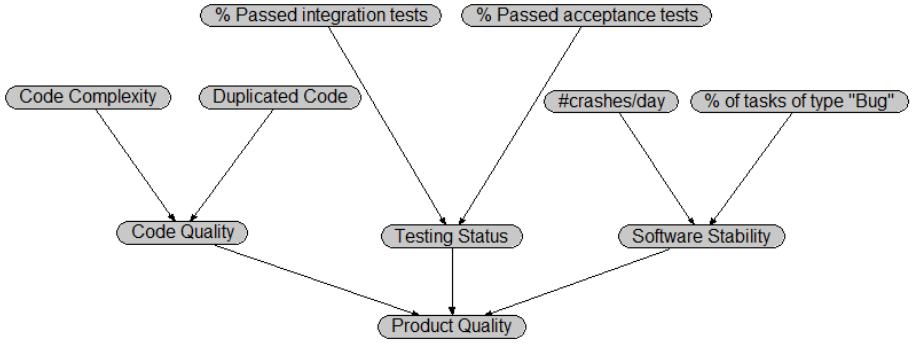


Figure 5 DAG of a Bayesian network for the *Product Quality* SSI

To specify the potential states for each node of the Bayesian network, expert knowledge is required. Therefore, domain experts should provide the potential states for each node (e.g., Low, Medium and High). For metric nodes (i.e., nodes whose values are continuously computed and directly extracted from data repositories through data collectors), domain experts should additionally specify the binning intervals for their values. For its use in the Bayesian network, these intervals will serve as discretization functions to transform the continuous values of metrics into the states defined for those metrics. For the reference example, the *Code Complexity* node can have the states "Low", "Medium", and "High", and the binning intervals are defined as [0-0.5), [0.5-0.75) and [0.75-1].

Tool support: To help domain experts with the specification of the binning intervals, we implemented two existing unsupervised binning methods, namely Equal-Width and Equal-Frequency binning [67] into a software tool [68]. This tool aims to support the specification of the numeric intervals for the states. The Equal-Width binning returns *n* intervals of equal size, and the Equal-Frequency divides the data into *n* intervals, each one having approximately the same number of values. The intervals obtained with the tool can be used as a starting point to be refined by domain experts according to their needs.

Step 3-CPTs Specification

This step aims to fill in the CPTs in order to specify the probability function over each node of the DAG. The process is performed differently according to the type of the node, but it is supported in every case using historical data.

To fill in the CPTs for root nodes, the probabilities of their states are directly computed from the historical data through frequency quantification.

CPTs for child nodes are built based on the combinations of the states of their parent nodes. As the CPTs for these nodes grow exponentially depending on the number of parent nodes and their potential states, it might be not feasible for domain experts to manually fill in the probabilities for each combination. Therefore, for these cases, we consider the use of the WSA technique [59] to ease the quantification of probabilities. This technique is based on expert knowledge and takes as input a set of compatible configurations for the node being quantified, their resulting probabilities and the relative weights of its parent nodes. Each *compatible configuration* is composed of a combination of the states of the parent nodes that are more meaningful and more likely to happen according to the domain expertise. From that input, the WSA technique infers the complete CPT, thus reducing the number of probabilities to be elicited from domain experts, a widely acknowledged problem in the literature [69]. A detailed explanation of the WSA technique is provided in [34,59].

Figure 6 shows the application of the WSA technique for the *Code Quality* node of the *Product Quality* SSI. At the top, some *compatible configurations* are shown, along with the provided probabilities for each *Code Quality* state and the relative weights. At the bottom, the CPT has been automatically filled by the WSA technique.

Tool support: We implemented a set of tools to support the semi-automatic generation of CPTs. For root nodes, the tool *getFrequencyQuantification* [70] computes the CPTs through frequency quantification over the historical data, using the provided states and its corresponding binning intervals (from Step 2). For child nodes, we implemented a version of the WSA technique (as we could not find any public available implementation). Our developed version is open source and available at GitHub [71], as well as a supporting tool *getCompatibleConfigurations* [72], automating the computation of *compatible configurations* required by the WSA, using historical data. The use of these tools significantly reduces the necessary effort to specify the required input data.

Weight = 0,3 Weight = 0,7

Code Complexity	Duplicated Code	Code Quality: 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	Low	40	55	5	0	0
Medium	Medium	10	40	50	0	0
High	High	0	0	20	75	5
Very High	Very High	0	0	0	10	90

Code Complexity	Duplicated Code	Code Quality: Very Low	Low	Medium	High	Very High
Very Low	Very Low	100	0	0	0	0
Very Low	Low	58	38.5	3.5	0	0
Very Low	Medium	30	40	30	0	0
Very Low	High	25	30	35	10	0
Very Low	Very High	15	30	40	15	0
Low	Very Low	95	5	0	0	0
Low	Low	40	60	0	0	0
Low	Medium	28.5	40	31.5	0	0
Low	High	1.5	20	43.5	35	0
Low	Very High	0.5	10.5	50	38	1
Medium	Very Low	74.25	17.5	8.25	0	0
Medium	Low	40	55	5	0	0
Medium	Medium	10	40	50	0	0
Medium	High	6	15.75	52.5	22.25	3.5
Medium	Very High	6	7	8.25	28.75	50
High	Very Low	68.25	1.75	6	22.5	1.5
High	Low	28	38.5	9.5	22.5	1.5
High	Medium	0	35	41	22.5	1.5
High	High	0	0	20	75	5
High	Very High	0	0	6	29.5	64.5
Very High	Very Low	68.25	1.75	0	3	27
Very High	Low	28	38.5	3.5	3	27
Very High	Medium	0	35	35	3	27
Very High	High	0	0	14	55.5	30.5
Very High	Very High	0	0	0	10	90

Figure 6 *Code Quality* input data required by the WSA (top) and inferred CPT after the WSA application (bottom)

Step 4 – Estimation Model Generation

Once the CPT for every node has been specified, the Bayesian network estimation model can be built using the DAG and CPTs from steps 2 and 3. To build the Bayesian network, in the context of this work, we used Netica® [64] and unBBayes [65].

An example of the resulting Bayesian network for the *Product Quality* SSI is shown in Figure 7. The figure shows the DAG structure and the probabilities of the nodes, computed from their individual CPTs and the CPTs of their parent nodes (if any). We have attached the two complete CPTs for *% Passed Integration Tests* and *% of tasks of type "Bug"* metric nodes (which coincide with their probabilities as they are root nodes), and the partial CPT for the *Software Stability* factor node.

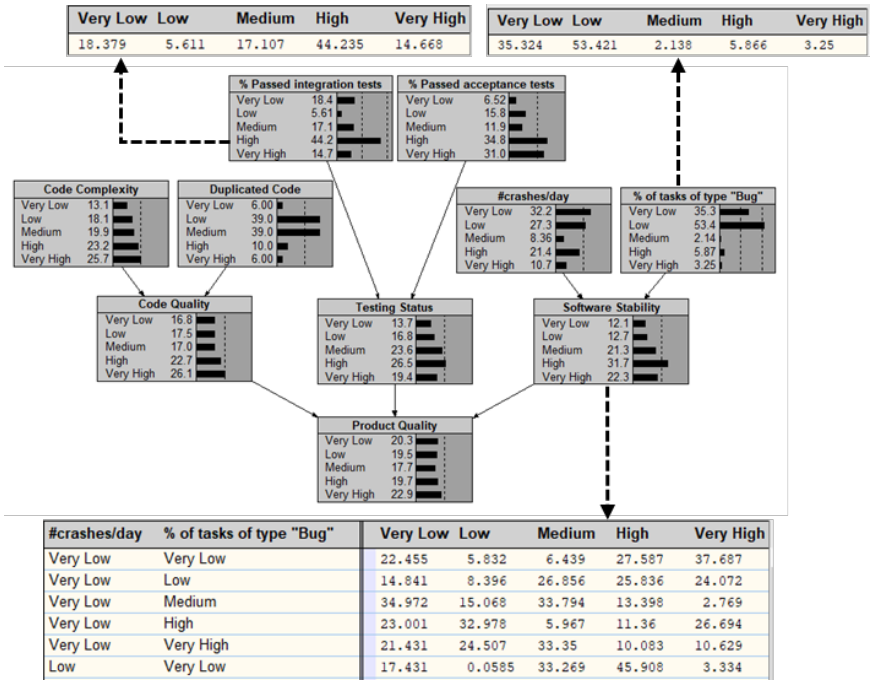


Figure 7 Example of a Bayesian network for the *Product Quality* SSI

Step 5 – Estimation Model Validation

This step aims to validate and/or recalibrate the model resulting from the previous step. With this purpose, other works have suggested two validation methods: Model Walkthrough and Outcome Adequacy [34,56]. The SESSI method also applies these validation methods, but in contrast to the mentioned works, SESSI conducts the validation for all the nodes of the model, except for the metric nodes (as their CPTs are directly quantified from the training data and refined by the domain experts). This is particularly relevant to ensure the trustworthiness of the entire model, especially because of the use of the WSA semi-automatic approach for supporting the construction of the CPTs for child nodes.

The application of the two validation methods is sequential and based on different strengths of evidence:

1) Model Walkthrough. It is the first validation and it aims at assessing the subjective accuracy of the estimation model, recalibrating it when necessary. This validation consists in the manual creation of hypothetical scenarios by the domain experts, and their perceived most probable resulting state for the node under validation. Each scenario is composed of a combination of states for the parent nodes of the node under validation. These scenarios are manually introduced into the Bayesian network as a "what-if" analysis. It is done to compare the resulting state perceived by the domain experts to the most probable state resulting from the

estimation model output, after introducing the scenario. If there is a mismatch between these states, the estimation model should be recalibrated by tuning the corresponding CPT and proceeding with the validation until mismatches are amended. The number of hypothetical scenarios to design in this validation can influence the accuracy of the model. However, the recommended number greatly depends on the number of parent nodes and their potential states.

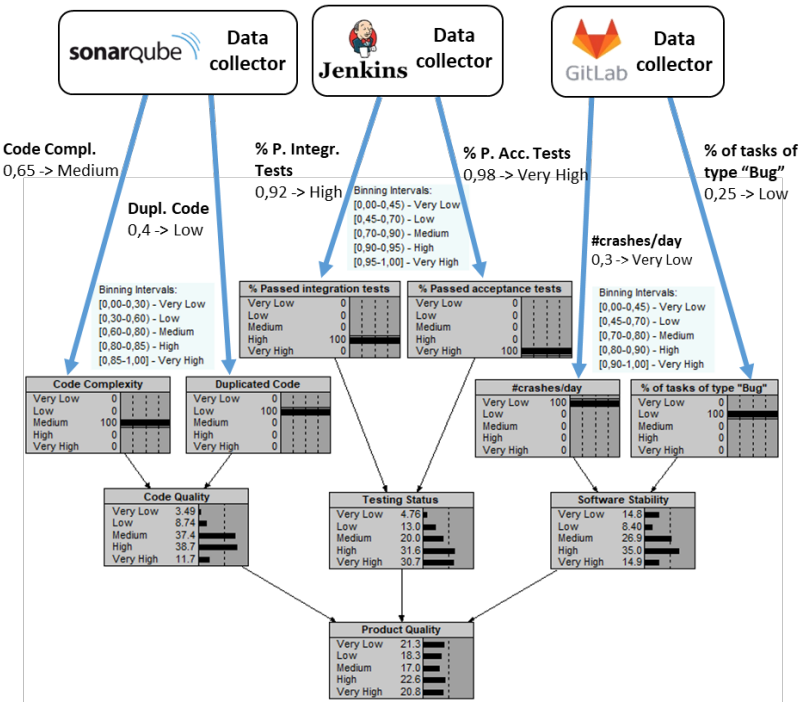
2) Outcome Adequacy. The goal of this second validation is to determine the validity of the Bayesian network with real scenarios from the validation dataset. It is done by comparing the judgements provided by the domain experts with the resulting outputs of the estimation model. Such real scenarios consist in combinations of states for the parent nodes of the node under validation, directly present in the validation dataset. For each scenario, the domain experts provide the expected state for the node under validation, based on what really happened in such scenario. Hence, the main difference with the previous validation stems from the premise that this validation stands on real scenarios instead of hypothetical ones, thus allowing domain experts to reason on the appropriateness of the resulting state provided by them and by the estimation model. The size of the validation dataset will impact on the number of real scenarios that will arise. If this number is too large, it might be needed to select a subset of them to conduct this validation.

Step 6 – Deployment and use of the SSI estimation model

Once the Bayesian network model has been built and validated, it can be deployed and connected to the data collectors to provide automatic and periodical SSI estimations. Netica® [64] or unBBayes [65] APIs can be used to connect the Bayesian network model with their data collectors. It is up to the organization to determine the architectural way to connect the model to the data collectors. One possibility is, for example, having a software orchestrator in the middle that feeds the SSI estimation model with the periodically collected metrics, then providing the output estimations through a dashboard.

Figure 8 shows an example of the *Product Quality* SSI estimation model connected to the data collectors, which automatically compute the metrics. The data collectors computed the *Product Quality* associated metrics, which are discretized through the binning intervals into the states of their corresponding nodes. Then, the probabilities were propagated from the metrics down to the *Product Quality* SSI node. In the same example, given a specific scenario composed of some metrics computed from the data sources and entered into the model, the state with highest probability for the SSI is "High", with 22.6% of probability.

Tool support: We developed a software library named *si_assessment* [73] that supports the connection of Bayesian Network models to data collectors. This software library is a wrapper of the unBBayes API, which performs the probabilistic inference through the junction tree algorithm [65]. This artifact can be used either embedded, or through a REST API [74], as it is a Java library which was also developed in the context of the Q-Rapids project.



565 **Figure 8 Product Quality SSI estimation model connected to the data collectors**

6. An Industrial Application of SESSI Method: ModelioNG Case Study

To evaluate the SESSI method in diverse industrial settings we have devised an 570 industrial summative evaluation plan. The objective of such plan is: *"to gain insights on the application of the method in different industrial contexts and to assess its potential worthiness"*. According to this objective, we chose multiple pre-post case studies as empirical approach. Case studies allow us to study in depth the contextual nature of the application of the method in different organizations. Pre-post case 575 studies [75] refer to the study of one research entity at two time points separated by a critical event. A critical event is one that would be expected to impact case observations significantly [76]. The two points of study established in our cases are: a) The application of the SESSI method in order to understand the feasibility of its application in the context of the organization and project being studied, and b) The 580 post-method deployment aimed to assess the worthiness of the SESSI method after using for some time the resulting model and tools from its former application.

To foster industrial participation, we have invited organizations from our industrial collaboration network. The only requirement for them to participate was that they were software-development intensive organizations with an interest on exploiting

their software development related data to improve their decision making. We offered them direct involvement and collaboration of researchers for applying the SESSI method in an action-research fashion [76] and long-term collaboration for performing the post-method deployment study. The selection of the case studies is opportunistic and based on the availability of the organizations and their willingness to participate.

In this section, we provide details of the application of the SESSI method in our first case study. The application of the method in this case study has finished recently and the post-method deployment study has not started yet as more time for the estimation model usage is required before planning it. Nevertheless, we considered crucial to present our current results as they are able to illustrate the feasible application of the method in a concrete industrial setting and can help to foster the industrial uptake of the method.

6.1 Case Study Setting

Softeam Group[6] is a software-development intensive organization with more than 1300 employees, providing "high-quality services and solutions in strategy, consulting, finance, digital, big data, analytics, performance and operations". As participant in the Q-Rapids project, Softeam promoted the participation of their subsidiary firms in this study. Modeliosoft[7], one of its subsidiary development firms, accepted to participate in the study.

Modeliosoft representatives opportunistically selected ModelioNG, a software product currently developed and maintained by them as the unit of analysis for applying the SESSI method. Their principal interest was to enable long-term monitoring for the product. The selection of ModelioNG was mainly based on the chance of having a project in an initial stage so that the specification of an SSI and the data collection during the development process would be less disruptive and feasible for the organization.

To understand ModelioNG setting, we conducted semi-structured interviews to gain insights on the roles, needs, processes, information flows and software development related tools used by ModelioNG.

The software development process in ModelioNG was defined as "close to agile", focused on the development of working software, face-to-face communication and close collaboration with customers. The overall way of working was characterized as follows: upon analysing the market, Modeliosoft team updates an annual roadmap and elaborates strategies to ensure the success of their long-lived product in a competitive environment. The planned features and requests from customers are then developed, and major issues (if any) are addressed. The code is periodically delivered to integration for nightly builds. Before delivering new releases, the team should ensure that the planned features work without blocking issues.

The software tools used by the development team are: 1) OpenProject[8] for project management (backlog management, issues and specification tracking), 2) Mantis[9] for

[6] www.softeamgroup.fr/en

[7] https://www.modelisoft.com/en

[8] https://www.openproject.org/

bug tracking and 3) Jenkins[10] for builds and tests triggering. Figure 9 illustrates the roles and tools used in ModelioNG.

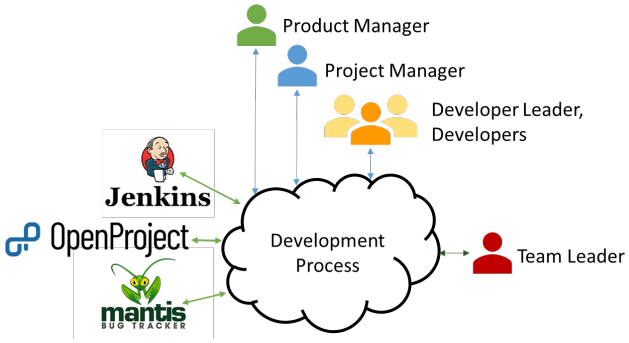


Figure 9 Roles that participated in the case study and tools used for the development of ModelioNG

6.2 Case Study Design

The design of the case study is strongly inspired by the well-known guidelines provided in Yin [77] and Runeson et al. [75]. The key research question leading this case study is: *"How is the application of the SESSI method in the studied project?"*. The main drivers for the case study design were the phases and steps suggested in the SESSI method and the specific characteristics and needs of ModelioNG. The case study design was flexible to deal with daily unexpected issues. Details of issues and/or decisions taken during the case study are detailed in the execution section.

We proposed the participation of 5 researchers (3 of them with previous experience in applying the method in other industrial contexts) that were able to provide hands-on support on the execution of the phases and steps of the method. Modeliosoft agreed and assigned the team leader of ModelioNG as the promotor of the application of the method. He provided us access to the software development related data and led the development of data collectors.

In addition to the team leader, other roles participated as domain experts: the product manager, the project manager, the developer leader and a developer. The team leader assigned the roles to each step of the method, according to their expertise and availability. Our interaction with these domain experts was direct or indirect, depending on their availability.

[9] https://www.mantisbt.org/
[10] https://jenkins.io/

6.3 Data Collection and Data Analysis

Data collection and data analysis were performed according to the phases and steps of the SESSI method described in Section 5. For instance, after studying the ModelioNG context and supporting the specification of the SSI, data collectors were developed for gathering data. This data together with domain experts' knowledge was used as the basis for constructing the estimation model. These activities were based on the guidelines of the SESSI method as it is further explained in the next subsection. We also used individual diaries to record all interactions, issues and relevant observations from the execution of the method. Further details of collected information cannot be provided given non-disclosure agreements with Modeliosoft.

In addition to the SESSI method procedures, we designed a survey based on a questionnaire as a data collection instrument for gathering practitioners' feedback. We used a previously defined questionnaire from the Q-Rapids project [78] as the basis for designing this one. The questionnaire was also piloted and approved by the team leader. As the questionnaire contained mostly closed questions, we processed the gathered data using spreadsheets. For the case of open questions, we planned to use content analysis for analyzing and categorizing all the responses [79]. The questionnaire was designed with the aim of being simple and brief, so it could be filled in 10-15 minutes.

6.4 Case Study Execution

The following subsections detail the execution of ModelioNG case study according to the SESSI method phases.

Phase 1: SSI Specification

The product manager of ModelioNG chose *Product Readiness* as the most appropriate SSI to be tackled by the SESSI method. We studied the literature with the aim of providing some related examples that could serve as starting point for its specification. We found some works specifying/assessing product readiness [27,28], however, none of them was suitable for the particular meaning of product readiness in ModelioNG. Therefore, the product manager and the team leader elicited the list of aspects related to product readiness based on their own knowledge and experience. After some iterations and discussions, the *Product Readiness* SSI specification was stated as shown in Table 2. It was based on 3 factors that at their turn were based on the stated metrics.

Table 2 Specification of *Product Readiness* SSI for the ModelioNG case study

General Definition: *Product Readiness provides high level information on product readiness for the next release. A product "ready to be released" implements the features planned for the release and without critical bugs.*					
Factor Name	**Factor Description**	**Metric Name**	**Metric Description**	**Data Source**	**Metric Definition** (In some cases, the metric definition shown is simplified)
Activities Completion	Represents the status of the completion of activities plan for this release, including development and specification tasks	Specification Task Completion	Represents the fulfilment of the required specification tasks for this release	Open Project	$\frac{total\_specification\_progress}{total\_specification\_planned}$ Where: $total\_specification\_progress = \sum_{i=1}^{N\ WPs\ of\ type\ "Specif."} WP_i spent\ time$ **And**: $total_{specification_{planned}} = \sum_{i=1}^{N\ WPs\ of\ type\ "Specifi."} WP_i spent\ time + + WP_i remainingtime$
		Development Task Completion	Represents the fulfilment status of the required development tasks for this release	Open Project	$\frac{total\_task\_progress}{total\_task\_planned}$ Where: $total\_task\_progress = \sum_{i=1}^{N\ WPs\ of\ type\ "Task"} WP\_i.spent\ time$ $total_{task_{planned}} = \sum_{i=1}^{N\ WPs\ of\ type\ "Task"} WP_i spent\ time + + WP\_i.remainingtime$
Known Remaining Defects	Measures defects/ bugs/crashes that lie outside the major bug category and can be deferred to next releases	Postponed Issues (Closed) Ratio	Ratio of the minor severity closed issues (of type Feature/Trivial/Text/Tweak/ Minor/Usability) with respect to the total number of low severity issues	Mantis	$\frac{no.of\ low\ severity\ closed\ issues}{no.of\ low\ severity\ closed\ issues + no.of\ low\ severity\ open\ issues}$
Product Stability	Measures the status of the operational software quality of the monitored release, considering the presence of major issues and the testing status	Build Stability	Percentage of successful builds with respect to the total of builds triggered in a seven days period	Jenkins	$\frac{successful\ builds}{total\ builds\ triggered}$
		Critical Issues (Closed) Ratio	Ratio of high severity closed issues (of type Crash/Block/Major) with respect to the total number of high severity issues	Mantis	$\frac{no.of\ high\ severity\ closed\ issues}{no.of\ high\ severity\ closed\ issues + no.of\ high\ severity\ open\ issues}$
		Passed Tests Percentage	Percentage of tests passed with respect to the total number of tests ran for the latest build	Jenkins	$\frac{tests\ passed}{total\ tests\ ran}$

Data collectors were developed in order to automatically extract, compute and store the metrics specified in Table 2. In particular, Modeliosoft developed data collectors for Mantis and OpenProject, while for Jenkins, they reused some available data collectors developed in the context of the Q-Rapids project [61]. Data collectors were configured to run once per day for collecting data and computing metrics. The computed values were saved into a database that we called HistoricalNG.

Phase 2: SSI Estimation Model Building

To execute this second phase of the method, we had to make sure of having enough collected data into the HistoricalNG database. ModelioNG team together with our research team decided to perform the second phase of the method to build the estimation model for the *Product Readiness* SSI once we had a period of 3 months of collected data into the HistoricalNG database.

The domain experts that participated in this phase were: the product manager, the project manager, the developer leader, the team leader and a developer. The steps were conducted as follows:

Step 1 – Data Splitting

The historical data from the HistoricalNG database was split into 80%-20% for training set and validation set, respectively. The training set was used for the estimation model building (steps 2 and 3) while the validation set was used for validation purposes in step 5.

Step 2 – DAG specification

Using the hierarchical specification of the Product Readiness SSI (see Table 2), a preliminary DAG was obtained. Figure 10 shows the obtained DAG, with metrics as root nodes, factors as intermediate nodes, and the SSI as the leaf node.

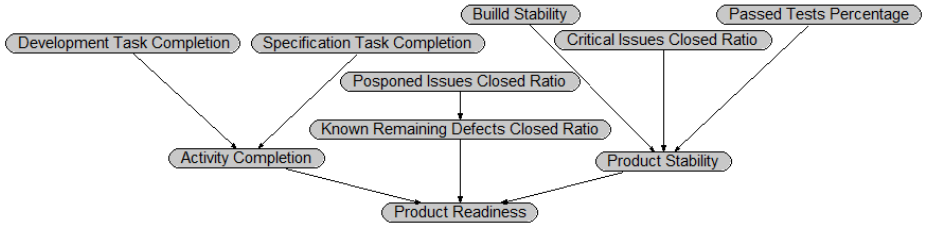


Figure 10 DAG specified for the *Product Readiness* estimation model

We held two virtual meetings with the team leader in order to complete a preliminary DAG with its corresponding states for each node. The team leader decided the states of the nodes based on the product rules commonly used in Modeliosoft. The elicited states for the metric nodes are shown in [80] (Table A1), while the elicited states for the factor nodes and the SSI node are shown in [80] (Table A2).

Additionally, for the metric nodes (i.e., root nodes) it was required to specify their binning intervals, as they were computed and stored in the HistoricalNG database in the continuous interval [0, 1]. To ease the specification of the binning intervals, we used our implemented versions of the two main unsupervised binning algorithms (Equal-Width and Equal-Frequency binning). We showed the results to the team leader as a suggestion of the binning intervals, but he preferred to define his own binning intervals for each metric node, according to their usual (implicit) rules and thresholds. The specified binning intervals can be found in [80] (Table A1, second column).

Step 3 – CPTs Specification

To specify the probability function for each node, each CPT was semi-automatically filled in using the training set and the domain experts' knowledge. Three members of the domain experts team participated in this step: the team leader, the project manager and the developer leader.

CPTs for metric nodes were automatically computed from the training set, the specified states and their corresponding binning intervals. To do so, we applied frequency quantification with our developed software tool *getFrequencyQuantification* [70]. The domain experts reviewed and refined the resulting CPTs according to their knowledge and rules. For instance, the *Build Stability* metric node (collected from Jenkins as the percentage of successful builds in a specified time period) had 5 ordinal categories defined by the domain experts, ranging from "VeryLow" to "VeryHigh", and binning intervals in the [0, 1] interval. The automatically computed CPT for this metric node ranged from 3% for the "VeryLow" category, up to 60% for the "VeryHigh", as the training data showed that software builds succeeded most of the time. The whole set of CPTs for metric nodes can be found in [80] (Table A1, third column).

CPTs for factor and SSI nodes were manually or automatically filled in depending on their size. For instance, the CPT for the factor node *Known Remaining Defects Closed ratio* was manually filled in by domain experts, as it only had a parent node (*Postponed Issues Closed Ratio*) with the states ("Low", "Medium", "High"). So, its CPT resulted in 9 entries (3 rows of 3 probabilities each).

The rest of the CPTs for factor and SSI nodes were large and required the application of the WSA technique to reduce the number of entries to fill in. The compatible configurations required by the WSA were automatically computed from the training data, using our tool *getCompatibleConfigurations* [72] previously explained. Domain experts were requested to provide the resulting probabilities for each *compatible configuration,* and, additionally, the relative weights of the parent nodes towards the quantified node.

For instance, the CPT for the *Activity Completion* factor node would require filling in 125 probabilities (25 rows of 5 probabilities each). The CPT for the *Product Stability* node would require 625 probabilities (125 rows of 5 probabilities each). And the CPT for the *Product Readiness* node would require 300 probabilities (75 rows of 4 probabilities each). In contrast, by applying our implementation of the WSA technique, the number of probabilities to be provided by domain experts decreased to 10 rows of 5 probabilities each for the *Activities Completion* node; 15 rows of 5 probabilities each for the *Product Stability* node; and 13 rows of 4 probabilities each for the *Product Readiness* node.

The complete set of CPTs for factor and SSI nodes are provided in [80] (Tables A3-A6).

Step 4 – Estimation Model Generation

Once the CPTs for all the nodes were specified, the complete Bayesian network for the *Product Readiness* SSI was created using Netica® software [64]. The resulting estimation model can be found in [80] (Figure A1).

Step 5 – Estimation Model Validation

Once the estimation model for the *Product Readiness* SSI was generated, we conducted the two validations mechanisms proposed by the SESSI method: Model Walkthrough and Outcome Adequacy over its factor and SSI nodes.

Model Walkthrough validation

This validation aims to test and recalibrate the estimation model using hypothetical scenarios and the domain experts' perceptions for these scenarios. Three domain experts participated in this activity: the project manager, the team leader and a developer.

A total of 41 hypothetical scenarios and their expected states were provided by the domain experts. For instance, the domain experts designed 14 hypothetical scenarios for the *Product Stability* node. One of the designed scenarios was *Build Stability* as "Medium", *Critical Issues (Closed) Ratio* as "VeryHigh" and *Passed Tests Percentage* as "Medium". For this scenario, the experts specified the most probable state for the *Product Stability* factor as "Medium", which matched the output of the estimation model. For the cases in which there were mismatches, the corresponding CPT was modified by tuning the probabilities of the corresponding row. Then, the previous scenarios were re-introduced into the estimation model, to make sure that the output kept matching with the domain experts' perception. Individual tables showing the conducted Model Walkthrough are in [80] (Tables A7-A10). The average accuracy obtained in this validation was suboptimal due to the high number of scenarios that required recalibration for the *Activity Completion* node. In [80] (Figure A2) we show the recalibrated estimation model resulting from this validation.

Outcome Adequacy

It aims to validate the model using real scenarios from the validation dataset. The project manager and the team leader participated in this validation as domain experts.

Each real scenario from the validation set consisted of the combination of the states of the parent nodes together with the date when the scenario happened. We showed such information to the domain experts and requested them to specify the resulting state for each scenario. Their answer was then compared with the output of the estimation model for that scenario. For instance, for the *Product Stability* node, there were 10 real scenarios in the validation set, and domain experts provided a potential state for each of these scenarios. 3 out of these 10 scenarios resulted in mismatches

between the domain experts' perception and the estimation model output. Therefore, they required model recalibration. Individual tables showing the conducted Outcome Validation can be consulted at [80] (Tables A11-A14).

Table 3 shows a summary of the results from the performed validations. It includes the number of considered scenarios, the number of mismatches that required model recalibration and the percentage of matches (accuracy).

Table 3 Summary of the two validations conducted in the ModelioNG case study

Model Walkthrough Validation			
Node	**Number of scenarios designed**	**Required recalibration**	**Matches (%)**
Activity Completion	12	7	41,6
Known Remaining Defects Closed Ratio	1	0	100
Product Stability	14	4	71,4
Product Readiness	14	4	71,4
Total	**41**	**15**	**63,4**
Outcome Adequacy Validation			
Node	**Number of real scenarios considered**	**Required recalibration**	**Matches (%)**
Activity Completion	1	0	100
Known Remaining Defects Closed Ratio	1	0	100
Product Stability	10	3	70
Product Readiness	6	1	83,3
Total	**18**	**4**	**77,7**

Once the model was successfully recalibrated, the final estimation model for the *Product Readiness* SSI was obtained. An excerpt of the resulting estimation model is shown in Figure 11 (a).

Step 6 – Deployment and use of the Product Readiness SSI estimation model

After obtaining the final estimation model for the *Product Readiness* SSI, we discussed with Modeliosoft the most feasible alternatives for its deployment in the organization.

Since the very beginning of the case study, Modeliosoft stated their interest on integrating the resulting estimation model into a dashboard [81] that was being promoted by its headquarter Softeam to visualize and monitor SSIs.

We built the infrastructure to connect the *Product readiness* estimation model with such dashboard. It was feasible as the dashboard has a software orchestrator component that allows the connection of data collectors with the mechanisms to process the data to be visualized. We developed a software library [73] that obtains the collected metrics and returns the resulting set of probabilities for each state of the SSI, which can then be visualized in the dashboard. This architecture is graphically represented in Figure 11 (b).

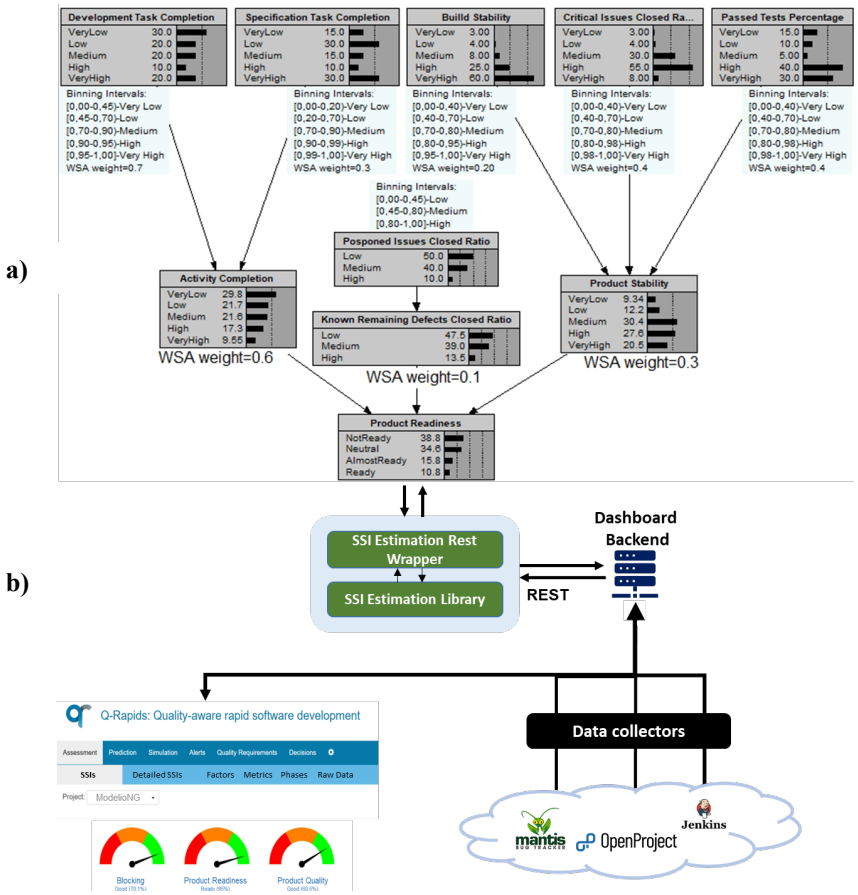


Figure 11 a) Final estimation model obtained after the two validation processes. b) Architecture to enable the automatic estimations and visualization of the *Product Readiness* SSI for ModelioNG

Modeliosoft was interested on what-if analysis to assess scenarios that could help them to take preventive actions, with the aim of reducing the risk of delivering the software product without meeting its requirements. To enable such analysis we used Netica® software [64]. Figure 12 shows an example of a what-if analysis conducted with the *Product Readiness* estimation model. We manually entered a scenario where the development of features is almost finished (that is, *Development Task Completion* and *Specification Task Completion* metrics are in "High" and "VeryHigh" states, respectively) but the percentage of minor bugs addressed is low, as well as every metric belonging to the *Product Stability* factor. In this scenario, the estimation model results in "NotReady" as the most probable state for the *Product Readiness* SSI. This is because even when the features to deliver are almost completed, the stability of the software and the percentage of non-closed minor bugs are deficient.

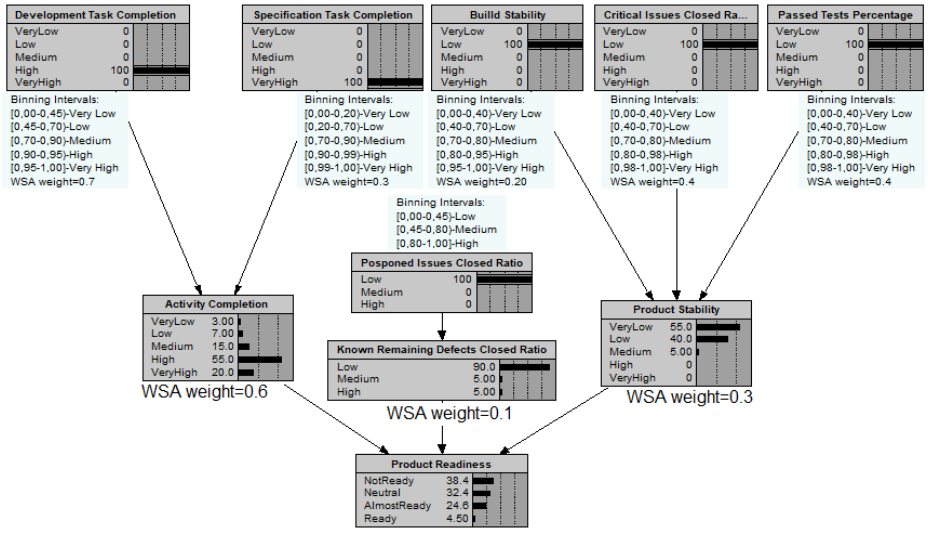


Figure 12 What-if analysis example using the *Product Readiness* SSI estimation model resulting from the case study

6.5 Preliminary Feedback

Right after the execution of the SESSI method, we requested the feedback on the application of the method from the case study participants. The questionnaire is available in [80] (page 8). It contained open and closed questions organized into three main sections. Each one of these sections focused on:

1. Ranking the execution of the method as: usable, clear, difficult, reliable, complete, comprehensive and repeatable.
2. Positive/negative aspects of the method observed by the participants during its execution.
3. Opinion on the reproducibility of the method in another case or context without our support.

Although we requested each participant to fill in the questionnaire, the team leader provided us with a single set of answers that was collaboratively agreed among all participants during an internal meeting. We did not have knowledge nor control on this meeting and the resulting answers. We rely on these answers as a representative agreement among all the participants. The results show that completion, reliability, comprehensibility, detail, interest and repeatability got the highest scores. Although none of the aspects requested by the questionnaire was negatively scored, the self-explanatory aspect was scored as neutral. This was somewhat expected as the participants were not requested to read or be formally trained for the method execution. Instead, the research team participated as experts on the method and guided all the activities and tasks.

Regarding the open aspects, on the one hand, participants highlighted some positive aspects related to the support provided by the research team that actively participated in the execution of the method. They specifically mentioned that the explanations provided by the research team were clear and contributed to the understanding of the steps of the method. On the other hand, as negative aspects they highlighted that "historical data selection [for the phase 2 of the method] was not totally clear". Such feedback is really appreciated to work on the improvement of the method.

Finally, regarding the repeatability of the method, participants agreed on the perception that they would be able to execute the method by themselves without our help.

6.6 Threats to Validity

In this section, we detail the main threats to validity of the case study.

- **Internal validity:** We conducted a participatory case study, where we as researchers played an active role assisting the domain experts from Modeliosoft in each step of the method and providing detailed explanations in an action-research fashion. We are aware that this greatly influences the observations on the execution of the method and the received feedback. On the one hand, we should emphasize that our results should be interpreted in the context of our objective that is: illustrating that the execution of the method is flexible to deal with particular organizational situations and needs. That is, our intention was to enable the method application as a fundamental step to achieve our long-term objective, which is to study the worthiness of the estimation model and related artifacts for decision making and monitoring purposes. Because of that, the perception of Modeliosoft participants regarding the execution of the method might be biased by our own implication as participants in the case study. For instance, they never faced any direct problem or challenge regarding the execution of the method as we were the ones in charge of fitting and guiding the activities.

  There are other factors that might affect the internal validity of the artifacts resulting from the method execution. For instance, participants from Modeliosoft that participated as domain experts were selected by the team leader, mainly based on the suitability of the person's expertise for the required tasks of the method. We are aware that such selection could have been affected by the availability of the people to participate in the case study. However, we did not experience any case where the domain expert did not have the required expertise for providing us the required information. This aspect is important in order to face the post-deployment part of the case study as the quality of the resulting artifacts will impact on their accuracy and usage.
- **Construct validity:** We used the SESSI method guidelines to drive the execution of the case study. In addition, we designed and validated a questionnaire to gather participants' feedback and used our own diaries to

register our observations as case study participants. As we mentioned above, an important aspect of this case study was that we adapted as much as possible the execution of the method to the needs of Modeliosoft. For instance, we prepared additional material and adapted some activities of the method to be performed online instead of face to face with domain experts as this was more convenient for Modeliosoft. Another adaptation was related to the feedback questionnaire. It was originally aimed to be answered by each Modeliosoft participant, however, Modeliosoft considered more convenient to fill in a single questionnaire with the agreement from all participants. To deal with this situation, we rely on the provided answers as representative of the general perception of Modeliosoft participants but added a triangulation activity for confirming the results. This was done by comparing the observations from our individual diaries with the received feedback. All in all, the design of the case study was quite flexible to deal with contextual situations and we did not experience relevant problems for such adaptations, and we could even reuse some previously developed data collectors.

- **External validity:** Our industrial summative evaluation plan envisages the execution of multiple case studies to tackle different organizational contexts. In line with such objective, the purpose of the single case study presented here is not to generalize our observations regarding the execution of the method but to learn and understand some practical implications of applying our method in a very specific industrial environment. Therefore, we described the setting of the case study as much as possible (under our non-disclosure agreements with Modeliosoft) and tried to provide details on the execution of each step of the SESSI method. Furthermore, the resulting estimation model and related artifacts should be interpreted with caution, considering that they were built in the specific context of the ModelioNG case and variations on the activities of the method and the results are expected for other cases.

7. Conclusions and future work

In this paper, we presented the SESSI method, aimed to support software-development intensive organizations with guidance and tools for exploiting software development related data and expert knowledge to improve their decision-making. We also illustrated its application in a case study related to the development of ModelioNG, a software product from Modeliosoft, a software development firm.

In general, the case study showed that the application of the SESSI method for specifying and monitoring *Product Readiness* SSI in ModelioNG was smooth and feasible. The obtained estimation model and its associated data collectors enabled monitoring and what-if analysis. In addition, we succeeded in putting forward the infrastructure for the connection of the estimation model with an organizational dashboard. This last was quite important for promoting the use of the estimation model in Modeliosoft for the long-term post-deployment study included in the industrial summative evaluation plan.

We should emphasize that we directly participated in the case study and provided hands-on support to ModelioNG team on the execution of the method. However, we obtained evidence on the positive perception of the participants regarding the execution of the method again by themselves without our direct support.

The experience gained in this case study is valuable for our goal of continuously improving the SESSI method to foster its industrial application. At this respect, we have learnt some important lessons:

- *Participatory involvement was useful to achieve industrial participation*. We found that one important factor that contributed to the willingness of Modeliosoft to participate in the case study was the fact that we offered them direct involvement in the case in such a way that their employees were disturbed as less as possible from their usual activities.
- *Tradeoff between flexibility and adaptation to organizational constraints/needs and method requirements.* Offering participatory involvement was not enough for smoothly executing the case study. We had to be flexible and creative for dealing with the organization constraints and rules. For instance, as the quality of the estimation model directly depends on the knowledge of the domain experts, we highlighted this point to the team leader, so he paid special attention to assign suitable personnel. We adapted guidelines and instruments to enable such personnel to fill in the required information taking into account their specific daily schedules and constraints. In addition, as expected in most organizations, access to project data was highly restricted and subjected to non-disclosure agreements. So, we had to set up secure protocols to anonymize the data while maximize its integrity.
- *Uncovered method improvements and future work from organizational needs*. Case study participants commented that having forecasting capabilities could help them preventing SSIs violations. In that matter, we plan to extend the method with forecasting capabilities at every level of the SSI hierarchy (SSI, factors and metrics), to enable the forecasting of SSIs while providing traceability support. On the other hand, based on the feedback we got from the survey applied to the case study participants, we realized that the selection of historical data for performing the phase 2 of the method was perceived as unclear. Therefore, we are working on the development of an extension of our current software tools to support the random selection of historical data and suitable partitions for training and validation sets. We believe that the enhancement of tool support will contribute to ease the adoption of the method.
- *Sharing the case study results is relevant to foster industrial uptake.* The smooth application of the method in an organization implicitly fosters industrial uptake. Therefore, we paid special attention to all aspects related to the execution of the case from the methodological and practical perspective. We wanted to make sure that we maximize all our efforts and resources to achieve a smooth collaboration that not only ensures the expected long-term involvement with Modeliosoft for the post-method deployment evaluation but also to promote the participation of other organizations. So far, we have shared the preliminary results of this case study with other industrial representatives from our own industrial collaboration network and they seem interested on applying the method with a similar approach

as the one used in Modeliosoft. This also supports our interest of presenting our preliminary results in highly reputed venues.

All in all, these results and lessons learnt are not only the basis for improvements and future research but also for shaping the design of our future case studies and for preparing the second part of this case study: the post-deployment part.

Acknowledgments

This work was supported by the European Union's Horizon 2020 research and innovation program (Q-Rapids project, grant agreement N° 732253), and partially by the Spanish Ministry of Economy and Competitiveness (project GENESIS, grant agreement TIN2016-79269-R). We also thank Softeam and Modeliosoft for promoting and participating in the case study, respectively.

References

[1] T. Menzies, T. Zimmermann, Software Analytics: What's Next?, IEEE Softw. 35 (2018) 64–70. https://doi.org/10.1109/MS.2018.290111035.

[2] C. Matthies, G. Hesse, Towards using Data to Inform Decisions in Agile Software Development: Views of Available Data, in: Proc. 14th Int. Conf. Softw. Technol. ICSOFT '19, SCITEPRESS - Science and Technology Publications, 2019: pp. 552–559. https://doi.org/10.5220/0007967905520559.

[3] N.B. Moe, A. Aurum, T. Dybå, Challenges of shared decision-making: A multiple case study of agile software development, Inf. Softw. Technol. 54 (2012) 853–865. https://doi.org/10.1016/j.infsof.2011.11.006.

[4] R.P.L. Buse, T. Zimmermann, Information needs for software development analytics, in: Proc. 34th Int. Conf. Softw. Eng. ICSE '12, IEEE, 2012: pp. 987–996. https://doi.org/10.1109/ICSE.2012.6227122.

[5] S. Martinez-Fernandez, A.M. Vollmer, A. Jedlitschka, X. Franch, L. Lopez, P. Ram, P. Rodriguez, S. Aaramaa, A. Bagnato, M. Choras, J. Partanen, Continuously Assessing and Improving Software Quality with Software Analytics Tools: A Case Study, IEEE Access. 7 (2019) 68219–68239. https://doi.org/10.1109/ACCESS.2019.2917403.

[6] R.B. Svensson, R. Feldt, R. Torkar, The Unfulfilled Potential of Data-Driven Decision Making in Agile Software Development, in: Proc. 20th Int. Conf. Agil. Softw. Dev. XP '19, Springer, Cham, 2019: pp. 69–85. https://doi.org/10.1007/978-3-030-19034-7_5.

[7] M. Choetkiertikul, H.K. Dam, T. Tran, A. Ghose, J. Grundy, Predicting Delivery Capability in Iterative Software Development, IEEE Trans. Softw. Eng. 44 (2018) 551–573. https://doi.org/10.1109/TSE.2017.2693989.

[8] M. Yan, X. Xia, X. Zhang, L. Xu, D. Yang, S. Li, Software quality assessment model: a systematic mapping study, Sci. China Inf. Sci. 62 (2019) 191101. https://doi.org/10.1007/s11432-018-9608-3.

[9] M. Perkusich, G. Soares, H. Almeida, A. Perkusich, A procedure to detect

problems of processes in software development projects using Bayesian networks, Expert Syst. Appl. 42 (2015) 437–450. https://doi.org/10.1016/j.eswa.2014.08.015.

[10] A. Freire, M. Perkusich, R. Saraiva, H. Almeida, A. Perkusich, A Bayesian networks-based approach to assess and improve the teamwork quality of agile teams, Inf. Softw. Technol. 100 (2018) 119–132. https://doi.org/10.1016/j.infsof.2018.04.004.

[11] M. Janssen, H. van der Voort, A. Wahyudi, Factors influencing big data decision-making quality, J. Bus. Res. 70 (2017) 338–345. https://doi.org/10.1016/j.jbusres.2016.08.007.

[12] H.E. Kyburg, J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference., J. Philos. 88 (1991) 434. https://doi.org/10.2307/2026705.

[13] Q-Rapids, Quality-aware Rapid Software development (Q-Rapids) project. European Union's Horizon 2020 research and innovation programme under grant agreement No 732253., (2019). https://www.q-rapids.eu (accessed June 13, 2020).

[14] R.J. Wieringa, Design Science Methodology for Information Systems and Software Engineering, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. https://doi.org/10.1007/978-3-662-43839-8.

[15] D. Avison, Action research: a research approach for cooperative work, in: Proc. 7th Int. Conf. Comput. Support. Coop. Work Des. CSCWD '03, COPPE/UFRJ, 2003: pp. 19–24. https://doi.org/10.1109/CSCWD.2002.1047641.

[16] M. Manzano, C. Gomez, C. Ayala, S. Martinez-Fernandez, P. Ram, P. Rodriguez, M. Oriol, Definition of the On-time Delivery Indicator in Rapid Software Development, in: Proc. IEEE 1st Int. Work. Qual. Requir. Agil. Proj. QuaRAP '18, IEEE, 2018: pp. 1–5. https://doi.org/10.1109/QuaRAP.2018.00006.

[17] M. Manzano, E. Mendes, C. Gómez, C. Ayala, X. Franch, Using Bayesian Networks to estimate Strategic Indicators in the context of Rapid Software Development, in: Proc. 14th Int. Conf. Predict. Model. Data Anal. Softw. Eng. PROMISE '18, ACM, New York, NY, USA, 2018: pp. 52–55. https://doi.org/10.1145/3273934.3273940.

[18] N. Nagappan, T. Ball, Static analysis tools as early indicators of pre-release defect density, in: Proc. 27th Int. Conf. Softw. Eng. ICSE '05, ACM Press, New York, New York, USA, 2005: p. 580. https://doi.org/10.1145/1062455.1062558.

[19] K. Petersen, A palette of lean indicators to detect waste in software maintenance: A case study, in: Proc. 13th Int. Conf. Agil. Softw. Dev. XP '12, Springer, Berlin, Heidelberg, 2012: pp. 108–122. https://doi.org/10.1007/978-3-642-30350-0_8.

[20] K. Petersen, C. Wohlin, Measuring the flow in lean software development, Softw. Pract. Exp. 41 (2011) 975–996. https://doi.org/10.1002/spe.975.

[21] S. Wagner, K. Lochmann, L. Heinemann, M. Kläs, A. Trendowicz, R. Plösch, A. Seidi, A. Goeb, J. Streit, The Quamoco product quality modelling and assessment approach, in: Proc. 34th Int. Conf. Softw. Eng. ICSE '12, IEEE,

2012: pp. 1133–1142. https://doi.org/10.1109/ICSE.2012.6227106.
[22] N. Fenton, M. Neil, D. Marquez, Using Bayesian networks to predict software defects and reliability, Proc. Inst. Mech. Eng. Part O J. Risk Reliab. 222 (2008) 701–712. https://doi.org/10.1243/1748006XJRR161.
[23] J.P. Carvallo, X. Franch, G. Grau, C. Quer, COSTUME: A method for building quality models for composite COTS-based software systems, in: Proc. 4th Int. Conf. Qual. Software, QSIC '04, IEEE, 2004: pp. 214–221. https://doi.org/10.1109/QSIC.2004.1357963.
[24] V. Antinyan, M. Staron, W. Meding, P. Osterstrom, E. Wikstrom, J. Wranker, A. Henriksson, J. Hansson, Identifying risky areas of software code in Agile/Lean software development: An industrial experience report, in: Proc. 1th Softw. Evol. Week - IEEE Conf. Softw. Maintenance, Reengineering, Reverse Eng. CSMR-WCRE, '14, IEEE, 2014: pp. 154–163. https://doi.org/10.1109/CSMR-WCRE.2014.6747165.
[25] S. Ilieva, P. Ivanov, E. Stefanova, Analyses of an agile methodology implementation, in: Proc. 30th Euromicro Conf. EUROMICRO '04., IEEE, 2004: pp. 326–333. https://doi.org/10.1109/EURMIC.2004.1333387.
[26] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu, V. Filkov, Quality and productivity outcomes relating to continuous integration in GitHub, in: Proc. 10th Jt. Meet. Found. Softw. Eng. ESEC/FSE '15, ACM Press, New York, New York, USA, 2015: pp. 805–816. https://doi.org/10.1145/2786805.2786850.
[27] M. Staron, W. Meding, K. Palm, Release Readiness Indicator for Mature Agile and Lean Software Development Projects, in: Proc. 13th Int. Conf. Agil. Softw. Dev. XP '12, Springer, Berlin, Heidelberg, 2012: pp. 93–107. https://doi.org/10.1007/978-3-642-30350-0_7.
[28] A. Asthana, J. Olivieri, Quantifying software reliability and readiness, in: Proc. IEEE Int. Work. Tech. Comm. Commun. Qual. Reliab. CQR '09, IEEE, 2009: pp. 1–6. https://doi.org/10.1109/CQR.2009.5137352.
[29] A.T. Misirli, A.B. Bener, Bayesian Networks For Evidence-Based Decision-Making in Software Engineering, IEEE Trans. Softw. Eng. 40 (2014) 533–554. https://doi.org/10.1109/TSE.2014.2321179.
[30] N. Fenton, M. Neil, W. Marsh, P. Hearty, Ł. Radliński, P. Krause, On the effectiveness of early life cycle defect prediction with Bayesian Nets, Empir. Softw. Eng. 13 (2008) 499–537. https://doi.org/10.1007/s10664-008-9072-x.
[31] M. Staron, W. Meding, C. Nilsson, A framework for developing measurement systems and its industrial evaluation, Inf. Softw. Technol. 51 (2009) 721–737. https://doi.org/10.1016/j.infsof.2008.10.001.
[32] M. Staron, W. Meding, Monitoring bottlenecks in agile and lean software development projects - A method and its industrial use, in: Proc. 12th Int. Conf. Prod. Focus. Softw. Process Improv. PROFES '11, Springer, Berlin, Heidelberg, 2011: pp. 3–16. https://doi.org/10.1007/978-3-642-21843-9_3.
[33] M. Perkusich, K. Gorgônio, H. Almeida, A. Perkusich, A Framework to Build Bayesian Networks to Assess Scrum-based Development Methods, in: Proc. 29th Int. Conf. Softw. Eng. Knowl. Eng. SEKE '07, 2017: pp. 67–73. https://doi.org/10.18293/SEKE2017-139.
[34] E. Mendes, P. Rodriguez, V. Freitas, S. Baker, M.A. Atoui, Towards

improving decision making and estimating the value of decisions in value-based software engineering: the VALUE framework, Softw. Qual. J. 26 (2018) 607–656. https://doi.org/10.1007/s11219-017-9360-z.

[35] E. Mendes, M. Perkusich, V. Freitas, J. Nunes, Using Bayesian Network to estimate the value of decisions within the context of Value-Based Software Engineering, in: Proc. 22nd Int. Conf. Eval. Assess. Softw. Eng. 2018, EASE'18, ACM Press, New York, New York, USA, 2018: pp. 90–100. https://doi.org/10.1145/3210459.3210468.

[36] X. Zheng, P. Martin, K. Brohman, L. Da Xu, Cloudqual: A quality model for cloud services, IEEE Trans. Ind. Informatics. 10 (2014) 1527–1536. https://doi.org/10.1109/TII.2014.2306329.

[37] R. Baggen, J.P. Correia, K. Schill, J. Visser, Standardized code quality benchmarking for improving software maintainability, Softw. Qual. J. 20 (2012) 287–307. https://doi.org/10.1007/s11219-011-9144-9.

[38] T. Bakota, P. Hegedus, P. Kortvelyesi, R. Ferenc, T. Gyimothy, A probabilistic software quality model, in: Proc. 27th IEEE Int. Conf. Softw. Maintenance, ICSM '11, IEEE, 2011: pp. 243–252. https://doi.org/10.1109/ICSM.2011.6080791.

[39] S. Wagner, A Bayesian network approach to assess and predict software quality using activity-based quality models, Inf. Softw. Technol. 52 (2010) 1230–1241. https://doi.org/10.1016/j.infsof.2010.03.016.

[40] K. Mordal-Manet, F. Balmas, S. Denier, S. Ducasse, H. Wertz, J. Laval, F. Bellingard, P. Vaillergues, The squale model - A practice-based industrial quality model, in: Proc. 25th IEEE Int. Conf. Softw. Maintenance, ICSM '09, IEEE, 2009: pp. 531–534. https://doi.org/10.1109/ICSM.2009.5306381.

[41] W. Pedrycz, J.F. Peters, S. Ramanna, Software quality measurement: concepts and fuzzy neural relational model, in: Proc. IEEE Int. Conf. Fuzzy Syst. Proceedings. IEEE World Congr. Comput. Intell., IEEE, 1998: pp. 1026–1031. https://doi.org/10.1109/FUZZY.1998.686259.

[42] L. Zhang, L. Li, H. Gao, 2-D software quality model and case study in software flexibility research, in: Proc. Int. Conf. Comput. Intell. Model. Control Autom. CIMCA '08, IEEE, 2008: pp. 1147–1152. https://doi.org/10.1109/CIMCA.2008.70.

[43] P. Johnson, R. Lagerström, P. Närman, M. Simonsson, Enterprise architecture analysis with extended influence diagrams, Inf. Syst. Front. 9 (2007) 163–180. https://doi.org/10.1007/s10796-007-9030-y.

[44] R. Lagerström, U. Franke, P. Johnson, J. Ullberg, A Method for creating enterprise architecture metamodels - applied to systems modifiability analysis, Int. J. Comput. Sci. Appl. 6 (2009) 89–120. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.161.4640.

[45] R. Lagerström, P. Johnson, D. Höök, Architecture analysis of enterprise systems modifiability – Models, analysis, and validation, J. Syst. Softw. 83 (2010) 1387–1403. https://doi.org/10.1016/j.jss.2010.02.019.

[46] P. Närman, M. Buschle, M. Ekstedt, An enterprise architecture framework for multi-attribute information systems analysis, Softw. Syst. Model. 13 (2014) 1085–1116. https://doi.org/10.1007/s10270-012-0288-2.

[47] P. Närman, H. Holm, M. Ekstedt, N. Honeth, Using enterprise architecture

analysis and interview data to estimate service response time, J. Strateg. Inf. Syst. 22 (2013) 70–85. https://doi.org/10.1016/j.jsis.2012.10.002.
[48] P. Närman, H. Holm, D. Höök, N. Honeth, P. Johnson, Using enterprise architecture and technology adoption models to predict application usage, J. Syst. Softw. 85 (2012) 1953–1967. https://doi.org/10.1016/j.jss.2012.02.035.
[49] S. Wagner, A. Goeb, L. Heinemann, M. Kläs, C. Lampasona, K. Lochmann, A. Mayr, R. Plösch, A. Seidl, J. Streit, A. Trendowicz, Operationalised product quality models and assessment: The Quamoco approach, Inf. Softw. Technol. 62 (2015) 101–123. https://doi.org/10.1016/j.infsof.2015.02.009.
[50] M. Yan, X. Xia, X. Zhang, L. Xu, D. Yang, S. Li, Software quality assessment model: a systematic mapping study, Sci. China Inf. Sci. 62 (2019) 191101. https://doi.org/10.1007/s11432-018-9608-3.
[51] T. Sommestad, M. Ekstedt, H. Holm, The Cyber Security Modeling Language: A Tool for Assessing the Vulnerability of Enterprise System Architectures, IEEE Syst. J. 7 (2013) 363–373. https://doi.org/10.1109/JSYST.2012.2221853.
[52] Q-Rapids, Q-Rapids Deliverable D3.1, 2017. https://www.q-rapids.eu/deliverables.
[53] A. Darwiche, Bayesian networks, Commun. ACM. 53 (2010) 80–90. https://doi.org/10.1145/1859204.1859227.
[54] C.A. Furia, R. Feldt, R. Torkar, Bayesian Data Analysis in Empirical Software Engineering Research, (2018). https://doi.org/10.1109/TSE.2019.2935974.
[55] A. Tosun, A.B. Bener, S. Akbarinasaji, A systematic literature review on the applications of Bayesian networks to predict software quality, Softw. Qual. J. 25 (2017) 273–305. https://doi.org/10.1007/s11219-015-9297-z.
[56] E. Mendes, Practitioner's knowledge representation: A pathway to improve software effort estimation, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. https://doi.org/10.1007/978-3-642-54157-5.
[57] E. Mendes, Using knowledge elicitation to improve Web effort estimation: Lessons from six industrial case studies, in: Proc. 34th Int. Conf. Softw. Eng. ICSE '12, IEEE, 2012: pp. 1112–1121. https://doi.org/10.1109/ICSE.2012.6227108.
[58] K.P. Murphy, An Introduction to Graphical Models, (2001). https://www.cs.ubc.ca/~murphyk/Papers/intro_gm.pdf (accessed September 7, 2020).
[59] B. Das, Generating Conditional Probabilities for Bayesian Networks: Easing the Knowledge Acquisition Problem, CoRR. (2004) 1–24. https://doi.org/DSTO-TR-0918.
[60] N.E. Fenton, M. Neil, J.G. Caballero, Using Ranked Nodes to Model Qualitative Judgments in Bayesian Networks, IEEE Trans. Knowl. Data Eng. 19 (2007) 1420–1432. https://doi.org/10.1109/TKDE.2007.1073.
[61] A. Wickenkamp, Q-Rapids-connect, (2019). https://git.io/JvGwf (accessed June 13, 2020).
[62] X. Franch, C. Ayala, L. Lopez, S. Martinez-Fernandez, P. Rodriguez, C. Gomez, A. Jedlitschka, M. Oivo, J. Partanen, T. Raty, V. Rytivaara, Data-Driven Requirements Engineering in Agile Projects: The Q-Rapids Approach,

in: Proc. IEEE 25th Int. Requir. Eng. Conf. Work. REW '17, IEEE, 2017: pp. 411–414. https://doi.org/10.1109/REW.2017.85.

[63] S. Martinez-Fernandez, A. Jedlitschka, L. Guzman, A.M. Vollmer, A Quality Model for Actionable Analytics in Rapid Software Development, in: Proc. 44th Euromicro Conf. Softw. Eng. Adv. Appl. SEAA '18, IEEE, 2018: pp. 370–377. https://doi.org/10.1109/SEAA.2018.00067.

[64] Norsys Software Corp., Netica TM, Application for Belief Networks and Influence Diagrams, (2002). https://www.norsys.com/netica.html (accessed February 13, 2020).

[65] S. Matsumoto, R. Carvalho, M. Ladeira, UnBBayes: a java framework for probabilistic models in AI, Http://Unbbayes.Sourceforge.Net/. (2011). http://sourceforge.net/projects/unbbayes/. (accessed February 13, 2020).

[66] S. Raschka, Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning, ArXiv. (2018). http://arxiv.org/abs/1811.12808 (accessed January 24, 2020).

[67] J. Dougherty, R. Kohavi, M. Sahami, Supervised and Unsupervised Discretization of Continuous Features, in: Mach. Learn. Proc. 1995, Elsevier, 1995: pp. 194–202. https://doi.org/10.1016/b978-1-55860-377-6.50032-3.

[68] M. Manzano, Qrapids-si_assessment – BayesUtils – makeEqualIntervals, (2019). https://git.io/JvspC (accessed June 13, 2020).

[69] N. Fenton, M. Neil, Managing risk in the modern world. Applications of Bayesian networks. A Knowledge Transfer Report, 2007. http://www.lms.ac.uk/activities/comp_sci_com/KTR/apps_bayesian_networks.pdf (accessed November 11, 2019).

[70] M. Manzano, Qrapids-si_assessment – BayesUtils – getFrequencyQuantification, (2019). https://git.io/Jvspr (accessed June 13, 2020).

[71] M. Manzano, WSA-Java implementation, (2019). https://git.io/Jvspi (accessed June 13, 2020).

[72] M. Manzano, Qrapids-si_assessment – BayesUtils – getCompatibleConfigurations, (2019). https://git.io/Jvsp1 (accessed June 13, 2020).

[73] M. Manzano, Qrapids-si_assessment, (2019). https://git.io/Jvsph (accessed June 13, 2020).

[74] A. Balletbó, Qrapids-si_assessment-rest, (2019). https://git.io/Jvshf (accessed June 13, 2020).

[75] P. Runeson, M. Höst, A. Rainer, B. Regnell, Case Study Research in Software Engineering: Guidelines and Examples, John Wiley & Sons, Inc., Hoboken, NJ, USA, 2012. https://doi.org/10.1002/9781118181034.

[76] C. Robson, Real world research : a resource for social scientists and practitioner-researchers, Blackwell Publishers, 2002.

[77] R.K. Yin, Case study research: Design and methods, 4th ed., Thousand Oaks, CA: SAGE Publications, 2009.

[78] Q-Rapids, Q-Rapids Deliverable D5.1, 2017. https://www.q-rapids.eu/deliverables.

[79] K. Krippendorff, Content analysis: an introduction to its methodology, SAGE Publications, 1980.

[80] M. Manzano, Appendix, 2020. www.essi.upc.edu/~mmanzano/files/Appendix_IST.pdf.
[81] Q-Rapids, Q-Rapids Dashboard, (2019). https://git.io/JvG0Z (accessed June 13, 2020).