IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 2, 2020, accepted October 26, 2020, date of publication November 2, 2020, date of current version November 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3035217

A Bayesian Networks-Based Method to Analyze the Validity of the Data of Software Measurement Programs

RENATA SARAIVA, AMAURY MEDEIROS, MIRKO PERKUSICH, DALTON VALADARES, KYLLER COSTA GORGÔNIO, ANGELO PERKUSICH, (Member, IEEE), AND HYGGO ALMEIDA

Embedded Systems and Pervasive Computing Laboratory, VIRTUS Research, Development and Innovation Center, Federal University of Campina Grande, Campina Grande 58429-900, Brazil

Corresponding author: Angelo Perkusich (perkusic@.ufcg.edu.br)

This work was supported in part by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

ABSTRACT Measures are essential resources to improve quality and control costs during software development. One of the main factors for having successful software measurement programs is measure trustworthiness, defined as how much a user can trust a measure to use it with confidence. Such confidence enables the users to interpret them and use them for supporting decision-making. ISO/IEC 15939:2007 describes four stages that influence such interpretability: measure selection, measure validation, threshold definition, and data validation. The literature is scarce in supporting data validation, which directly impacts the measure's trustworthiness value. This article aims to detail a method that uses Bayesian networks for supporting data validation and shows its application in practice to four software development projects from one company. The proposed method uses Bayesian networks to calculate the degree to which a collected number or symbol represents the real value for the measures and is integrated with GQM for assessing the measurement program's goals. First, the measurement users must create GQM model hierarchical structures, use it as input for constructing the Bayesian network, validate the Bayesian network, and, finally, use it to support decision-making. A tool to support the proposed method was developed and is freely available. Further, herein, the results of the case study are presented. We identified four benefits in using the proposed method: *Externalization*, *Diagnosis support*, *Measure interpretation improvement*, and *Decision-making support*. Given this, even though the initial effort to use the proposed method lasted, on average, one hour and fourteen minutes, the benefits of using it outweighed the effort of applying it. Therefore, our findings suggest that there was a positive intention in adopting the proposed method in practice.

INDEX TERMS Goal-question-metric, Bayesian network, software measurement.

I. INTRODUCTION

Measurement is an important activity to improve quality and control costs during software development [1]. According to Finkelstein and Leaning [2], software measurement is the process of defining numbers or symbols for the attributes of software entities (e.g., the attributes size, maintainability, and efficiency), which is a challenging endeavor given the abstract nature of software [3]. Mathias *et al.* [4] claimed that whenever a value is defined for an attribute, there is a measure, and a measure is a combination of measures. In practice, the terms ''measure'' and ''measure'' are used interchangeably; in this article, we use the term ''measure'' as defined in ISO/IEC 15939:2007 [5].

Software measurement enables planning, monitoring, controlling, and evaluating processes, products, and resources [6]–[9]. The data provided by software measurement gives the users a detailed view of the processes' execution and enables them to make more informed decisions [10], [11]. Large-scale organizations such as HP, Nokia, and NASA have reported that when used during the early stages of the development cycle, software measurement helps fixing and preventing defects [12]. More recently, Prause and Hönle present a domain-dependent software analytic tool in the context of the European Cooperation for Space Standardization (ECSS) measure framework to improve the transparency

The associate editor coordinating the review of this manuscript and approving it for publication was Xiao Liu.

of software development in customer-supplier relationships of space missions [13]. Chorás *et al.* [14], [15] report a practical experience on using measures related to the software development process to support Small and Medium Enterprises in developing software following an Agile methodology. Martinez-Fernandez *et al.* [16] present a case study on the use of analytics tools to continuously assess and improve software quality. Ram *et al.* [17], [18] discuss the use of measures in the context of agile software development. A more recent study by Manzano *et al.* [19] present a novel method called SESSI (Specification and Estimation of Software Strategic Indicators) to support software-development intensive organizations with guidance and tools for exploiting software development related data and expert knowledge to improve their decision making.

Although there are potential benefits to applying software measurement, a poorly defined measurement program can lead to several issues. Such issues might be related to collecting useless, redundant, incomplete, or low-quality data, resulting in the waste of effort and inconclusive and erroneous data analysis (i.e., ''garbage-in, garbage-out''). Implementing a measurement program is challenging, and despite having much available documentation about it (e.g., CMMI-Dev, Six Sigma, IEEE Std 1061, Practical Software Measurement, and ISO/IEC 15939:2007), information on how to execute them in practice is scarce [20]. Further, Meding and Staron [21] argue that interpreting the existing standards is hard.

As a result, more than 80% of software measurement initiatives fail within their first 18 months. An explanation for such a phenomenon is the challenge to use and understand the measures [1]. Fenton and Neil [22] argued that measures are mostly used for quantification purposes, but not for its original purpose: to support decision-making. Later, Fenton and Bieman [23] claimed that most initiatives fail because they fail to collect useful data and focus on the easy or convenient ones. Such a claim leads to the main factor for having successful software programs: measure trustworthiness [24], described by Ram et al [24] as how much a user can trust a measure to use it with confidence. Unfortunately, as discussed by Ram *et al.* [24], despite its importance, it is the less known factor in the literature. We argue that for a measure to be useful to support decision-making (i.e., interpreted), it must be trusted by the measurement users (e.g., project manager or quality manager).

According to ISO/IEC 15939:2007, there are four stages of the measurement process that influence such interpretability:

- **Measure selection**, which consists in selecting adequate measures to represent the attributes of the entities of interest;
- **Measure validation**, which consists in determining if the selected measures measure what they are intended to;
- ***Thresholds* definition**, which consists in defining values (or rules) that assists in classifying the measures (e.g., a red flag indicating that the product should not be released);
- **Data validation**, which consists of analyzing the level of confidence in the accuracy (i.e., reliability) of the data associated with the measures.

Regarding *Measure selection*, Bukhari, Yahaya and Deraman [25] discussed several proposed solutions such as Best Professional Judgement [26], Historical Precedence [27], Web Quality Model [28], Meta-measures [29], Multi-Criteria Decision Analysis [30] and Goal-Question-Metric (GQM) [31]. More recently, Quality Model-specific initiatives have been proposed, such as Quamoco [32] and Q-Rapids [33], which are similar to GQM. According to Bukhari, Yahaya, and Deraman [25], even though there is no consensus about how to select measures, GQM is the most recommended approach.

Measure validation has been extensively discussed [34]–[36]. Even though, as with *Measure selection*, there is no consensus about how to validate measures, the studies of Smith and Williams [34] and Antinyan *et al.* [36] are extensive guidelines on how to perform this stage.

There have also been several studies covering *Thresholds definition* (i.e., reference values) [37]–[39], which are expert-driven or data-based. On the other hand, despite been discussed by Basili [40], *Data Validation* has not been much explored by the literature. The main concern on this stage is the degree to which a collected number or symbol represents the real value for a given measure, which we define as *measure reliability*. For instance, if the *number of defects* for a given system is 0, but no tests were performed, this measure is not reliable. Therefore, the *Data validation* stage and measure reliability directly impact the measures' trustworthiness value [24].

Definition 1 (Measure reliability): The degree to which a collected number or symbol represents the real value for a given measure.

As far as we are concerned, the only study to address this stage was Perkusich *et al.* [41], which presented a method to build Bayesian networks to assist in interpreting measures considering risks of the measurement process. However, Perkusich *et al.* [41] have the limitations presented in what follows:

- It did not present the necessary details of how to construct the Bayesian networks (e.g., it was claimed that the measurement idiom [42] was applied, but the paper has no details regarding how it was applied).
- The terminology used was not in conformance with ISO/IEC 15939:2007.
- It did not present tool support.
- The validation results were presented superficially.

For improving the current support to the *Data Validation* stage, we complemented the state-of-the-art by evolving the method presented in Perkusich *et al.* [41], by integrating it with GQM (i.e., the most recommended approach for *Measurement selection*) and adapting it to conform with ISO/IEC 15939:2007. Further, we developed a tool for supporting the method's adoption in the industry and evaluated the proposed

solution with a case study with four software development projects.

This article discusses the theory behind *Measurement reliability* and shows how it can be applied, in combination with Bayesian networks, to support the *Data Validation* stage. Further, it explains how to construct Bayesian networks for calculating the degree to which a collected number or symbol represents the real value for a given measure and presents an overview of a tool developed for supporting such an endeavor. Finally, it summarizes the results of applying the proposed solution to four software development projects.

In what follows, Section II presents a brief description of *Bayesian networks* and justify their use in our method; Section III explains the proposed method in details; Section IV presents an overview of the tool developed to support adopting the proposed method; Section V reports the design and results of the case study; and, finally, Section VI presents our conclusions including our final remarks and future works.

II. BAYESIAN NETWORKS OVERVIEW

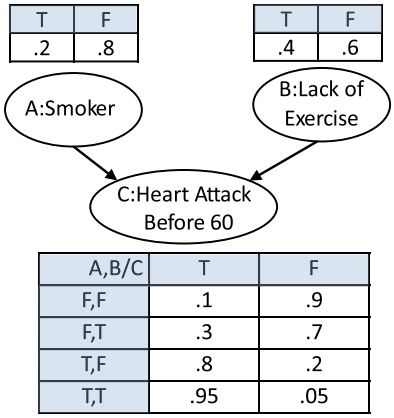
Bayesian networks belong to the family of probabilistic graph models and are used to represent knowledge about an uncertain domain. A *Bayesian network*, B, is a directed acyclic graph that represents a joint probability distribution over a set of random variables V. The network is defined by the pair $B = \{G, \Theta\}$. G is the directed acyclic graph in which the nodes $X_1, \ldots, X_n$ represent random variables and the arcs represent the direct dependencies between these variables. Θ represents the set of the probability functions. This set contains the parameter $\theta_{x_i|\pi_i} = P_B(x_i|\pi_i)$ for each x_i in X_i conditioned by π_i, the set of the parameters of X_i in G. Equation 1 presents the joint distribution defined by B over V.

$$P_B(X_1, \ldots, X_n) = \prod_{i=1}^{n} P_B(x_i|\pi_i) = \prod_{i=1}^{n} \theta_{X_i|\pi_i} \qquad (1)$$

Figure 1 presents an example of a *Bayesian network*. In Figure 1, ellipses represent the nodes, arrows represent the arcs, and tables represent the probability functions. Even though the arcs represent the causal connection's direction between the variables, information can propagate in any direction [43].

Bayesian networks have many benefits, such as suitability for small and incomplete data sets, structural learning possibility, the combination of different sources of knowledge, an explicit treatment of uncertainty, support for decision analysis, and fast responses [44]. Therefore, they are commonly applied to support systems with uncertainty [45]. *Bayesian networks* have been used for several expert systems such as assisting in safety decision making in complex project environments [46] and predicting performance in innovation projects given their transformational leadership characteristics [47].

Bayesian networks have many benefits, such as suitability for small and incomplete data sets, structural learning possibility, the combination of different sources of knowledge, an explicit treatment of uncertainty, support for decision analysis, and fast responses [44]. Therefore, they are commonly applied to support systems with uncertainty.



A,B/C	T	F
F,F	.1	.9
F,T	.3	.7
T,F	.8	.2
T,T	.95	.05

FIGURE 1. A *Bayesian network* example.

There are two challenges to build *Bayesian networks*: building the directed acyclic graph and defining the probability functions [48]. To assist in building the directed acyclic graph, [42] presents a set of idioms, which are *Bayesian network* fragments that represent the graphical part of generic types of uncertain reasoning.

In this paper, we only present the idioms used in our method: cause-consequence, measurement, and synthesis. Cause-consequence idiom models the uncertainty of a causal process with observable consequences. Measurement idiom models the uncertainty about the accuracy of any measurement. Synthesis idiom models the synthesis or combination of many nodes into one node to organize the *Bayesian network*. Figure 2 presents examples of such idioms.

The *Bayesian network*'s probability functions are usually represented as node probability tables. The two forms to collect data and define node probability tables are through (*i*) databases and (*ii*) domain experts [49]. Defining node probability tables from databases can be automated by a process called *batch* learning [50]. However, for many practical problems, one rarely finds an adequate database. Manually defining node probability tables through domain experts can become unfeasible depending on the number of nodes and states.

As shown by Fenton *et al.* [51], all kinds of inconsistencies could occur if domain experts try to elicit the probability table exhaustively for a node with a large number of states (e.g., 125). There are several methods to reduce this complexity and to encode expertise in large node probability tables. Noisy-OR [52] and Noise-MAX [53] are well-established methods, but Noisy-OR only applies to Boolean nodes Noisy-MAX does not model the range of relationships we seek here. Das [54] proposed the weighted-sum algorithm to populate node probability tables while easing the extent of knowledge acquisition for nodes with an ordinal scale.

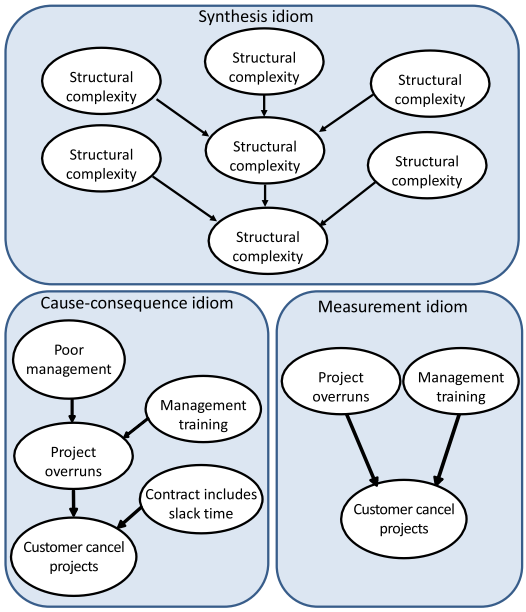


FIGURE 2. Examples.

III. PROPOSED SOLUTION

This section describes the proposed solution for supporting the *Data validation* stage of the software measurement process, as described on ISO/IEC 15939:2007. It consists of a method for constructing Bayesian networks to calculate the measurement reliability. As previously discussed, we define measurement reliability as ''the degree in which a collected number or symbol represents the real value for a given measure''. Notice that *measure reliability* is different than *measure validity*. *Measure validity* is related to how representative the measure is for measuring the intended attributes of the given software entities and is associated with the *Measure validation* stage of ISO/IEC 15939:2007. *Measure reliability* is related to the procedures deployed to operationalize the measurement program (e.g., data collection procedures); thus, it is associated with the *Data validation* stage of ISO/IEC 15939:2007. Both, *measure validity* and *measure reliability* have a direct influence on *measure trustworthiness* [24]. In what follows, Section III-A presents the conceptual model that we defined for *measure reliability*, and Sections III-B and III-C describe the proposed method, which applies the conceptual model for building the Bayesian networks.

A. MEASURE RELIABILITY

We describe *measure reliability* in terms of three uncertainty sources: *entity bias*, *measure collection process* and *measure reporting process*. *Entity bias* refers to the bias in creating the artifact that negatively influences the measurement program. Examples of *entity bias* are using mocks or fake methods to increase the code coverage by testing tools. *Measure collection process* is the process to collect the data associated with a measure, which can be manual (e.g., perform manual tests to measure the number of unique defects) or automatic (e.g., use of code coverage tool to measure the size of the test suite). *Measure reporting process* is the process to report measures, which can rely on humans (e.g., report a defect) or not (e.g., report the code coverage). To exemplify the difference between both processes, an invalid defect (i.e., an expected system behavior that is incorrectly considered as a defect by the tester) is an issue of the *measure collection process*; a duplicated defect, of the *measure reporting process*. Thus, we define the concept of *reliability factors* in what follows.

Definition 2 (Reliability factors): The factors (i.e., activity, role, or tool) that influence the measure's reliability. In this context, there are three sources of uncertainty (or types of factors): entity bias, measure collection process, and measure reporting process.

Notice that both, the *measure collection process* and *measure reporting process* are also subject to bias (or risks). For instance, examples of bias concerning the *measure collection process* are writing unit tests but not attempting to break the code [55], writing test code with no *asserts*, and computing code coverage using incorrect filters (i.e., filtering out relevant packages of the software). Regarding *measure reporting process*, an example of bias would be to have a testing team that is paid per defect found and purposefully registers duplicated or invalid defects to increase the defect count and, consequently, its payment. Figure 3 shows the conceptual model for measure reliability.

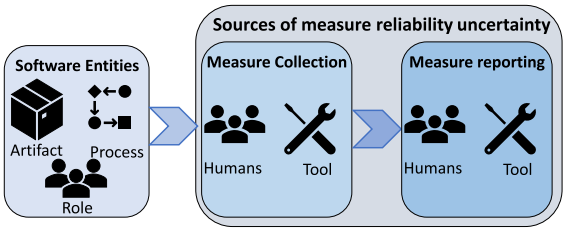


FIGURE 3. Conceptual model for measure reliability.

B. METHOD OVERVIEW

This section introduces the proposed method to build *Bayesian network*-based models to evaluate the reliability of measures, answer each question, and assess the goals defined using GQM. The models consider reliability factors (i.e., *entity bias*, *measure collection process factors* and *measure reporting process factors*) and risks. For each collected measure, the end goal is to calculate its interpretation, which reflects the confidence the decision-maker should have to use the given measure.

As shown in Figure 4 the proposed method consists of five steps: (*i*) goals, questions, and measures identification; (*ii*) Directed Acyclic Graph (DAG) construction; (*iii*) probability functions definition; (*iv*) Model validation; and (*v*) Model usage. In the following paragraphs, we briefly describe the proposed method.

In step (*i*), a knowledge acquisition meeting with practitioners is performed to identify the goals, questions, and

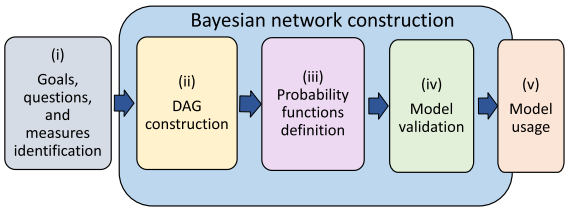


FIGURE 4. Overview of the method.

measures associated with the quality focus stated in the goals. Abstraction sheets or a similar tool can be used to acquire knowledge, keeping in mind that we are not interested in variation factors (i.e., factors that explain baseline hypotheses), but in reliability factors (i.e., factors that explain the confidence that we have in the collected measure).

As with variation factors, it is possible to define questions and measures for each reliability factor. By the end of this step, we end up with groups of quality focus and reliability factors measures. Each group contains measures used to answer a question related to a quality focus. Notice that a measure can belong to more than a group. We present more details on how to execute this step in Section III-C.

Step (*ii*) is also executed during this meeting. This step focuses on building the DAG corresponding to the project. This step includes the identification of relationships between measures, risks, and reliability factors. The end goal is to have a DAG constructed for each goal to be assessed.

In step (*iii*), the probability functions of the Bayesian networks are defined. There are several possibilities for executing this step, such as applying the ranked nodes method [51] or the weighted-sum algorithm [54], for ordinal nodes, or Noisy-OR, for Boolean nodes.

In step (*iv*), the model is evaluated by the measurement users, using expert-based Bayesian network validation procedures such as model walkthrough (i.e., simulated scenarios) [56], [57], and, if data is available, using data-driven validation such as outcome adequacy (i.e., predictive accuracy) [56], [58]). Notice that steps (*ii*), (*ii*), and (*iv*) are directly related to the Bayesian network and are aligned with the Expert-based Knowledge Engineering of Bayesian networks (EKEBN) method [56]. Finally, whenever the measurement users are satisfied with the model performance, it is used for performing decision analysis, in step (*v*).

C. METHOD DETAILED DESCRIPTION

This section presents details about how to execute the method. The method is presented formally to avoid ambiguity, but also with a running example, to improve its understandability. Figure 5 presents details of the five steps of the proposed method. In what follows, Sections III-C1, III-C2, III-C3, III-C4, and III-C5 present details of the method's steps (*i*), (*ii*), (*iii*), (*iv*), and (*v*), respectively.

1) STEP (*i*): GOALS, QUESTIONS, AND MEASURES IDENTIFICATION

In step (*i*), we apply GQM, and identify the goals, questions, and measures. Additionally, we perform tasks to organize the data for constructing the Bayesian network (i.e., steps (*ii*), (*iii*), and (*iv*)).

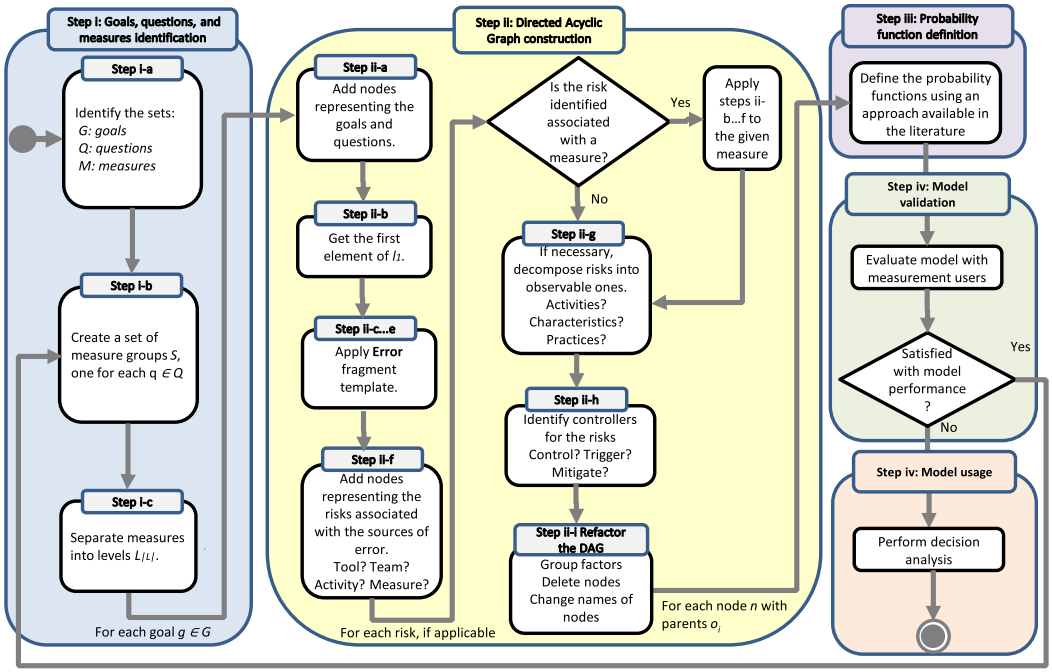


FIGURE 5. Proposed method steps.

Step i-a: identify the set of goals, questions and measures regarding the project. We define a set of goals as $G = \{g_1, \ldots, g_{|G|}\}$, where g_i represents a project goal and $|G|$ is the number of goals defined by project. Then, we define a set of questions $Q = \{q_1, \ldots, q_{|Q|}\}$, where q_i represents a question related to, at least, one goal used in the project and $|Q|$ is the total number of questions. Since each question should be associated with at least one goal, we create a set of relationships between goals and questions T and $t(g, q)$ means that goal g and question q are related. Finally, we define a set of measures $M = \{m_1, \ldots, m_{|M|}\}$, where m_i is a measure and $|M|$ is the number of measures selected.

Example: a GQM model hierarchical structure containing the goals, questions, and measures of a project is shown in Figure 6. In this example, the goal of the practitioner is to "**Analyze** the software product **with respect to** the number of defects **for the purpose of** characterization **from the point of view of** the developer". For the quality focus, the measurement user formulates the question, "How many defects does the product have?". For the reliability factor, the measurement user formulates the question, "What is the testing quality?". For the first question, the measurement user chooses the measure *Number of open defects* and, for the second, *Code coverage*. If the *Number of unique defects* identified is (near) zero and *Code coverage* is high, and it increases the confidence that the software behaves as expected.

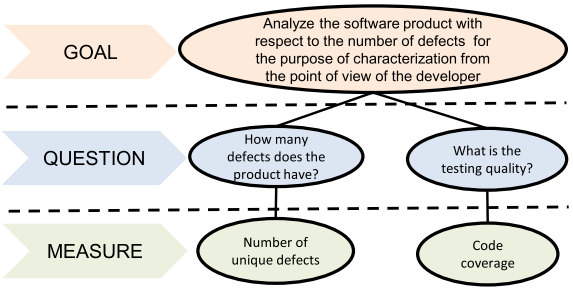


FIGURE 6. Example of a GQM model hierarchical structure.

Step i-b: group the measures, given the question they answer. The same measure can belong to multiple measure groups because it can be used to answer multiple questions. Each measure group is represented by a set, where each element belongs to M.

Example: measures related to quality focus *Number of defects*, such as *Number of opened defects* and *Number of static code analysis warnings*, might belong to the same group.

Formally, we represent the set of measure groups as $S = \{S_1, \ldots, S_{|Q|}\}$, where S_i represents the ith group of measures. Each group contains only measures used in the project and corresponds to only one question, so $\forall_{s \in S_i}\ (s \in M)$ and $m_i \in$ one or more s.

We represent the relationship between a measure group and a question as $r(S_i, q)$, which means that measures contained in the group S_i answer question q. We also create a set of A that contains every relationship between measure groups and questions. Thus, $\forall_{s \in S} \exists_{q \in Q}\ (r(s, q) \in A)$.

Step i-c: classify each group $S_i \subset S$ in levels according to the dependency between the measures and the importance of them regarding the question they answer. Thus, we define a set of levels $L = \{L_1, \ldots, L_{|L|}\}$, where $|L|$ is the number of hierarchical levels of the group S_i. For each level $L_i \in L$, we have to make sure that each measure belonging to the level is also contained in S_i, so $\forall_{L_i \in L} \forall_{m \in L_i}\ (m \in S_i)$.

In practice, the levels can be defined using a simple rule: if the measure is related to the quality focus, place it in the first level; otherwise, if it is related to reliability factors, place it in the second level. Afterward, with more analysis, it is possible to decompose S into more levels.

Example: for the example given earlier, since the number of defects detected depends on the quality of the tests (i.e., reliability factor), *Number of opened defects* $\in L_1$ and *Test coverage* $\in L_2$. *Number of opened defects* belongs to the first level because it is directly related to the quality focus of the question *"How many defects?"*. Conversely, *Test coverage* belongs to the second level because it is related to the test strategy, which is the means to collect data regarding software defects. Therefore, it is considered a *reliability factor*.

Notice that, since each measure group $S_i \subset S$ is associated with a question $q_i \in Q$, by logical consequence, given T, each S_i is associated with at least one goal $g_j \in G$.

2) STEP (*ii*): DAG CONSTRUCTION

In step (*ii*), for each goal $g_i \in G$, we define a DAG. Until each g_i and the associated S_i are empty, we perform a sequence of steps. The process of constructing the DAG identifies the reliability factors (might be associated with measures) and measure collection risks that influence the measure's reliability. The reliability factors can be decomposed by applying the synthesis idiom until the practitioner defines that the leaf nodes represent observable variables. An observable variable represents a process factor that the practitioner judges to be easily observable in practice, such as identifying if there are sufficient tools available for the team to perform the necessary tests. A risk is decomposed into factors that control, trigger or mitigate it. This step is subdivided into nine steps, which are described as follows.

Step ii-a: For each goal g_i, add the relationships $p(q, g_i)$, where $p(x, y)$ indicates that x is a parent of y. The relationships p_i must be defined in terms of T. In other words, for each $t \in T$, there must be a relationship p connecting questions and goals $(\forall_{g \in G}\ (t(g, q) \in T \rightarrow p(q, g) \in P))$.

P is the set of parent-child relationships representing the DAG of the Bayesian network. In practice, at the end of this step, the DAG of the Bayesian network contains nodes for each goal and question and edges connecting the nodes following the rules defined in T, where the questions are parent nodes, and the goals are child nodes.

Step ii-b: the steps described from here on are applied for each $S_i \subset S$. First, get and remove each element contained in the first nonempty set that represents a level (starting from L_1)

of S_i and call it f_j. Add a relationship to P, a set of parent-child relationships: $p(f_j, q)$, where $r(S_i, q) \in A$ and $p(x, y)$ indicates that x is a parent of y.

Step ii-c: create a node I_{F_j}, which represents the interpretation of f_j. The interpretation of a measure represents its value considering the reliability factors, further explained in ***step ii-d***.

Step ii-d: create two nodes, I_{F_i-M} and I_{F_i-E}, which represents, respectively, the actual data collected for the measure and the measurement error. Add two relationships to P: $p(I_{F_i}, I_{F_i-M})$ and $p(I_{F_i-E}, I_{F_i-M})$. Up to this point, we have only applied the measurement idiom [48]. For I_{F_i-M} and I_{F_j} and the measurement user can model it as Boolean or ordinal nodes (i.e., the same scale for both nodes). We recommend to model I_{F_i-E} as an ordinal node.

It is worth mentioning that the raw data collected for the measures are usually on different scales, such as numerical or proportional. Therefore, to operationalize the conversion of scales into a Boolean or ordinal one, the measurement user should use thresholds using any literature's techniques. Therefore, this step relates to the stage *Thresholds definition*, as described in ISO/IEC 15939:2007. For example, for the *Code coverage* measure, the measurement user might decide to use a 3-points Likert scale (i.e., *Bad*, *Moderate*, *Good*) and that it is bad if lower than 50%, moderate if greater than 50% and below 80%, and good if greater or equal than 80%. I_{F_i-E} represents the *Error* and represent the measurement's uncertainty, which is further discussed in ***step ii-e***. It is out of the scope of this paper to discuss details regarding how to define thresholds.

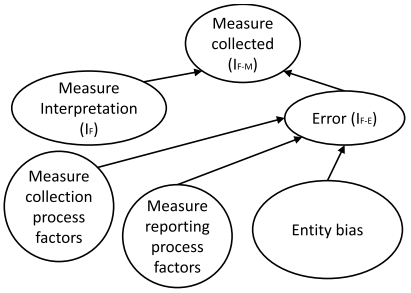


FIGURE 7. Error fragment template.

Step ii-e: if it makes sense, apply the synthesis idiom [48] and decompose the *Error* node (I_{F_i-E}) into: *measure collection process factors*, *measure reporting process factors*, and *entity bias*. We call this structure the Error fragment template (see Figure 7). Continuing the example started in Figure 6, Figure 8-1 presents the result of applying the Error fragment template. The *measure collection process factors* is represented by *test process quality* and *measure reporting process factors*, by *error report quality*. For this case, there are no *entity bias* to be considered. However, the user is free to modify the template given the measurement context. In this case, we considered that the user found valuable to represent *Test bias*, which is related to *measure collection process factors* separately. Further, it is important to notice that the

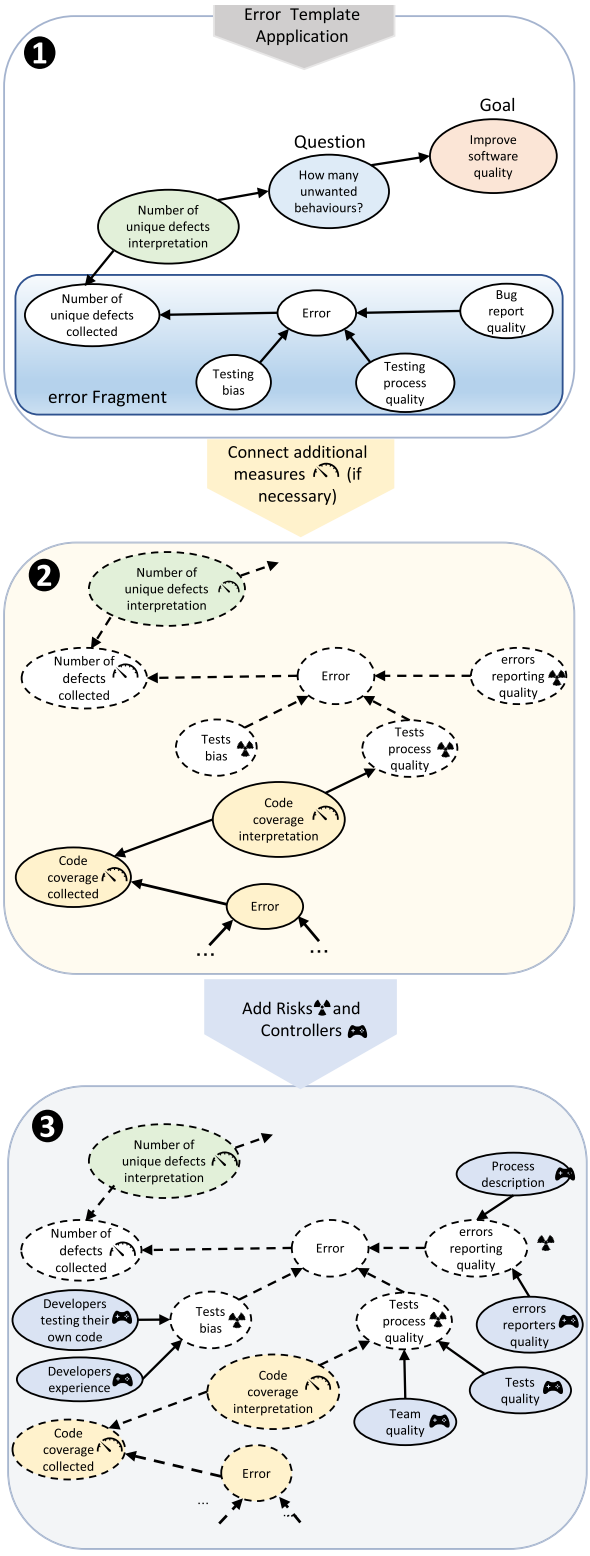


FIGURE 8. Example of applying the Error fragment template.

measure used in the template is *Number of defects*, because, in this example, it is the only measure belonging to the first level.

Step ii-f: For each parent P_E of I_{F_i-E}, create a set of risks R and, for each element $r \in R$, add a relationship $p(r, P_E)$ to P.

Risks can exist due to tools, teams, activities, or the collection of other measures. The measurement users are responsible for identifying the risks since they know the measurement program's context (i.e., organization and project information).

Step ii-g: If necessary, for each risk $r \in R$, apply the synthesis idiom, and decompose it into factors that the practitioners can observe (e.g., activity, practice or characteristic). This step is similar to the process of breaking down questions into subquestions, in GQM. For instance, say that r_1 represents a factor associated with a team. In this case, r_1 could be decomposed into *Communication*, *Experience* and *Expertise*. For every $r \in R$, a node must be added to the DAG. In case another measure m contained in the next level of S_i ($m \in L_{l+1} \wedge m \neq F_i \wedge F_i \in L_l$) influences the measure f_j, repeat steps *ii-a* to *ii-d* using m instead of f_j and add a relationship $p(I_m, I_{f_j-E})$ to P. For instance, in the example shown in Figure 8-2, the measure *Code Coverage*, contained in level two, is part of the *measure collection process factor* of the measure *Number of unique defects*, contained in level one.

Step ii-h: For each risk $r \in R$, add a controller c and add a relationship $p(c, r)$ to P. The controller should be something that controls, triggers or mitigates the risk. It is possible for a risk to have multiple controllers. Add one relationship for each controller. For instance, *Developers experience* and *Developers testing their own code* could control the factor *Tests bias*. Figure 8-3 presents some risks and controller nodes corresponding to the example illustrated previously in Figure 8-2.

Notice that, before executing this step, it might be the case that the leaf nodes of the *Bayesian network* can already be considered as a controller. For instance, for the example shown in Figure 8-3, the nodes *Team quality* and *Tests quality* might have been defined as a refinement of the node *Tests process quality* in *step ii-g*. Therefore, while executing *step ii-h*, the user might consider that, semantically, *Team quality* is a controller for the risk of having a low-quality testing process. The same reasoning could have been applied for *Tests quality*.

Step ii-i: Refactor the directed acyclic graph. First, make sure all the relationships $r \in Q$ make sense (e.g., the risk nodes represent a risk to the measurement that they are related to). For performance purposes, group common factors to avoid combinatorial explosion (i.e., children nodes with a high number of parents). As a rule of thumb, attempt to have nodes with at most four parents. For a grouping to be valid, some of its parent node state combinations must be equivalent in terms of their effect on the child node [42]. Furthermore, if a parent node has only one child, the practitioner can evaluate if it is valuable to discard it and connect its child node directly to its parent node. Finally, for understandability purposes, modify the names of the nodes whenever it is appropriate.

3) STEP (*iii*): PROBABILITY FUNCTIONS DEFINITION

The last step (*iii*) is to define the probability functions for all the nodes in the Bayesian networks. For this purpose, there are many alternatives available in the literature, as discussed in Sections II and III-B. If data is available, the probability function parameters can be learned. Otherwise, expert knowledge might be elicited using, for instance, the ranked nodes method [51] or the weighted-sum algorithm [54].

The probability function for the node representing the collected measure (represented by I_{F-M} in Figure 7) is calculated differently than other nodes because it is an instance of the measurement idiom. Therefore, it is necessary to use a partitioned expression [42] with a partition for each possible state of the node I_{F-E}. For instance, assuming that a 3-point Likert scale represents I_{F-E}, there would be three partitions, in which a probability function is defined for each partition.

Depending on the calculated values for I_{F-E}, the resulting value for I_F (i.e., the measure "interpretation" node) is calculated combining the probability functions of I_{F-M} using backpropagation. For instance, given that $I_{F-E} = (Low, 0.3), (Medium, 0.45), (High, 0.25)$, to calculate the values for I_F, it would be necessary to use the weight of 0.3 for the I_{F-M} probability function related to the state "Low" of I_{F-E}; 0.45 for the state "Medium"; and 0.25 for the state "High".

4) STEP (*iv*): MODEL VALIDATION

After having the first complete version of the Bayesian network constructed, which includes its structure (i.e., DAG) and parameters (i.e., probability functions), it is necessary to validate it. For this purpose, we followed the EKEBN method [56], and recommend to execute it with two steps: first, an expert-driven procedure (i.e., model walkthrough) and, if possible, a data-driven one (i.e., outcome adequacy). The model walkthrough aims to obtain a subjective assessment of how well the experts "feel" that the model calculates what it is supposed to estimate, at face value (i.e., face validity) [59]. For this purpose, the experts (i.e., measurement users) must define "what-if" scenarios (i.e., simulated scenarios), establishing a set of inputs and expected outputs, and compare them with the calculated outputs.

The next step is to apply the outcome adequacy method. Notice that, usually, there is no historical data regarding the variables included in the model. For these cases, we recommend applying this step as part of an empirical cycle (e.g., action research) such as the one described by Antinyan *et al.* [36], in which the model would be applied. Still, it would be used with care to support decision-making until the measurement users verify that the model is representative of the reality.

For this purpose, real case scenarios are used to assess, for each case, the frequency (i.e., probability) with which the state (e.g., "Low", "Medium", "High") for the Bayesian network's goal nodes with the highest probability matches the state identified in practice. For instance, if the model indicates that, for a given goal node, the "High" state has the highest probability, and, in practice, it was observed that the goal was "High", this scenario confirms that the model is performing correct goal assessments.

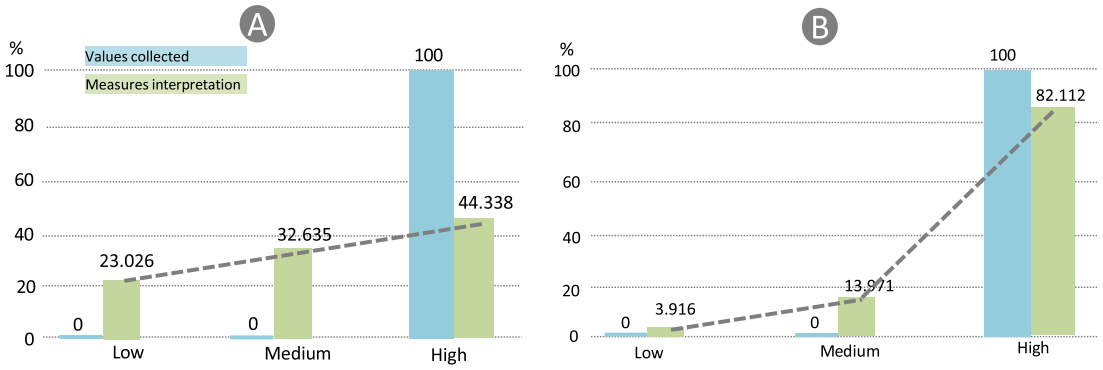


FIGURE 9. Example of scenarios for data collected and "interpreted".

Conversely, suppose the model's calculations do not conform with the real observations. In that case, the measurement user must analyze if such disagreement results from the model's scope, an unexpected outcome, or a model error [60]. Regarding the model scope, the model might purposefully not consider variables that might influence the goal assessment. In this case, the measurement user might decide to clarify the model's scope, maybe including factors that were not initially considered.

An unexpected outcome is a low-probability observed outcome (i.e., it was not expected to happen). In this case, it is concluded that the model's assessment was correct, and no model changes are triggered. However, this case clarifies the importance of understanding the probabilities' meaning represented with the random variables that compose the Bayesian network.

For the third case (i.e., model error), it might be necessary to go back to step (*ii*) and modify the model. Finally, in the cases where there is historical data, the outcome adequacy is applied using a subset of real case scenarios extracted from historical data to verify the Bayesian network's goal nodes.

5) STEP (*v*): MODEL USAGE

After applying the described steps, the Bayesian network is constructed and ready for use. The measurement users should use the model to analyze the reliability of the measures and, consequently, follow the models' causal reasoning and assess the goals.

Figure 9 shows two possible scenarios for the data collected and "interpreted" (i.e., considering its calculated reliability) for a given measure. Since each node in the Bayesian network is a random variable, each possible state is associated with a probability (i.e., probability function). The shape of the probability function for the collected data always has the value of 100% associated for a given state, and 0 for the remaining ones. The exception here would be for the case of using probabilistic thresholds [61]–[64], which the shape of the probabilistic function would follow from the uncertainty of the defined thresholds.

In Figure 9, for both examples, there is a value of 100% associated with the state *High*, meaning that the collected data for the given measure (i.e., the evidence) was associated with the state *High*. The shape of the probability function for the "measure interpretation" varies given the input of the nodes associated with the *Defect* fragment associated with the given measure. In Figure 9, for case (A), the slope of the curve that fits the data is shallow, and there is only 44.338% of the real value being the same as the collected. Such uncertainty is a result of flaws in the measurement collection or reporting process or unhandled risks. As a result, it is recommended to use such a measure for goal assessment with care. Additionally, the measurement user should use the Bayesian network to diagnose the measurement process and improve it. For instance, it could be diagnosed that the significant source of uncertainty is the lack of experienced testers.

Conversely, for case (B), notice that we have a high probability (82.112%) of the "interpreted measure" to be the same as the evidence collected for the measure. In this case, notice that the line that fits the data has a steep slope. Ideally, the slope for such a line should be undefined (i.e., vertical), meaning that the "interpreted" data matches the collected data perfectly; however, this is unrealistic. Therefore, the measurement user should aim for having very steep slopes.

Finally, notice that it is possible, if needed, to use multiple levels of questions. For instance, for a question related to *Software Maintainability*, it might be necessary to add subquestions related to *Analyzability*, *Modularity*, *Modifiability*, *Reusability*, and *Testability*. Even though we have not previously formalized this scenario in the proposed method, the procedure would be analogous, resulting in an additional layer on the Bayesian network for each question layer.

IV. TOOL SUPPORT

This section presents an overview of a Web-based tool, namely, Software Attributes Measurement (SAM), developed to operationalize the proposed method. The development of such a tool, contemplating all the steps of the proposed method, the underlying conceptual models, and the relationship between the information entities, can be considered, in effect, a semi-formal demonstration of the feasibility of

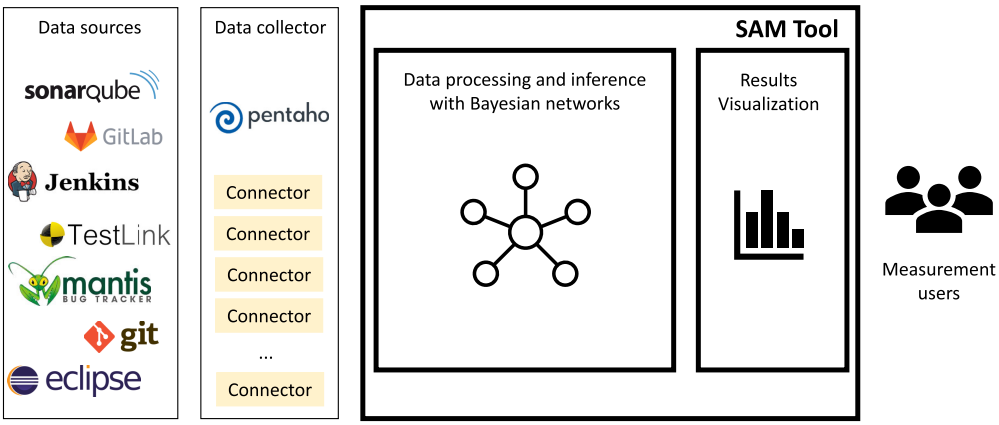


FIGURE 10. SAM tool architecture.

the practical application of the concepts related to modeling the measures' reliability and goal assessment with Bayesian networks, inherent to the proposed method. The main goal of developing SAM was to serve as a proof of concept, supporting the usage of the proposed method and easing technology transfer to the industry.

We developed the SAM tool using the MEAN.JS[1] stack for Web development. For the Bayesian network inferences, we used the SMILE engine[2] and KaizenRNM[3]). Its source code is freely available on GitHub.[4]

Figure 10 presents SAM tool's architecture. Notice that SAM delegates the responsibility of communicating to data sources (e.g., project management, continuous integration, or source code management tools) to a *Business Intelligence* tool. In practice, we implemented such connections using Pentaho.[5] However, theoretically, any tool that exports data using JSON could be easily connected to SAM, which eases the process of automating the process of collecting measures for feeding the Bayesian network.

SAM tool supports applying the proposed method, including a guided construction of the GQM model and measure reliability modes, and configuring the measures' thresholds. For setting the measure's thresholds, it is possible to connect a node from the Bayesian network to a data source, and the number of thresholds is defined given the node's states (e.g., two states for a Boolean node). In SAM, it is possible to define the threshold rules, given the collected data's original scale.

The connection with the data source, through a *Business Intelligence*, is configured for each node of interest. For this purpose, it is only required that the *Business Intelligence* tool is connected correctly and that it exports the data through JSON to be consumed by the SAM tool.

[1]http://meanjs.org/
[2]https://www.bayesfusion.com/smile/
[3]https://github.com/isevirtus/KaizenRNM
[4]https://github.com/isevirtus/SAM
[5]https://www.hitachivantara.com/en-us/products/data-management-analytics/pentaho-platform.html

V. CASE STUDY

To evaluate the proposed method in terms of adoption potential, we applied it to an industrial case study. According to Runeson and Höst, case studies provide a more in-depth understanding of phenomena under study in its real context [65]. We performed a case study with four industrial software development projects at VIRTUS,[6] a Research, Development, and Innovation Center in Information Technology, Communication, and Automation, in Campina Grande, Brazil. VIRTUS was chosen due to existing academia-industry cooperation projects executed by professionals and in partnership with companies to develop market software products. Therefore, our evaluation context is on real industry projects, and the duration of the case study was 45 days. In what follows, this section characterizes the case study's context (Section V-A), design (Section V-B), results (Section V-C), and threats to validity (Section V-D).

A. CONTEXT CHARACTERIZATION

This section presents information to characterize VIRTUS following the context facets described by Petersen and Wohlin [66]: *product*, *process*, *practices and techniques*, *people*, *organization* and *market*.

VIRTUS is a research, development, and innovation nucleus focusing on information, communication, and automation technology. It is a supplementary organization of the Federal University of Campina Grande (UFCG). The organization manages hundreds of engineers and researchers, colocated in its headquarters in Campina Grande, Brazil.

VIRTUS was founded in 2015 by researchers from UFCG with over fifteen years of experience in research and development projects and executes projects in several technological domains, including data science, Web systems, mobile systems, artificial intelligence, augmented reality, embedded systems, and hardware. Furthermore, the projects focus on diverse market segments, including security, biometry, and business intelligence. The projects usually last between ten

[6] https://www.virtus.ufcg.edu.br/

and eighteen months and result from incentive mechanisms between academy and industry promoted by the Brazilian government. Typically, between forty and fifty projects are executed per year, having industry partners such as Asus, Envision, Ericsson, and many other small, medium, and large-size technology companies.

From the structural perspective, VIRTUS can be considered to be hierarchical and project-oriented. However, it has a quality department responsible for defining the guidelines for the projects' quality process - including the measurement system - and audit them. During the study's execution, VIRTUS was not certified by quality models such as ISO 9001 or maturity models such as CMMI or MPS.Br. However, its list of industry partners is an indicator of its development process's quality and maturity.

In general, the projects are executed using agile approaches such as Scrum or Kanban. The development practices and tools follow the guidelines defined for the organization but are adapted given the needs of the projects (e.g., programming language and type of system). VIRTUS uses a proprietary tool to support project management, which integrates requirement, test and issue management, source code repositories, and software build systems. Such a tool enables the traceability of the development artifacts following a model similar to the *Agile Traceability Information Model*, popular in agile management tools (*e.g.,* Rally[7]).

B. DESIGN

We executed the case study following the guideline presented by Runeson and Höst [65]. In what follows, Sections V-B1, V-B2, and V-B3 present, respectively, the objectives, units of analysis and subjects, and procedures of the case study.

1) OBJECTIVES

The objective of the case study to *analyze the proposed method concerning its adoption intention from the viewpoint of project leaders*. For this purpose, we used as the ground theory for defining the research questions the *Technology Acceptance Model* (TAM) construct [67]. TAM is composed of two constructs: *Perceived usefulness* and *Perceived ease of use*. Given this, we defined the following research questions:

- ***RQ1*** What are the perceived benefits of using the proposed method for supporting managerial decision making at VIRTUS?
- ***RQ2*** What is the perceived effort in adopting the method at VIRTUS?

RQ1 relates to TAM's *Perceived usefulness* construct and takes into consideration that software measures should be used for supporting decision-making [23]. ***RQ2*** relates to TAM's *Perceived effort* construct.

2) UNITS OF ANALYSIS AND SUBJECTS

The case study had four units of analysis. Each unit of analysis was a software development project from VIRTUS. All projects are composed of small teams. We present details regarding the team's composition in what follows:

- *Project 1* with ten members, one project leader, seven developers, and two testers;
- *Project 2* with six members, one project leader, three developers, and two testers;
- *Project 3* with eleven members, two project leaders, six developers, and three testers;
- *Project 4* with twelve members, two project leaders, eight developers, and two testers.

All projects focused on developing Android-based applications for mobile devices (i.e., smartphones and tablets). Project 1's goal was to develop a prototype that applies video playback techniques to assist viewers while watching videos. Project 2 and 4's goal was to deliver an application integrated with wearable devices, such as smartwatches and smart wristbands. Project 3's goal was to implement a distributed volunteer computing application to help to simulate biomolecular phenomena [68]. Also, all projects used iterative and incremental development processes and *Scrum* as the project management framework.

For each unit of analysis (i.e., project), the subjects were project leaders. At VIRTUS, the project leaders executed tasks related to the process and team leadership (i.e., management, as a *Scrum Master*), design, and implementation. Table 1 presents the profile of the subjects. By analyzing their profile, we noted that they had few or no experience with GQM, despite most of them being experienced software engineers.

TABLE 1. Subjects profile.

	Subjects			
Characteristic	**1**	**2**	**3**	**4**
Experience in years working in software development projects	10	4	3	1
Experience in years leading software development projects	25	20	8	1
Experience in years using software measures to support decision making	11	5	5	0
Experience in years using GQM	8	2	2	2

3) PROCEDURE

For each unit of analysis, we executed the procedure presented in Figure 11, which is divided into two phases.

a: MODELS CONSTRUCTION PHASE

During this phase, the subjects learned basic concepts regarding the method. Afterward, with one of the researchers' assistance, they built GQM models and applied the proposed method to construct the *Bayesian networks*. The following activities were executed:

Introductory course - A researcher acted as a trainer, explaining the concepts necessary to apply the proposed method, e.g., GQM and *Bayesian networks*. Basic concepts and examples were presented. Some questions addressed

[7]http://www.rallydev.com

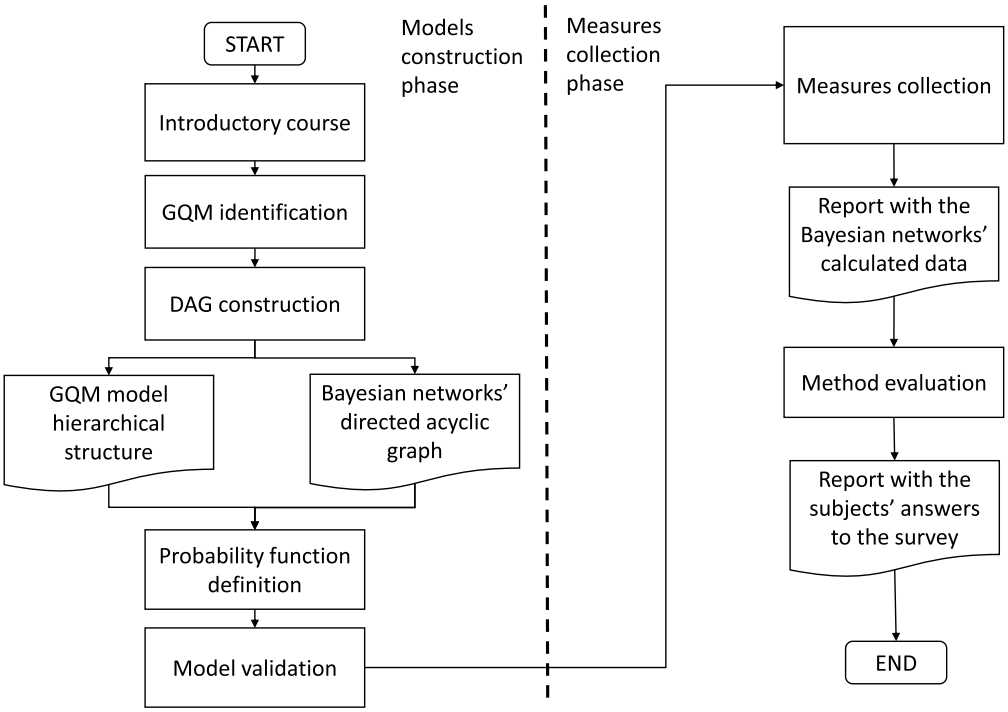


FIGURE 11. Procedure applied for each case study's unit of analysis.

were: "How to define goals?", "Given the goals, how to define questions and measures?", "What is a *Bayesian network*?", "How to apply the method?" and "When to stop decomposing the factors?".

GQM identification - A researcher assisted the subjects in executing the method's step (*i*), which consisted of building the GQM model hierarchical structure. For all nodes, the subjects decided to use an ordinal scale.

DAG construction - A researcher assisted the subjects in executing the method's step (*ii*), which consisted of building the *Bayesian networks*' DAG. For all nodes, the subjects decided to use an ordinal scale.

Probability function definition - In this activity, the method's step (*iii*) was executed. For this purpose, since all the nodes were ordinal, we applied the ranked nodes method (RNM) [51]. In RNM, each node is represented as a doubly truncated normal distribution (TNormal distribution). The TNormal distribution is characterized by two parameters: mean (μ) and variance (σ^2). μ is calculated by a weighted expression that reflects the parents' influence on a given node. μ is calculated by a weighted expression that reflects the parents' influence on a given node. There are four weighted expressions: weighted mean, weighted minimum, weighted maximum, and mixed min-max. σ^2 represents the uncertainty about the results.

For configuring the parameters μ and σ^2, we applied the procedure presented in Perkusich *et al.* [49]. The first step was to, for each child node, order the relationships between them and their parents their relative magnitude. For example, if a node B has a more significant influence on A than C, the order

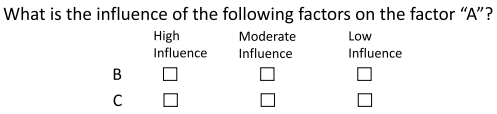


FIGURE 12. Example of survey question for a node "*A*" .

should be $B-A$ and $C-A$, in which $B-A$ is the relationship between B and A while $C-A$ is the relationship between C and A. This is done by collecting data through a survey with a question for each child node in the DAG. All the survey questions followed the same format, as shown in Figure 12, in which B and C are parents of A.

For instance, considering that a practitioner defined that the influence of *Functional tests quality* (B) on *Tests quality* (A) is greater than the influence of *Non-functional tests quality* (C). In this case, the weighted expression representing these relationships would be assumed to be $A = 2 \times B + C$. Afterward, the subjects defined which type of weighted expression to be used to generate the probability functions. Finally, they defined the nodes' variance.

Model validation - In this activity, the method's step (*iv*) was executed. After having the Bayesian networks constructed, the subjects defined simulated scenarios, representing their experience with software projects, to perform the model walkthrough (i.e., face validity). After this task, no modifications were performed in the models. In this study, we did not execute the outcome adequacy, given that there was no historical data regarding the models' variables. The Bayesian networks' calculations were analyzed during the case study.

b: MEASURES COLLECTION PHASE

This phase lasted 45 days, or three iterations (i.e., sprints). During this phase, the subjects entered evidence into the models, used the method (step v), and evaluated it. The following activities have been planned:

Measures collection - At the end of each iteration, the subjects and a researcher met. During this meeting, the subject inputted measures collected into the Bayesian networks as evidence of the corresponding nodes (i.e., the nodes representing the data collected for the given measures). Furthermore, the subjects inputted evidence regarding the reliability factors and risks' controllers (i.e., leaf nodes). Finally, the subjects and one researcher analyzed the calculated data.

Measures evaluation - A post mortem structured interview was executed in which we collected qualitative data from the subjects to evaluate the proposed method.

Given the study's research questions, during the method's evaluation, we needed to measure the variables *Perceived benefits* and *Perceived effort*. In what follows, we present how we measured them.

Perceived benefits is the extent to which the person thinks using the system enhances his or her job performance [69]. To measure the perceived benefits of the proposed method, during the post mortem structured interview sessions, we asked two questions: ''Did using the proposed method increase the confidence in the decision-making regarding the goals' assessment?'' and ''Did the use of the proposed method support decision-making to improve the software process?''. For both questions, we collected open-ended answers (qualitative data). Furthermore, we considered the artifacts constructed by the subjects during the case study.

Perceived effort consists of analyzing the effort (i.e., cost) in adopting a solution. To measure the perceived effort of using the proposed method, we asked the subjects, ''How do you analyze the effort of applying the proposed method?''. Thus, we collected an open-ended answer (i.e., qualitative data). We also considered the period taken to execute the proposed method's tasks during the case study (i.e., quantitative data).

It is worthy of mentioning that, traditionally, when using the TAM construct, the data is collected through surveys (i.e., questionnaires) using predefined instruments with close-ended answers and statistically analyzed. However, given that we only collected data from four subjects, for having richer data, we decided to use open-ended questions.

Therefore, we used a mixed-methods approach to collect the data for the case study. For this purpose, we used methods such as interviews, questionnaires, and project data. We used an unstructured interview to collect information from the subjects for building the GQM model hierarchical structures and the Bayesian networks' DAG. We used a questionnaire to collect data and define the probability functions. The subjects collected data from the project to input as evidence on the models. Finally, we used a structured interview to collect data from the subjects for evaluating the proposed method.

C. RESULTS AND DISCUSSION

This section discusses the results of executing the proposed method by following the procedure presented in Section V-B3 to four projects. Section V-C1 presents the constructed artifacts and Sections V-C2 and V-C3 discuss, respectively, the results for ***RQ1*** and ***RQ2***.

1) CONSTRUCTED ARTIFACTS

For each project, we constructed two artifacts: a GQM model hierarchical structure and the Bayesian network. All projects shared the goal of assessing product quality. All projects, except Project 3, shared the goal of assessing the team's productivity. On the other hand, since the projects have different contexts, different measures were identified, which, as a consequence, resulted in different models.

The Bayesian networks for Project 1 had 70 nodes; for Project 2, 45; for Project 3, 63; and for Project 4, 56. The number of nodes reflects the number of measures chosen and the details of the process (i.e., reliability factors and risks) that the subjects modeled, following the steps presented in Section III. Table 2 presents the measures defined for each project.

TABLE 2. Measures used in the analyzed projects.

	Project			
Measure	**1**	**2**	**3**	**4**
Test Status	No	No	Yes	No
Number of tests	Yes	No	Yes	No
Number of defects	Yes	Yes	Yes	Yes
Code coverage	Yes	No	Yes	Yes
Static code analysis warnings	Yes	Yes	Yes	Yes
Checkstyle warnings	Yes	No	No	Yes
Sprint burndown	Yes	Yes	No	Yes
Javadoc warnings	No	No	Yes	No

Due to space limitations, this article only presents the artifacts and resulting analysis for Project 1, which was the one with the most used measures and largest models. Figure 13 shows the resulting GQM model hierarchical structure. Figures 14 and 15 shows resulting the Bayesian networks' DAG. Figures 16 and 18 presents data regarding the independent measures from the report generated with the calculated data. The constructed artifacts and the data for the remaining projects are publicly available[8].[9]

By analyzing Figure 16, we concluded that for this project, the confidence regarding *Number of defects* was low to moderate. Such reasoning was a logical consequence of the observed difference between the collected and interpreted values. For instance, for Sprint 1 (i.e., first iteration), the collected value was 100% *Medium* and the interpreted, 35.482% *Medium*. Such a scenario means that even though the *Number of defects* collected was within the threshold *Medium*, after considering the reliability factors (i.e., accuracy defects), we were not confident about the real status of *Number*

[8]http://bit.ly/BNValidFiles

[9]https://1drv.ms/b/s!AojD1Jf8hcECiM83989tusS3XTEVeQ?e=ap77aq

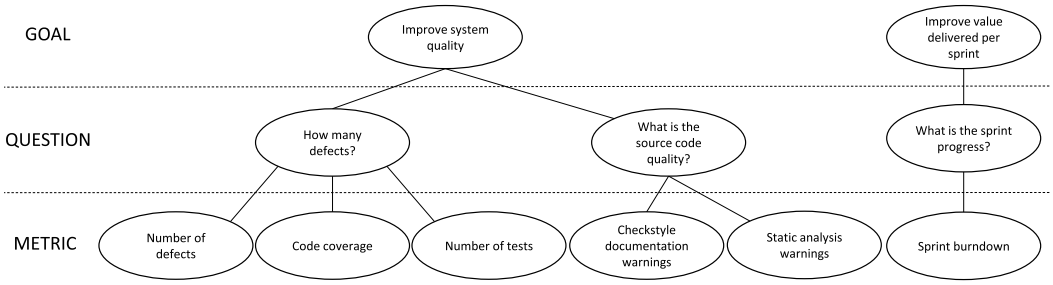


FIGURE 13. GQM model hierarchical structure used by Project 1.

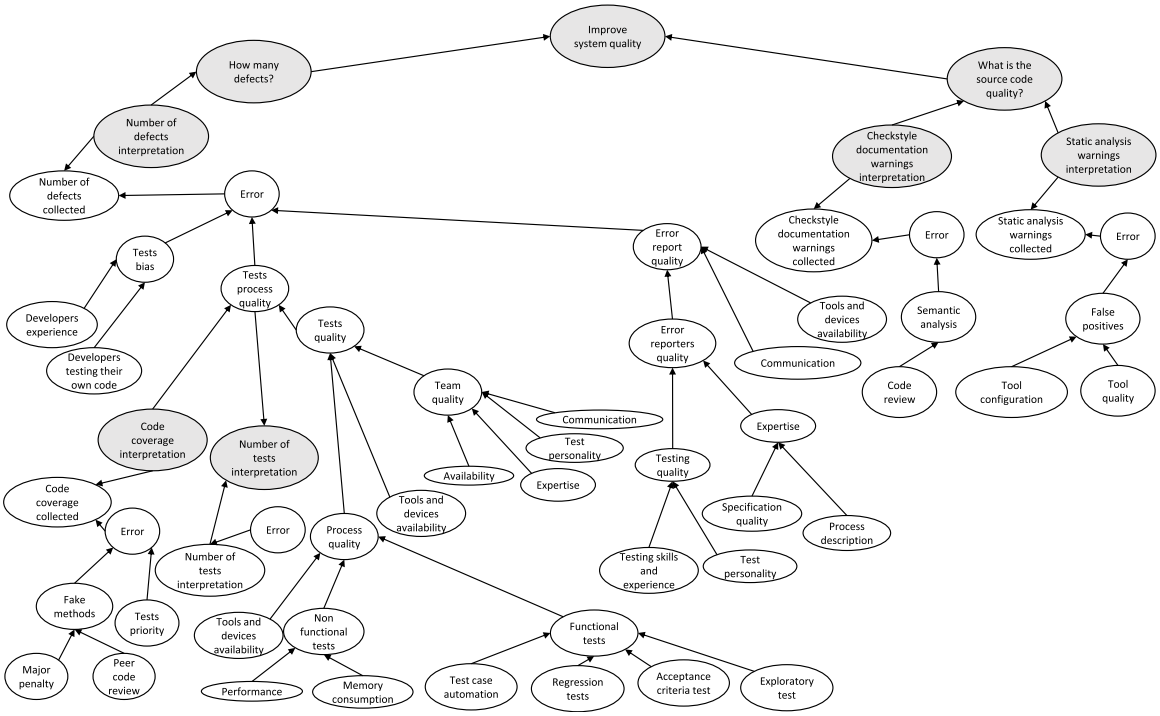


FIGURE 14. Complete DAG constructed for Project 1 goal *Evaluate system quality*.

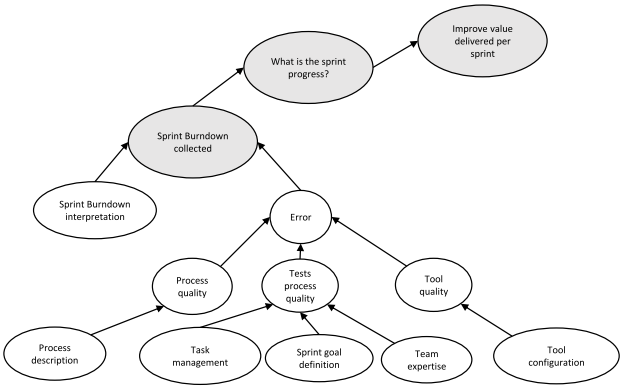


FIGURE 15. Complete DAG constructed for Project 1 goal *Evaluate value delivered per sprint*.

of defects. The explanation for such reasoning was because its probability of being *Low*, *Medium*, or *High* was similar (i.e., the curve fitting the data had a shallow slope).

The subject concluded that this confidence reflects the lack of proper testing practices, issues with tools, and devices' availability for testing purposes. The confidence regarding *Static code analysis warnings* was also low to moderate. This confidence reflected issues such as false positives caused by tools' misconfigurations. On the other hand, the confidence for *Checkstyle documentation warnings* was high. This level of confidence was reflected in the calculated values for the goal *Evaluate system quality* shown in Figure 17. The calculated results implied that given the project's situation, the *ScrumMaster* should be careful to answer the questions *"How many defects?"* and *"What is the source code quality?"*, and, therefore, assess the goal *Evaluate system quality*.

On the other hand, by analyzing Figure 18, we concluded that the confidence regarding *Sprint Burndown* was high, because the curve fitting the data had a steep slope. This level of confidence was reflected in the calculated values for the goal *Improve value delivered per sprint* shown in Figure 17. Therefore, the *ScrumMaster* should be confident to answer

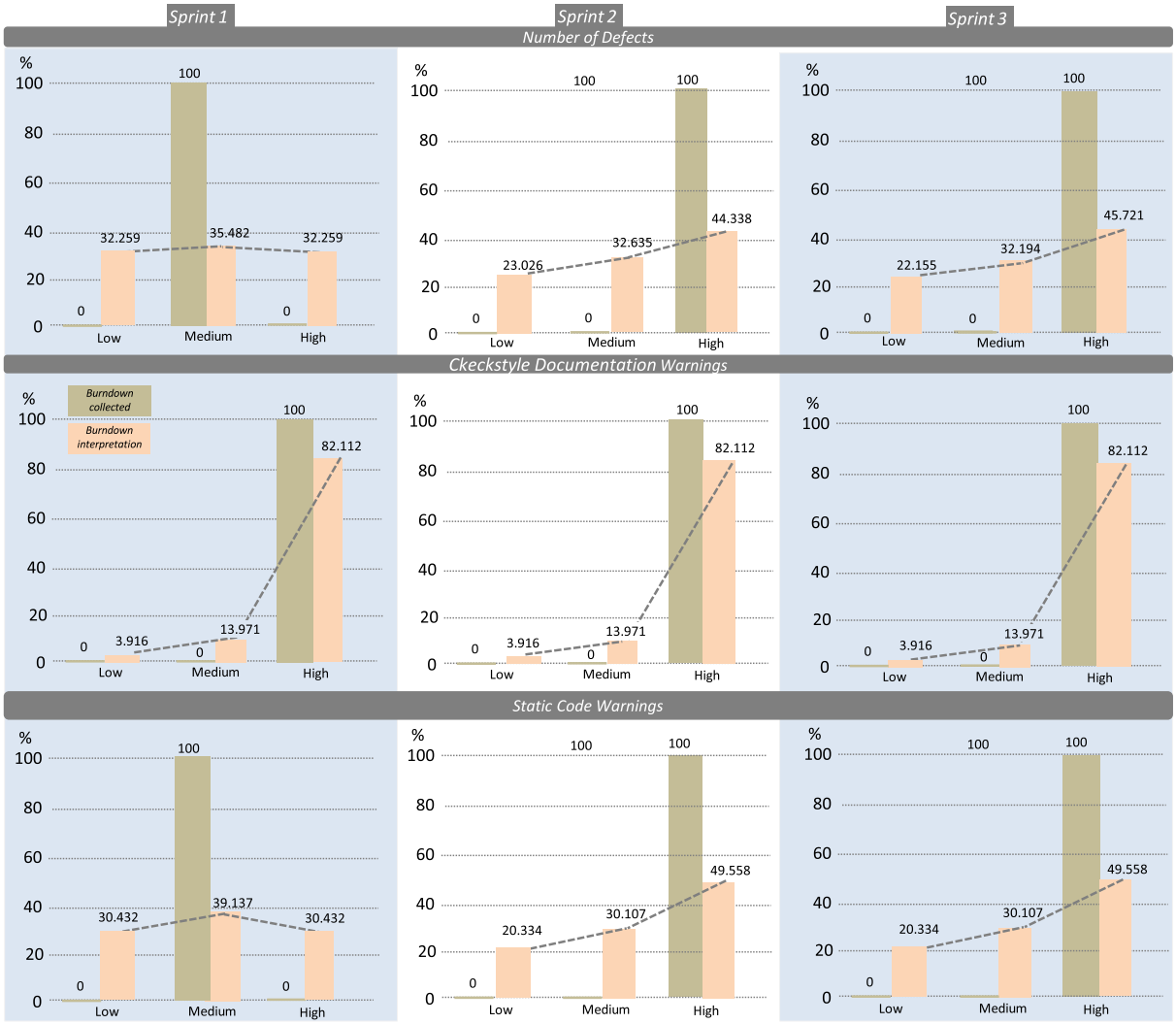


FIGURE 16. Collected and interpreted measures for Project 1 goal *Evaluate system quality*.

the question *What is the sprint progress?* and assess the goal *Improve value delivered per sprint*.

2) PERCEIVED BENEFITS OF APPLYING THE PROPOSED METHOD (*RQ1*)

This section discusses ***RQ1***, which explored the perceived benefits of applying the proposed method. As discussed in Section V-B3, after constructing the artifacts, we interviewed the subjects and asked them two questions related to this research question. Tables 3 and 4 present the answers for both questions.

After the interview, Subject 4 complemented his answers by claiming that, before applying the proposed method, he did not trust the measure "*Number of defects*". Using the proposed method and analyzing the data, he performed a root-cause analysis and identified improvement opportunities for the testing process. Furthermore, he claimed that the Bayesian network helped to visualize the process as a whole to interpret the measures, and, consequently, assess the goals. Finally, he said that, as a result of the analysis, he started to research better tools to manage the development process and for static code analysis.

We analyzed the collected answers (see Tables 3 and 4) using an inductive codification approach. As a result, we identified four benefits: *Externalization*, *Diagnosis support*, *Measure interpretation improvement* and *Decision-making support*. *Externalization* refers to transforming tacit knowledge, regarding the process and measures, into explicit knowledge through the Bayesian network. Such benefit is supported by Subject 1 saying that the proposed method "...showed something that we already had a feeling." Subject 4 claimed that "With the method, we have concrete data."

Diagnosis support, which is a known characteristic of Bayesian networks [70], refers to supporting root-cause analysis to diagnose the source of the uncertainty in the measure's interpretation. Such benefit was claimed by Subject 3 when he said that "...It provided a very useful insight into what was happening on our project." and that "it helped to

FIGURE 17. Calculated values for Project 1 goals.

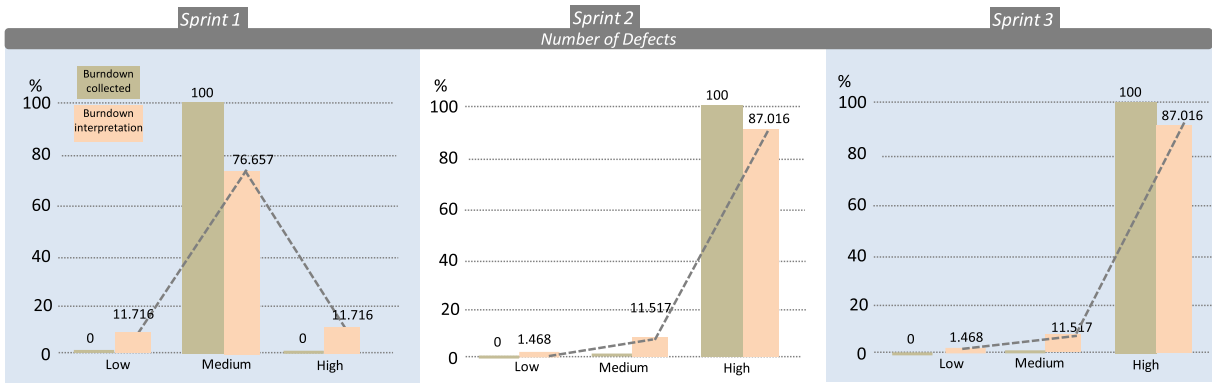


FIGURE 18. Collected and interpreted measures for Project 1 goal *Evaluate value delivered per sprint*.

identify problems by having a look in the big picture''. Further, Subject 4 claimed that, by using the proposed method, his team could ''identify where the main problems are...''. Subject 4 also gave a detailed example of a root cause analysis performed by the team: ''...we knew that we had issues with tests, but we could not objectively identify the root causes. After using the method, it is clear that the problem is that the team members do not have enough experience writing tests.'' Subject 1 claimed that using the proposed method gave them a ''deeper understanding of issues in our process.'' Such analysis is also backed by analyzing the subjects' Bayesian networks, which were detailed enough to enable the subjects to diagnose the measurement process.

Having *Diagnosis support*, leads to *Decision-making support*. Subject 3 claimed that ''...Based on this feedback, we focus on some metrics in favor of others. For the ones that demonstrated fewer confidence levels, we had internal discussions to find out what could be the reason for such results.'' Subject 1 and 4 suggested that the proposed method supported improving the process. Further, using the proposed method also had the benefit of supporting *Measure interpretation improvement*. Subject 3 claimed that, by analyzing the data generated using the proposed method, his team

TABLE 3. Answers to question "Did using the proposed method increase the confidence in the decision-making regarding the goals' assessment?".

Subject	Answer
1	Yes, because it showed something that we already had a feeling. More specifically, somehow, we knew that we could not trust the metric Number of defects. We were always careful about alerting the client that even though there were not too many reported defects; it did not mean that the product did not have unwanted behaviors
2	We rely more on the quality of the test process than on the number of defects. We have many known issues caused by 3rd part libraries that are defects but cannot be fixed and do not affect the general quality
3	Yes. It provided a very useful insight into what was happening on our project. Based on this feedback, we focus on some metrics in favor of others. For the ones that demonstrated fewer confidence levels, we had internal discussions to determine the reason for such results. Then we worked on fixing some aspects that were decreasing the confidence level in these metrics
4	With the method, we have concrete data. The method explicitly shows improvement opportunities. For instance, we knew that we had issues with tests, but we could not objectively identify the root causes. After using the method, it is clear that the problem is that the team members do not have enough experience writing tests

TABLE 4. Answers to question "*Did the use of the proposed method support decision-making to improve the software process?*".

Subject	Answer
1	Yes, because now we have a deeper understanding of issues in our process, then we can continuously improve the process
2	It provides a framework to produce meaningful results of bug reporting raw data; it is probably better for projects that are bigger and have a constant stream of incoming bug reports
3	It helped to identify problems by having a look in the big picture
4	We can identify where the main problems are, and elaborate action points to improve the process

"worked on fixing some aspects that were decreasing the confidence level in these metrics."

A possible adoption limitation was pointed about by Subject 2 when he claimed that using the proposed method "is probably better for projects that are bigger and have a constant stream of incoming bug reports." However, it is worthy to notice that the given project's goal was to develop a prototype with less emphasis on product and process quality.

> ***RQ1***: *The proposed method has the benefits of Externalization, Diagnosis support, Measure interpretation improvement and Decision-making support improvement.*

3) PERCEIVED EFFORT OF APPLYING THE METHOD (***RQ2***)

This section discusses ***RQ2***, which explored the perceived effort of applying the proposed method. On average, the introductory course lasted 30 minutes for each project; the DAG construction meeting, 1 hour; the probability function definition, 14 minutes. Furthermore, the measures collection meeting lasted, on average, 25 minutes. Therefore, on average, the practitioners' total effort to execute the proposed method was 2 hours and 9 minutes. Furthermore, as discussed in Section V-B3, after constructing the artifacts, we interviewed the subjects and asked them one question related to this research question. Tab 5 presents the answers for such a question.

TABLE 5. Answers to question "*How do you analyze the effort of applying the proposed method?*".

Subject	Answer
1	It was worth it because the models provided will be used to define a roadmap of corrective actions
2	I believe that projects with different characteristics than mine would have a better cost-benefit relation
3	After the first evaluation, which takes more effort, the following usages of the method are really straightforward and takes almost no effort
4	The time invested in constructing the Bayesian networks and inputting data were overweighted by the analysis performed. Now we have more data to assist in decision-making and define corrective actions to improve the team's performance. If the team improves, the process improves, control over the results improves, the product quality increases, and, finally, customer satisfaction increases

By analyzing the answers shown in Table 5, notice that Subjects 1, 2, and 4 concluded that the effort to use the method was overweighted by the benefits provided. Further, it is worthy to notice that Subject 3 and 4 indicated that applying the steps (i) - (iv) of the proposed method was costly. Such observations are consistent with the time spent to execute such steps. We registered an average of one hour and fourteen minutes to build the Bayesian networks. Afterward, for each iteration, the subjects spent 25 minutes to use the constructed Bayesian networks. Considering that they work a regular 40 hours/week, it took 0.5% of their work capacity. Further, notice that, for this study, the data were collected manually. However, at least part of the data could be collected automatically.

On the other hand, subject 2 judged that it was not worthy of using his project method. As already mentioned, in project 2, since the goal was to build a prototype, it was not worth spending effort on process and quality improvement activities. Therefore, we believe that his perception is not specifically for the proposed method but for any software measurement initiative.

> ***RQ2***: *The initial effort to use the proposed method is considerable, given the need to construct the models. However, the effort of using the Bayesian networks is low, even with manual data collection. It is worth mentioning that using the method might not be worth it for projects to deliver prototypes (or proof of concept).*

D. THREATS TO VALIDITY

This section discusses the threats to the validity of the case study in light of the classification presented by Runeson and Höst [65] and Yin *et al.* [71]. Therefore, we classified

the threats to validity as internal, construct, external, and reliability.

The internal validity relates to how much the study's conduction assures that the causal relationship between treatment and outcome, if any, is not a result of a factor not controlled by the researcher. In our case, during the DAG construction (step ii), given its duration, the subject might have fatigued and, consequently, built low-level models. To minimize this threat, we paused whenever necessary and used the researchers' prior knowledge with Bayesian networks to construct them. A threat to the study's internal validity is related to the researchers' presence, using the proposed method, especially constructing the Bayesian networks. However, it is a characteristic of the proposed method to assume that the measurement analyst has enough knowledge on applying Bayesian networks.

The construct validity refers to the relationship between theory and observation, assuring that the treatment reflects the cause's construct, and the outcome reflects the construct of the effect. In other words, the construct validity relates to how much the case study measured what it was supposed to. The study's conclusions were based on interview analysis and the data collected during the events and artifacts constructed during the case study. We minimized this threat by using the TAM construct and having open-ended questions to collect data from the subjects reviewed by an independent researcher to avoid bias.

The external validity refers to the ability to generalize the results given the sample of the population. As discussed previously, the outcomes of the study cannot be statistically generalized. However, Sections V-A and V-B2 detail the context in which the proposed method was applied.

The reliability refers to how much the data and analyzes depend on the researchers. We minimized this threat by having the data collected reviewed by the study's subjects. Finally, the generated codes from the inductive analysis of the qualitative data were peer-reviewed by the researchers.

VI. FINAL REMARKS

This article presents a method, integrated with GQM, to support validating the data from software measurement programs. The proposed method relies on the concept of measure reliability, which we defined as ''the degree in which a collected number or symbol represents the real value for a given measure''. The proposed method conforms with ISO/IEC 15939:2007 and uses Bayesian networks to model the reliability of measures. We developed a tool to support adopting the proposed method, which is freely available. Further, we evaluated the proposed method regarding its adoption intention with the project leaders from four software development projects through a case study. The case study results indicated that, for projects that need to focus on product quality and meet time-to-market, the benefits of using the proposed method outweigh the effort to apply it. We identified four benefits of using the proposed method: *Externalization*, *Diagnosis support*, *Measure interpretation improvement* and *Decision-making support*. Therefore, our findings suggest that there was a positive intention in adopting the proposed method in practice.

A limitation of our results is that it was only applied to VIRTUS, which might have biased our conclusions regarding the proposed method's adoption potential. Such reasoning follows from the fact that the personnel at VIRTUS are used to dealing with innovation daily, which might influence their positive attitude towards the innovativeness of the proposed method. Further, a characteristic of the proposed method is the need to have the measurement analyst to have a background in Bayesian networks. However, Bayesian networks have been widely used for software measurement [72]–[74], and it is important to point out that having a champion and a guru is known to be a key factor for having successful measurement programs [24], [75]. An external measurement guru helps raise initial enthusiasm for the program, while the champion helps sustain it. Both can introduce and/or improve team maturity and engagement on new knowledge and skills. Another benefit if using Bayesian networks as the foundations for the software measurement process models is the potential to reuse the existing models available in the literature.

Our work focused on improving the measures' trustworthiness, which is the main factor for maximizing the chances of success of a measurement program's long-term use in agile software development. As a logical consequence, we aimed to improve the interpretability of software measures. Our future work will involve integrating and evaluating the proposed method combined with measure validation and thresholds definition approaches. Further, we will explore how to ease the process of constructing Bayesian networks for software measurement by reusing the knowledge already available in the literature by applying reuse-driven software engineering principles such as reusing components, API, frameworks, and libraries.

REFERENCES

[1] L. G. Wallace and S. D. Sheetz, ''The adoption of software measures: A technology acceptance model (TAM) perspective,'' *Inf. Manage.*, vol. 51, no. 2, pp. 249–259, Mar. 2014.

[2] L. Finkelstein and M. S. Leaning, ''A review of the fundamental concepts of measurement,'' *Measurement*, vol. 2, no. 1, pp. 25–34, Jan. 1984.

[3] L. M. Laird and M. C. Brennan, *Software Measurement and Estimation: A Practical Approach*, vol. 2. Hoboken, NJ, USA: Wiley, 2006.

[4] K. S. Mathias, J. H. Cross, T. D. Hendrix, and L. A. Barowski, ''The role of software measures and metrics in studies of program comprehension,'' in *Proc. 37th Annu. Southeast Regional Conf. (CD-ROM)*. New York, NY, USA: Association for Computing Machinery, 1999, p. 13. [Online]. Available: https://doi.org/10.1145/306363.306381

[5] I. C. Society. (2009). *IEEE Standard Adoption of ISO/IEC 15939:2007—Systems and Software Engineering—Measurement Process*. [Online]. Available: https://www.iso.org/standard/44344.html

[6] B. Kitchenham, S. L. Pfleeger, and N. Fenton, ''Towards a framework for software measurement validation,'' *IEEE Trans. Softw. Eng.*, vol. 21, no. 12, pp. 929–944, Dec. 1995.

[7] L. C. Briand, C. M. Differding, and H. D. Rombach, ''Practical guidelines for measurement-based process improvement,'' *Softw. Process, Improvement Pract.*, vol. 2, no. 4, pp. 253–280, Dec. 1996.

[8] O. Gómez, H. Oktaba, M. Piattini, and F. García, "A systematic review measurement in software engineering: State-of-the-art in measures," in *Software and Data Technologies*. Berlin, Germany: Springer, 2006, pp. 165–176.

[9] T. Tahir, G. Rasool, and C. Gencel, "A systematic literature review on software measurement programs," *Inf. Softw. Technol.*, vol. 73, pp. 101–121, May 2016.

[10] R. S. Pressman, *Software Engineering: A Practitioner's Approach*. London, U.K.: Palgrave Macmillan, 2005.

[11] F. García, M. Serrano, J. Cruz-Lemus, F. Ruiz, and M. Piattini, "Managing software process measurement: A metamodel-based approach," *Inf. Sci.*, vol. 177, no. 12, pp. 2570–2586, Jun. 2007.

[12] M. J. Ordonez and H. M. Haddad, "The state of metrics in software industry," in *Proc. 5th Int. Conf. Inf. Technol., New Generat. (ITNG)*, Apr. 2008, pp. 453–458.

[13] C. R. Prause and A. Hönle, "Emperor's new clothes: Transparency through metrication in customer-supplier relationships," in *Product-Focused Software Process Improvement*, M. Kuhrmann, K. Schneider, D. Pfahl, S. Amasaki, M. Ciolkowski, R. Hebig, P. Tell, J. Klünder, and S. Küpper, Eds. Cham, Switzerland: Springer, 2018, pp. 288–296.

[14] M. Choras, T. Springer, R. Kozik, L. Lopez, S. Martinez-Fernandez, P. Ram, P. Rodriguez, and X. Franch, "Measuring and improving agile processes in a small-size software development company," *IEEE Access*, vol. 8, pp. 78452–78466, 2020.

[15] M. Choraś, R. Kozik, D. Puchalski, and R. Renk, "Increasing product owners' cognition and decision-making capabilities by data analysis approach," *Cognition, Technol. Work*, vol. 21, no. 2, pp. 191–200, May 2019, doi: 10.1007/s10111-018-0494-y.

[16] S. Martínez-Fernández, A. M. Vollmer, A. Jedlitschka, X. Franch, L. López, P. Ram, P. R. Marín, S. Aaramaa, A. Bagnato, M. Choras, and J. Partanen, "Continuously assessing and improving software quality with software analytics tools: A case study," *IEEE Access*, vol. 7, pp. 68219–68239, 2019, doi: 10.1109/ACCESS.2019.2917403.

[17] P. Ram, P. Rodriguez, M. Oivo, and S. Martinez-Fernandez, "Success factors for effective process metrics operationalization in agile software development: A multiple case study," in *Proc. IEEE/ACM Int. Conf. Softw. Syst. Processes (ICSSP)*, O. Armbrust and R. Hebig, Eds., Montreal, QC, Canada, May 2019, pp. 14–23, doi: 10.1109/ICSSP.2019.00013.

[18] P. Ram, P. Rodríguez, and M. Oivo, "Software process measurement and related challenges in agile software development: A multiple case study," in *Proc. 19th Int. Conf., Product-Focused Softw. Process Improvement (PROFES)* (Lecture Notes in Computer Science), Wolfsburg, Germany, vol. 11271, M. Kuhrmann, K. Schneider, D. Pfahl, S. Amasaki, M. Ciolkowski, R. Hebig, P. Tell, J. Klünder, and S. Küpper, Eds. Cham, Switzerland: Springer, Nov. 2018, pp. 272–287. [Online]. Available: https://doi.org/10.1007/978-3-030-03673-7_20

[19] M. Manzano, C. Ayala, C. Gómez, A. Abherve, X. Franch, and E. Mendes, "A method to estimate software strategic indicators in software development: An industrial application," *Inf. Softw. Technol.*, Sep. 2020, Art. no. 106433, doi: 10.1016/j.infsof.2020.106433.

[20] J. M. De Oliveira, K. De Oliveira, and A. Belchior, "Measurement process: A mapping among CMMI-SW, ISO/IEC 15939, IEEE Std 1061, Six Sigma and PSM," in *Proc. Int. Conf. Service Syst. Service Manage.*, vol. 1, Oct. 2006, pp. 810–815.

[21] W. Meding and M. Staron, "Making software measurement standards understandable," in *Proc. Eval. Assessment Softw. Eng.*, Apr. 2020, pp. 368–370.

[22] N. E. Fenton and M. Neil, "Software metrics: Roadmap," in *Proc. Conf. Future Softw. Eng. (ICSE)*. New York, NY, USA: ACM, 2000, pp. 357–370.

[23] N. Fenton and J. Bieman, *Software Metrics: A Rigorous and Practical Approach*. Boca Raton, FL, USA: CRC Press, 2014.

[24] P. Ram, P. Rodriguez, M. Oivo, and S. Martinez-Fernandez, "Success factors for effective process metrics operationalization in agile software development: A multiple case study," in *Proc. IEEE/ACM Int. Conf. Softw. Syst. Processes (ICSSP)*, May 2019, pp. 14–23.

[25] Z. Bukhari, J. Yahaya, and A. Deraman, "Software metric selection methods: A review," in *Proc. Int. Conf. Electr. Eng. Informat. (ICEEI)*, Aug. 2015, pp. 433–438.

[26] I. Linkov, D. Loney, S. Cormier, F. K. Satterstrom, and T. Bridges, "Weight-of-evidence evaluation in environmental assessment: Review of qualitative and quantitative approaches," *Sci. Total Environ.*, vol. 407, no. 19, pp. 5199–5205, Sep. 2009.

[27] M. Convertino, K. Baker, C. Lu, J. T. Vogel, K. McKay, and I. Linkov, "Metric selection for ecosystem restoration," DTIC, Fort Belvoir, VI, USA, Tech. Rep. ERDC TN-EMRRP-EBA-19, 2013.

[28] C. Calero, J. Ruiz, and M. Piattini, "Classifying Web metrics using the Web quality model," *Online Inf. Rev.*, vol. 29, no. 3, pp. 227–248, Jun. 2005.

[29] A. Stefani and M. Xenos, "Meta-metric evaluation of E-Commerce-related metrics," *Electron. Notes Theor. Comput. Sci.*, vol. 233, pp. 59–72, Mar. 2009.

[30] P. Longley, *Geographic Information Systems and Science*. Hoboken, NJ, USA: Wiley, 2005.

[31] V. R. Basili and D. M. Weiss, "A methodology for collecting valid software engineering data," *IEEE Trans. Softw. Eng.*, vol. SE-10, no. 6, pp. 728–738, Nov. 1984.

[32] S. Wagner, A. Goeb, L. Heinemann, M. Kläs, C. Lampasona, K. Lochmann, A. Mayr, R. Plösch, A. Seidl, J. Streit, and A. Trendowicz, "Operationalised product quality models and assessment: The quamoco approach," *Inf. Softw. Technol.*, vol. 62, pp. 101–123, Jun. 2015.

[33] S. Martinez-Fernandez, A. M. Vollmer, A. Jedlitschka, X. Franch, L. Lopez, P. Ram, P. Rodriguez, S. Aaramaa, A. Bagnato, M. Choras, and J. Partanen, "Continuously assessing and improving software quality with software analytics tools: A case study," *IEEE Access*, vol. 7, pp. 68219–68239, 2019.

[34] A. Meneely, B. Smith, and L. Williams, "Validating software metrics: A spectrum of philosophies," *ACM Trans. Softw. Eng. Methodol.*, vol. 21, no. 4, p. 24, 2012.

[35] K. Srinivasan and T. Devi, "Software metrics validation methodologies in software engineering," *Int. J. Softw. Eng. Appl.*, vol. 5, no. 6, p. 87, 2014.

[36] V. Antinyan, M. Staron, A. Sandberg, and J. Hansson, "Validating software measures using action research a method and industrial experiences," in *Proc. 20th Int. Conf. Eval. Assessment Softw. Eng. (EASE)*, 2016, p. 23.

[37] M. Manzano, E. Mendes, C. Gómez, C. Ayala, and X. Franch, "Using Bayesian networks to estimate strategic indicators in the context of rapid software development," in *Proc. 14th Int. Conf. Predictive Models Data Analytics Softw. Eng.*, 2018, pp. 52–55.

[38] R. Shatnawi, "Identifying threshold values of change-prone modules," in *Proc. Int. Conf. E-Bus. Mobile Commerce (ICEMC)*, 2018, pp. 39–43.

[39] Aarti, G. Sikka, and R. Dhir, "Threshold-based empirical validation of object-oriented metrics on different severity levels," *Int. J. Intell. Eng. Inform.*, vol. 7, nos. 2–3, pp. 231–262, 2019. [Online]. Available: https://www.inderscienceonline.com/doi/abs/10.1504/IJIEI.2019.099090

[40] V. R. Basili, "Software modeling and measurement: The goal/question/metric paradigm," College Park, MD, USA, Tech. Rep. CS-TR-2956, UMIACS-TR-92-96, 1992.

[41] M. Perkusich, A. Medeiros, L. C. e Silva, K. C. Gorgńio, H. O. de Almeida, and A. Perkusich, "A Bayesian networkapproach to assist on the interpretation of software metrics," in *Proc. 30th Annu. ACM Symp. Appl. Comput. (SAC)*, Salamanca, Spain, Apr. 2015, pp. 1498–1503, doi: 10.1145/2695664.2695749.

[42] N. Fenton and M. Neil, *Risk Assessment and Decision Analysis With Bayesian Networks*, 2nd ed. Boca Raton, FL, USA: CRC Press, Sep. 2018.

[43] J. Pearl and S. Russell, "Bayesian networks," in *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA, USA: MIT Press, 1995.

[44] L. Uusitalo, "Advantages and challenges of Bayesian networks in environmental modelling," *Ecol. Model.*, vol. 203, nos. 3–4, pp. 312–318, May 2007.

[45] E. Lee, Y. Park, and J. G. Shin, "Large engineering project risk management using a Bayesian belief network," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 5880–5887, Apr. 2009.

[46] L. Zhang, X. Wu, L. Ding, M. J. Skibniewski, and Y. Yan, "Decision support analysis for safety control in complex project environments based on Bayesian networks," *Expert Syst. Appl.*, vol. 40, no. 11, pp. 4273–4282, Sep. 2013.

[47] M. A. de Oliveira, O. Possamai, L. V. O. Dalla Valentina, and C. A. Flesch, "Applying Bayesian networks to performance forecast of innovation projects: A case study of transformational leadership influence in organizations oriented by projects," *Expert Syst. Appl.*, vol. 39, no. 5, pp. 5061–5070, Apr. 2012.

[48] M. Neil, N. Fenton, and L. Nielson, "Building large-scale Bayesian networks," *Knowl. Eng. Rev.*, vol. 15, no. 3, pp. 257–284, Sep. 2000.

[49] M. Perkusich, A. Perkusich, and H. O. D. Almeida, "Using survey and weighted functions to generate node probability tables for Bayesian networks," in *Proc. BRICS Congr. Comput. Intell. 11th Brazilian Congr. Comput. Intell.*, Sep. 2013, pp. 183–188.

[50] D. Heckerman, "Learning in graphical models," in *A Tutorial on Learning With Bayesian Networks*, M. I. Jordan, Ed. Cambridge, MA, USA: MIT Press, 1999, pp. 301–354.
[51] N. E. Fenton, M. Neil, and J. G. Caballero, "Using ranked nodes to model qualitative judgments in Bayesian networks," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 10, pp. 1420–1432, Oct. 2007.
[52] K. Huang and M. Henrion, "Efficient search-based inference for noisy-OR belief networks: TopEpsilon," in *Proc. 12th Int. Conf. Uncertainty Artif. Intell. (UAI)*. San Francisco, CA, USA: Morgan Kaufmann, 1996, pp. 325–331.
[53] F. J. Diez, "Parameter adjustment in Bayes networks. the generalized noisy or-gate," in *Proc. 9th Conf. Uncertainty Artif. Intell.* San Mateo, CA, USA: Morgan Kaufmann, 1993, pp. 99–105.
[54] B. Das, "Generating conditional probabilities for Bayesian networks: Easing the knowledge acquisition problem," *CoRR*, vol. cs.AI/0411034, pp. 1–24, Nov. 2004.
[55] G. Çalıklı and A. B. Bener, "Influence of confirmation biases of developers on software quality: An empirical study," *Softw. Qual. J.*, vol. 21, no. 2, pp. 377–416, Jun. 2013.
[56] E. Mendes, *Practitioner's Knowledge Representation: A Pathway to Improve Software Effort Estimation*. Berlin, Germany: Springer-Verlag, 2014.
[57] M. Perkusich, G. Soares, H. Almeida, and A. Perkusich, "A procedure to detect problems of processes in software development projects using Bayesian networks," *Expert Syst. Appl.*, vol. 42, no. 1, pp. 437–450, Jan. 2015.
[58] A. C. Constantinou, N. Fenton, W. Marsh, and L. Radlinski, "From complex questionnaire and interviewing data to intelligent Bayesian network models for medical decision support," *Artif. Intell. Med.*, vol. 67, pp. 75–93, Feb. 2016.
[59] E. A. Drost, "Validity and reliability in social science research," *Educ., Res. Perspect.*, vol. 38, no. 1, p. 105, 2011.
[60] B. Yet, Z. Perkins, N. Fenton, N. Tai, and W. Marsh, "Not just data: A method for improving prediction with knowledge," *J. Biomed. Informat.*, vol. 48, pp. 28–37, Apr. 2014.
[61] R. Marinescu, "Detection strategies: Metrics-based rules for detecting design flaws," in *Proc. 20th IEEE Int. Conf. Softw. Maintenance*, 2004, pp. 350–359.
[62] H. Zhang, F. Shu, Y. Yang, X. Wang, and Q. Wang, "A fuzzy-based method for evaluating the trustworthiness of software processes," in *Proc. Int. Conf. Softw. Process.* Springer, 2010, pp. 297–308.
[63] F. Khomh, S. Vaucher, Y.-G. Guéhéneuc, and H. Sahraoui, "BDTEX: A GQM-based Bayesian approach for the detection of antipatterns," *J. Syst. Softw.*, vol. 84, no. 4, pp. 559–572, Apr. 2011.
[64] R. Saraiva, M. Perkusich, H. Almeida, and A. Perkusich, "A systematic process to define expert-driven software metrics thresholds (S)," in *Proc. 31st Int. Conf. Softw. Eng. Knowl. Eng.*, Lisbon, Portugal, Jul. 2019, pp. 171–226, doi: 10.18293/SEKE2019-217.
[65] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Softw. Eng.*, vol. 14, no. 2, pp. 131–164, Apr. 2009.
[66] K. Petersen and C. Wohlin, "Context in industrial software engineering research," in *Proc. 3rd Int. Symp. Empirical Softw. Eng. Meas.*, Oct. 2009, pp. 401–404.
[67] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," *MIS Quart.*, vol. 13, no. 3, pp. 319–340, Sep. 1989.
[68] A. L. Beberg, D. L. Ensign, G. Jayachandran, S. Khaliq, and V. S. Pande, "Foldinghome: Lessons from eight years of volunteer distributed computing," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, May 2009, pp. 1–8.
[69] C. K. Riemenschneider, B. C. Hardgrave, and F. D. Davis, "Explaining software developer acceptance of methodologies: A comparison of five theoretical models," *IEEE Trans. Softw. Eng.*, vol. 28, no. 12, pp. 1135–1145, Dec. 2002.
[70] K. Verbert, R. Babuška, and B. De Schutter, "Bayesian and Dempster–Shafer reasoning for knowledge-based fault diagnosis–A comparative study," *Eng. Appl. Artif. Intell.*, vol. 60, pp. 136–150, Apr. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197617300118
[71] Y. Robert, *Case Study Research: Design and Methods*, vol. 761, no. 92553. Newbury Park, CA, USA: Sage, 2003, p. 8.
[72] A. T. Misirli and A. B. Bener, "Bayesian networks for evidence-based decision-making in software engineering," *IEEE Trans. Softw. Eng.*, vol. 40, no. 6, pp. 533–554, Jun. 2014.
[73] I. M. del Águila and J. del Sagrado, "Bayesian networks for enhancement of requirements engineering: A literature review," *Requirements Eng.*, vol. 21, no. 4, pp. 461–480, Nov. 2016.
[74] A. Tosun, A. B. Bener, and S. Akbarinasaji, "A systematic literature review on the applications of Bayesian networks to predict software quality," *Softw. Qual. J.*, vol. 25, no. 1, pp. 273–305, Mar. 2017.
[75] T. Hall and N. Fenton, "Implementing effective software metrics programs," *IEEE Softw.*, vol. 14, no. 2, pp. 55–65, Mar./Apr. 1997.

RENATA SARAIVA received the Ph.D. degree in computer science. She is currently a member of the Intelligent Software Engineering Research Group. Her current research interests include software measurement programs and intelligent techniques applied to software engineering.

AMAURY MEDEIROS received the M.Sc. degree in computer science. His current research interests include software measurement programs and intelligent techniques applied to software engineering.

MIRKO PERKUSICH received the Ph.D. degree in computer science. He is currently a Research Manager with VIRTUS, leading the Intelligent Software Engineering Research Group. His current research interests include applying intelligent techniques, including recommender systems, to solve complex software engineering problems.

DALTON VALADARES received the bachelor's degree in computer science, the master's degree, the M.B.A. degree in project management, and a technical course in informatics, and the Ph.D. degree in computer science from the Federal University of Campina Grande (UFCG). Since 2016, he has been studying security/privacy in cloud/fog-based Internet of Things applications. He is currently a Professor with the Federal Institute of Pernambuco (IFPE). He has over 15 years of experience in IT, having worked on several research and development projects assuming different roles: systems analyst, embedded systems/web developer, quality/testing analyst, and project manager. He also develops research collaboration at the ISE Group, Embedded Laboratory (UFCG), and at the GPRSC, UTFPR.

KYLLER COSTA GORGÔNIO graduated in computer science from the Universidade Federal da Paraíba, in 1999. He received the master's degree in computer science from the Universidade Federal da Paraíba, in 2001, and the Ph.D. degree in software from the Universitat Politècnica de Catalunya, in 2010. He is currently a Professor with the Universidade Federal de Campina Grande. He has experience in computer science focusing on software engineering. His research interests include Petri nets, protocols, asynchronous communication mechanisms, coloured Petri nets, and model checking.

ANGELO PERKUSICH (Member, IEEE) received the master's and Ph.D. degrees in electrical engineering from the Federal University of Paraíba, in 1987 and 1994, respectively. He was a Visiting Researcher with the Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, USA, from 1992 to 1993, and developed research activities on software engineering and formal methods. He has been a Professor with the Department of Electrical Engineering (DEE), Federal University of Campina Grande (UFCG), since 2002. He is also the principal investigator of research projects financed by public institutions, such as FINEP (Brazilian Agency for Research and Studies) and CNPq (Brazilian National Research Council), as well as private companies. He is also the Founder and the Director of VIRTUS Innovation Center and the Embedded and Pervasive Computing Laboratory. The focus on research projects are on formal methods, embedded systems, mobile pervasive and ubiquitous computing, and software engineering. He has over 30 years of teaching experience in the university as well as training courses for industry in the context of software for real-time systems, software engineering, embedded systems, computer networks, and formal methods. His research interests include embedded systems, software engineering, mobile pervasive computing, and formal methods, with more than 300 articles published, and 80 master thesis and 21 doctoral dissertations advised.

HYGGO ALMEIDA is currently the Head of the Intelligent Software Engineering Research Group. He is also a Professor with the Federal University of Campina Grande. His current research interests include applying intelligent techniques, including recommender systems, to solve complex software engineering problems.