Data-driven Risk Management for Requirements Engineering: An Automated Approach based on Bayesian Networks

Florian Wiesweg
Technische Universität Berlin
Berlin, Germany
florian.wiesweg@tu-berlin.de

Andreas Vogelsang
Technische Universität Berlin
Berlin, Germany
andreas.vogelsang@tu-berlin.de

Daniel Mendez
Blekinge Institute of Technology
Blekinge, Sweden
daniel.mendez@bth.se

***Abstract*—Requirements Engineering (RE) is a means to reduce the risk of delivering a product that does not fulfill the stakeholders' needs. Therefore, a major challenge in RE is to decide how much RE is needed and what RE methods to apply. The quality of such decisions is strongly based on the RE expert's experience and expertise in carefully analyzing the context and current state of a project. Recent work, however, shows that lack of experience and qualification are common causes for problems in RE. We trained a series of Bayesian Networks on data from the NaPiRE survey to model relationships between RE problems, their causes, and effects in projects with different contextual characteristics. These models were used to conduct (1) a post-mortem (diagnostic) analysis, deriving probable causes of sub-optimal RE performance, and (2) to conduct a preventive analysis, predicting probable issues a young project might encounter. The method was subject to a rigorous cross-validation procedure for both use cases before assessing its applicability to real-world scenarios with a case study.**

***Index Terms*—Risk management, machine learning, data-driven RE**

I. Introduction

The purpose of Requirements Engineering (RE) is to elicit, document, analyze, and manage requirements to minimize the risk of delivering a system that does not meet the stakeholders' desires and needs [1]. Over the last 30 years, a number of methods, processes, tools, and best practices have been proposed to support this goal. However, there is no silver-bullet method or process that fits every project. In fact, a large part of the job of a requirements engineer in practice is to observe and analyze the context and current state of a project carefully and decide how much and what kind of RE is beneficial. As already addressed in the above-mentioned definition of RE, this decision is often a matter of controlling risks. Conducting RE tasks always comes with costs that ideally pay off in the sense that they lower a particular risk for a project. Making such decisions demands social and technical skills but also a lot of experience. Recent studies have shown that lack of experience and lack of qualification of RE team members are the second and third most common causes for problems in RE (lack of time being the top cause) [2]. As a result, a number of projects fail either because of too little RE leading to stakeholder dissatisfaction or too much RE leading to high costs and developer frustration.

In this paper, we propose a data-driven approach to risk management in RE. Our goal is to predict RE problems, their causes, and effects for a given project. For this purpose, we evaluated different versions of Bayesian Networks that model the relations between causes, problems, and effects in RE. We trained the models on data that was collected through two surveys with answers from 228 and 488 practitioners, respectively, about problems, causes, and effects encountered in real projects. These surveys also provide data on the context of the projects.

We use the trained models for the following two use cases:

- **Post-Mortem Analysis:** Given a set of problems and effects observed in a failing or failed project, the approach diagnoses the most likely causes leading to these issues (known as *diagnostic reasoning* in literature [3]).
- **Preventive Analysis:** Given a set of causes and effects observed in a new or running project, the approach predicts the most likely problems to be faced (known as *predictive reasoning* in literature [3]).

We implemented the approach as an easily consumable web service, on which we based a graphical user interface to enter evidence and analyze the resulting predictions.

We performed two types of evaluations for our approach. Firstly, we performed cross-validation to compare the predictive power of different models. We achieved the best results for both use cases with surprisingly simple models, which ignore the causal structure implied by the original survey, but include a set of context factors. For varying probability thresholds $t \in \{0.3, 0.5, 0.7\}$, the best diagnostic model achieves recalls of 0.6, 0.48, 0.44 and precisions of 0.76, 0.92, 0.99, respectively. The best predictive model achieves recalls of 0.84, 0.69, 0.59 and precisions of 0.71, 0.89, 0.99. A ranking-based output of the top-5 predictions results in a recall of 0.81 and a precision of 0.38 for the best diagnostic model and a recall of 0.73 and precision of 0.71 for the best predictive model.

Secondly, we conducted a case study in industry to evaluate the external validity of the approach. We compared and discussed the predictions of the tool with the expectations of an

RE expert for the diagnostic reasoning use case. Furthermore, we elicited feedback regarding the importance of recall vs. precision for the problem and how the tool should be tailored in detail to support practitioners best. In a nutshell, the case study showed that the method achieves good congruence between its predictions and the results expected by the expert, but requires additional tuning towards high precision.

We conclude that such data-driven approaches are very likely to be practical and advantageous, but that the remaining potentials in the underlying data and the user interface should be realized first.

II. Related Work

A. NaPiRE Initiative

The survey data used for our analysis originates from the NaPiRE project, which was presented at several occasions [2], [4], [5]. Formerly a German initiative, it has incorporated a variety of teams of other nationalities since its inception and is now supported by RE researchers from all over the world. Most analyses run on the data have so far been of a descriptive nature, e.g., comparing summary statistics from different countries [6], or trying to find the most prevalent problems, their causes, and their effects in RE projects [2]. Two studies, however, applied Bayesian Networks to analyze the relationship between these items, with the purpose of either supporting so-called *Defect Causal Analysis* [7] (the diagnostic reasoning use case) or allowing data-driven risk-management (the predictive reasoning use case) [8]. Both studies relied heavily on the commercial Netica tool and lack a sophisticated validation procedure including an evaluation of their predictive power.

B. Bayesian networks in SE and RE

While certainly not a common tool for software engineers, Bayesian Networks have seen a variety of applications in Software and Requirements Engineering according to a mapping study [9], ranging (in declining importance) from software fault detection over software project management to design and testing. We would position this work in the second category, software project management. The survey also examines the methodological approach taken in the field: 80% of the 117 works rely solely on categorical variables, just as we do, while empirical data is used in only about 45% of the cases to learn the parameters of the model. In 24% of the cases, the network layout is inferred from the data as well, which we do heuristically. According to a follow-up survey by the same authors [10], there has been a trend towards data-driven methods and continuous variables. These claims should, however, be seen in the light of the very low sample size of only 10 papers.

A survey of 20 studies matching our application in RE a bit more closely can be found in [3], confirming similar trends: the network layout is usually constructed manually, while a more significant, but not overwhelming part of the authors use quantitative, data-driven methods for parameter learning. Two of these studies treat the more specific topic of the RE process: Tang et al. [11] use a Bayesian Network based on survey data to propose a set of requirements engineering techniques for different phases of the RE process. Nagy et al. [12] employ a network with manually specified layout and parameters for development release planning and project health monitoring. None of the two approaches was systematically validated and different model designs were not evaluated.

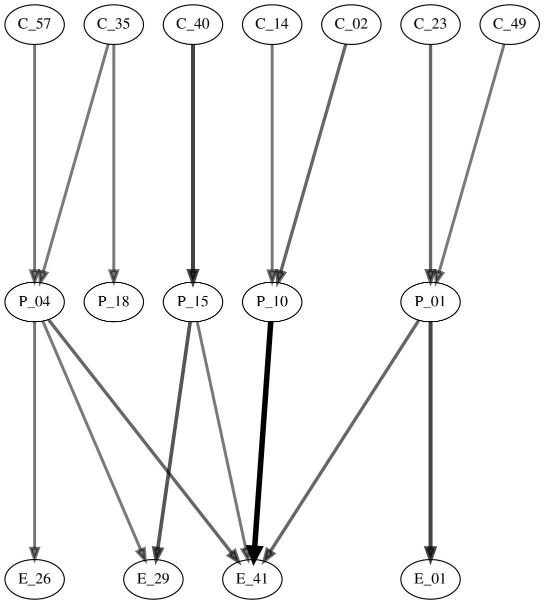


Fig. 1. Visualisation of a Bayesian Network. Line thickness indicates how often a given the edge was named by participants.

C. Use Cases

The diagnostic reasoning use case is inspired by concepts such as *Root Cause Analysis* or *Defect Causal Analysis* with original works stemming from the early 90s [13], [14], focusing mainly on quality management techniques. An interesting list of data-driven approaches is presented by Solé et al. [15] (unfortunately only as a pre-print), including a variety of applications of Bayesian Networks to the topic.

The preventive analysis use case aids risk management by estimating likely problems. These risks can then be tackled by choosing matching RE methods, so it is possible to implement Just-in time-RE [16], Just enough RE [17], or Value-oriented RE [18] in an empirically founded way. The model-based risk management tools available so far [19], [20] neither apply specifically to RE nor do they incorporate larger scale data sets.

III. Bayesian Networks

Bayesian Networks (also called belief networks) are part of a class of stochastic models called *graphical models*, also including hidden Markov models and conditional random fields, which are popular in machine vision applications. They encode the joint stochastic distribution of a set of discrete and dependent random variables $X = \{X_1, \ldots, X_n\}$ in a directed acyclic graph such as the one depicted in Figure 1. Each node in the graph represents a random variable X_i whose distribution is dependent on its predecessors $pa(X_i)$, and this conditional distribution is stored along the node. The explicit specification of dependencies strongly reduces the size of the parameter vector $\boldsymbol{\theta}$: a toy network of $n = 4$ binomial variables X_1, X_2, X_3, X_4 with edges pointing from X_i to X_{i+1} would require a vector of $2^n = 16$ scalars in a naive joint distribution specification, while a Bayesian Network formulation only requires $2 \cdot n = 8$ scalars.

It follows from the above definition that a variable X_i in such a network is (conditionally) independent from any other variable in the graph given its predecessors $pa(X_i)$, so the joint probability function of the model is

$$P(X_1, \ldots, X_n) = \prod_{i}^{n} P(X_i \mid pa(X_i), \boldsymbol{\theta})$$

Given m instantiations of all variables $x_1^j, \ldots, x_n^j, 1 \leq j \leq m$, it is thus possible to calculate the likelihood of such a model:

$$L(\boldsymbol{\theta} \mid x^1, \ldots, x^m) = \prod_{j=1}^{m} P(X_1 = x_1^j, \ldots, X_n = x_n^j \mid \boldsymbol{\theta})$$

The optimization of this function with (usually infeasible) analytical or (usually gradient-based) numerical methods yields the maximum likelihood point estimate of $\boldsymbol{\theta}$[1].

If the actual instantiations of any other random variables are known, this evidence $E,\ E \cap V_o = \emptyset$ can be introduced into the network by manually modifying the conditional distributions at the respective nodes to generate a prediction tailored to the situation. $v(X)$ being the set of all possible assignments to a set of random variables, the conditional probability is calculated as follows:

$$\begin{aligned} P(V_o = v_o \mid E = e) &= \frac{P(V_o = v_o \wedge E = e)}{P(E = e)} \\ &= \frac{\sum_{v(X_i/E/V_o)} P(E = e, V_o = v_o)}{\sum_{v(X_i/E)} P(E = e, X_i = x_i)} \end{aligned}$$

This marginalization operation is more expensive than it would be for a naive joint distribution formulation, but the reduced memory requirements and statistical advantages of the smaller parameter space generally outweigh this concern. In addition, a variety of optimized approximate algorithms (such as Belief Propagation, or Gibbs Sampling, which we used) exploit the specific structure of the inference problem to reduce the overall computation time considerably.

For a more in-depth introduction to the theoretical foundations of Bayesian Networks we suggest the textbook by Koski and Noble [21].

TABLE I
DESCRIPTIVE STATISTICS OF THE NAPIRE DATA SETS

Parameter	2014	2018
Participants	228	488
...from Africa	0	3
...from Asia	0	24
...from Europe	126	208
...from North America	28	40
...from South America	74	185
...from elsewhere	0	28
Group size	company	team
... $x \leq 50$	69	443
... $51 \leq x \leq 250$	33	39
... $251 \leq x$	114	4
...unknown	2	2
Development method		
...agile	92	194
...hybrid	58	161
...plan-driven	46	124
...unknown	22	0

TABLE II
AVAILABLE PROBLEMS, CAUSES, AND EFFECTS PER DATA SET

	source	V	2014	2018
problems	predefined	`P`	21	20
causes	coded	`C`	92	120
cause categories[1]	predefined	`CC`	5	n/a
effects	coded	`E`	49	55
effect categories[2]	predefined	`EC`	5	n/a

[1] Input, Method, Organization, People, Tools
[2] Implementation, Organization, Product, Customer, Validation

IV. APPROACH

A. Preprocessing

The 2014 NaPiRE data was obtained from FigShare, as proposed by the project [22]. On request, the NaPiRE team provided the most recent data from the 2018 edition of the survey. A short summary of both data sets in terms of descriptive statistics is available in Table I; for more details, please refer to Méndez Fernández et al. [2]. Both consist of a set of context variables for each subject, which were generated from closed questions, and five answers to the problems-causes-effects question: the participants were asked to think of a recent project and to select five problems experienced in the project from a closed list. Afterwards, they were asked to assign a rank $r \in \{1 \ldots 5\}$, a cause, and an effect to each of these problems. The latter two were coded manually, mostly in accordance with the principles established by Grounded Theory [2]. The 2014 data set provided coarse categories into which causes and effects were grouped by the authors. Merging both data sets would have been a natural step to increase the statistical foundation of our approach, but was unfortunately prevented by incompatibilities due to survey improvements and the distinct manual coding processes.

We assigned each of the available variables to a set V (the *variable type*, as depicted in Tables II and III) and then

[1] The stochastic formulation of the problem enables the inference of confidence intervals for all parameters and the predictions, which might be an interesting extension of this work if practitioners show interest.

TABLE III
AVAILABLE CONTEXT FACTORS PER DATA SET

2014	V	type	indicators
company size	CS	categorical	8
development method[1]	CDM	categorical	5
distributed projects	CD	binary	1
2018		type	
team size	CS	continuous	6
development method[2]	CDM	ordinal	5
distributed project	CD	binary	1
quality of customer relation	CR	ordinal	5
system type[3]	CT	categorical	3

[1] Waterfall, V-Model XT, Scrum, XP, RUP
[2] Agile, rather agile, hybrid, rather plan-driven, plan-driven
[3] Embedded system, business information system, hybrid

transformed it into one or more binary variables $v_i \in V$ according to the following variable type specific rules.

- Problem, cause and effect questions: Add one binary variable per possible answer. The variable is true if the subject selected this answer in the survey, false otherwise.
- Cause and effects categories: Add one binary variable per category. True if the cause selected by the subject belongs to this category, false otherwise.
- For each context factor, we added a variable type V_c, transforming it according to the data type mentioned in Table III:
  - Binary: Add one binary indicator variable.
  - Categorical: Add one binary indicator variable per value.
  - Ordinal: Add one binary indicator variable per value.
  - Continuous: Discretize into a set of equiprobable intervals and add one binary indicator variable per interval.

This resulted in eight variable types for both, the 2014 and the 2018 data set with a total of 196 (2014) or 216 binary variables (2018). Given that only 28 (2014)[2] or 20 (2018)[3] of the v_i can actually be true for each participant, the input data matrix is relatively sparse.

B. Network construction

In this work, the graph representing a Bayesian Network is defined by its architecture $\mathcal{A}$, which is a set of tuples of variable types. Each $(V_i, V_j) \in \mathcal{A}$ indicates that (1) all binary variables $v \in V_i \cup V_j$ are contained in the graph as a node and (2) that an edge is added from each $v_i \in V_i$ to each $v_j \in V_j$. For example, an architecture $\mathcal{A} = \{(\texttt{C}, \texttt{P})\}$ specifies that all cause nodes are connected to all problem nodes, edges pointing to the problem nodes. This would result in $92 \cdot 21 = 1932$ (2014) or $120 \cdot 20 = 2400$ (2018) edges. More nodes and edges to other variable types can be specified by appending more tuples to $\mathcal{A}$ as long as the restrictions the inference algorithm places on the graph are respected.

These considerations show that even for simple architectures, the resulting graph quickly suffers from the curse of dimensionality, which is aggravated by the manually coded cause/effect statements in the survey leading to a high number of nodes. Learning of and inference on such models would require prohibitively large amounts of memory and CPU time, so we introduced two simple filter mechanisms to reduce complexity:

- **Minimum Variable Occurrence Filter:** The number of true values for each variable in the data is counted. If this number is less than $f(V)$ (i.e. less than f subjects reported this fact), its node is excluded from the graph. With this filter, variables with very little support in the dataset can be excluded.
- **Minimum Relation Occurrence Filter:** The number of times the two variables connected by an edge are both true is counted. If this number is less than $g(V_1, V_2)$ (i.e. the dependence was reported by less than g subjects), the edge is excluded from the graph. With this filter, relations with very little support in the dataset can be excluded.

More fine-grained control was achieved by not summing the number of occurrences of nodes or edges, but of the inverse rank $r_{\text{inv}} = 5 - r$ of a given cause-problem-effect triple. These heuristics worked well in our case, although it might be an option to explore more sophisticated approaches like the K2 structure learning algorithm [21].

In the above notation, a use case case is equivalent to a set of binary variables V_o which constitutes the output of a model. The choice of $\mathcal{A}$ is independent of the use case, as long as all variables in V_o are included in $\mathcal{A}$. Any model can thus be applied to any use case; early experimentation has shown, however, that $\mathcal{A}$ has a major influence on the quality of the predictions. We formally define our use cases as follows:

Diagnostic reasoning

$$V_D = \{v \in \texttt{C} \mid v \text{ is contained in } \mathcal{A}\}$$

Predictive reasoning

$$V_P = \{v \in \texttt{P} \mid v \text{ is contained in } \mathcal{A}\}$$

In other words, diagnostic reasoning is the prediction of causes (C) and predictive reasoning is the prediction of problems (P).

An example, which will later be called the *Survey architecture*, is inspired by the causality assumptions of the NaPiRE survey:

$$\mathcal{A} = \{(\texttt{C}, \texttt{P}), (\texttt{P}, \texttt{E})\}$$

It is depicted graphically in Figure 1 with unrealistically high filter values to allow for a readable representation including individual binary variables. The following visual representations will be limited to variable types only.

[2] 2014: 5 cause, 5 problem, 5 effect, 3 context factor, 10 category nodes.
[3] 2018: 5 cause, 5 problem, 5 effect, 5 context factor, 0 category nodes.

C. Implementation and Reproducibility

In order to circumvent the performance problems verbally reported by other authors [7], we implemented this method with Julia [23], relying on the (formidable) BayesNets.jl library [24] for inference and on graphviz for visualization [25]. The algorithms are wrapped in a simple web service to allow for easy integration into RE support tools and surveys. We are committed to ensuring the reproducibility of the following results, so our code and the data are freely accessible. We cordially invite other researchers to verify and extend this work[4] containing our validation results.

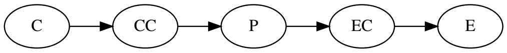
V. Evaluation

We validated this method with a two-step approach. Firstly, to evaluate whether we produce predictions which are consistent with the data, it was subject to 10-fold Monte Carlo cross validation (leaving out 30 samples in each iteration, Section V-A). Secondly, we selected the architecture yielding the best performance and conducted a case study to test its applicability to real-world scenarios (Section V-B).

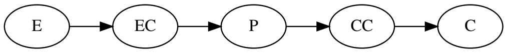
A. Internal validity

1) Architectures: We defined the following eight architectures to be cross-validated. Four of them are inspired by the available literature.

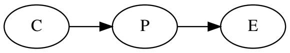
$\mathcal{A}_1$**:** Kalinowski architecture [7]



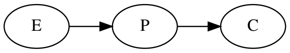
$\mathcal{A}_2$**:** Inverse Kalinowski architecture [7]



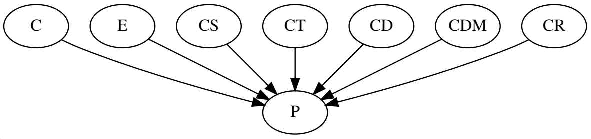
$\mathcal{A}_3$**:** Survey architecture [4]



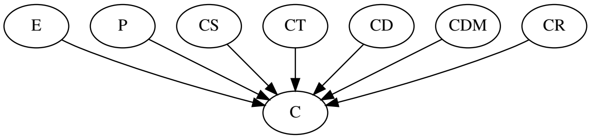
$\mathcal{A}_4$**:** Inverse survey architecture [4]



$\mathcal{A}_5$**:** Simple architecture (without context factors)
$\mathcal{A}_6$**:** Simple architecture (with context factors)



$\mathcal{A}_7$**:** Inverse simple architecture (without context factors)
$\mathcal{A}_8$**:** Inverse simple architecture (with context factors)



To estimate whether our method achieves better performance than simple guessing does, we also defined a baseline algorithm $\mathcal{A}_0$, predicting for each $v_o \in V_o$ its relative frequency in the input data set.

$\mathcal{A}_1$ and $\mathcal{A}_2$ were evaluated on the 2014 data set, since it is the only one providing the necessary cause/effect categories. The remaining architectures were evaluated on the 2018 data set due to the higher number of participants and the better availability of context factors. Filter parameters were determined empirically by repeatedly running the validation on standard hardware, incrementing the filter values until results were obtained within 48 hours and without provoking out-of-memory errors. This results in a maximum duration of $48\text{h}/300 \approx 10$ minutes per inference, which we deemed the maximum acceptable inference duration for the case study.

2) Metrics: Each tuple of architecture $\mathcal{A}$ and use case V_o was evaluated with the following metrics. Let s be the number of samples in the validation set, $i \in \{1, \ldots, s\}$ the sample index, and $t = \{0.1, 0.2, \ldots, 0.9\}$ a set of probability thresholds. For convenience, $E_i(v_o)$ is the variable's actual value for the sample i, $T_i(v_o, t) = (P(V_o = v_o | E = e_i) > t)$ is a boolean indicator function based on the evidence e_i given by sample i, and boolean values are equal to 1 or 0 in summation if they are true or false, respectively. $V_o^k = \{v_1, \ldots, v_k\}$ is the set of the k output variables with the highest predicted probabilities.

- **binary accuracy**: the number of correct predictions to the number of all predictions.

$$acc(t) = \frac{1}{s \cdot |V_o|} \sum_{i=1}^{s} \sum_{v_o \in V_o} \Big(T_i(v_o, t) = E_i(v_o) \Big)$$

- **precision**: the number of correct predictions of true to the number of all predictions of true.

$$pre(t) = \frac{1}{s} \sum_{i=1}^{s} \left(\sum_{v_o \in V_o | T_i(v_o,t)} E_i(v_o) \right) \left(\sum_{v_o \in V_o} T_i(v_o, t) \right)^{-1}$$

- **recall**: the number of correct predictions of true to the number of all actually true variables.

$$rec(t) = \frac{1}{s} \sum_{i=1}^{s} \left(\sum_{v_o \in V_o | E_i(v_o)} T_i(v_o, t) \right) \left(\sum_{v_o \in V_o} E_i(v_o) \right)^{-1}$$

- **ranking precision**: the number of actually true variables in the ranking to the ranking length.

$$rre(k) = \frac{1}{s} \sum_{i=1}^{s} \left(\sum_{v_o \in V_o^k} E_i(v_o) \right) k^{-1}$$

- **ranking recall**: the number of actually true variables in the ranking to the number of all actually true variables.

$$rpr(k) = \frac{1}{s} \sum_{i=1}^{s} \left(\sum_{v_o \in V_o^k} E_i(v_o) \right) \left(\sum_{v_o \in V_o} E_i(v_o) \right)^{-1}$$

As mentioned before, the dataset is relatively sparse, meaning that our baseline algorithm will correctly predict the absence of most output variables due to their low overall probability, resulting in a high binary accuracy for this trivial method

[4] Temporary anonymized package: https://figshare.com/s/5f261b5d82aea52151e6

TABLE IV

RESULTS OF THE INTERNAL VALIDATION

use case	architecture		dataset	$\|V_o\|^1$	$\overline{acc}$	$\overline{rec}$	$\overline{pre}$	$rre(5)$	$rpr(5)$
V_D	$\mathcal{A}_0$	Baseline	2018	29	0.89	0.07	0.04	0.32	0.16
V_D	$\mathcal{A}_1$	Kalinowski	**2014**	28	0.83	0.10	0.02	0.30	0.10
V_D	$\mathcal{A}_2$	Inverse Kalinowski	**2014**	28	0.92	0.12	0.37	0.48	0.15
V_D	$\mathcal{A}_3$	Survey	2018	25	0.89	0.13	0.23	0.47	0.21
V_D	$\mathcal{A}_4$	Inverse Survey	2018	25	0.89	0.14	0.32	0.53	0.21
V_D	$\mathcal{A}_5$	Simple	2018	26	0.89	0.18	0.35	0.57	0.26
V_D	$\mathcal{A}_6$	Simple with context	2018	15	0.91	0.25	0.81	**0.88**	0.26
V_D	$\mathcal{A}_7$	Inverse Simple	2018	30	0.90	0.33	0.73	0.66	0.30
V_D	$\mathcal{A}_8$	Inverse Simple with context	2018	24	**0.93**	**0.54**	**0.83**	0.81	**0.38**
V_P	$\mathcal{A}_0$	Baseline	2018	20	0.71	0.26	0.20	0.44	0.39
V_P	$\mathcal{A}_1$	Kalinowski	**2014**	20	0.74	0.28	0.59	0.47	0.39
V_P	$\mathcal{A}_2$	Inverse Kalinowski	**2014**	20	0.73	0.30	0.62	0.50	0.43
V_P	$\mathcal{A}_3$	Survey	2018	20	0.8	0.31	0.71	0.59	0.51
V_P	$\mathcal{A}_4$	Inverse Survey	2018	20	0.81	0.49	0.75	0.63	0.55
V_P	$\mathcal{A}_5$	Simple	2018	19	0.84	0.57	0.82	0.70	0.60
V_P	$\mathcal{A}_6$	Simple with context[2]	2018	20	**0.89**	**0.73**	**0.84**	**0.73**	**0.71**
V_P	$\mathcal{A}_7$	Inverse Simple	2018	19	0.77	0.41	0.69	0.61	0.53
V_P	$\mathcal{A}_8$	Inverse Simple with context	2018	19	0.82	0.47	0.80	0.66	0.63

[1] We kept the number of output variables close to 30 (V_D) and 20 (V_P) to produce comparable metrics. Divergences are caused by the limited amount of available memory.

[2] To reduce training time for this particular architecture, we had to limit the number of parents per node to 15.

(known as the *accuracy paradox*). Thus, this metric is less of an indicator of quality, but rather points out if there are fundamental misconceptions in our approach.

Recall should be relatively high for low thresholds t, but decrease as less variables are considered to be true. Precision should show the inverse behaviour, i.e. starting low and increasing as the higher threshold filters out more false positives.

The ranking performance measures produce a good estimate of how accurate a result list similar to the one presented in Figure 4 is. There is a variety of other metrics measuring the quality of a ranking (most notably MAP, DCG and NDCG [26]), but for the external validation with non-experts, a straightforward and easy-to-interpret definition appeared more relevant to us. Recall should increase with the length of the ranking k, while the precision value should decrease, which is in accordance with the canonical definitions. A notable difference is that for $k < 5$, 100% recall is unlikely to be achieved since the vast majority of survey participants responded with the expected 5 problem/recall/effect tuples. Because of this effect, we report $rre(5)$ and $rpr(5)$ instead of the respective averages.

3) Results: The results of our internal evaluation are presented in Table IV. All architectures except $\mathcal{A}_1, \mathcal{A}_2$ perform clearly better than the baseline algorithm. This comparison is not entirely valid since both were evaluated on a different data set. However, our early experiments with the other architectures on 2014 data set, which we do not present here for the sake of brevity, have hinted at a similar disparity in performance. Consequently, the effectiveness of a manual cause/effect categorization is dubious at best and the benefits should be weighed carefully against the effort required during the manual coding process.

Overall, simple models, which resemble Naive Bayesian classifiers and do not use the manifest causality assumptions implied by survey design, perform much better than models with a complex cause effect chain. Their effectiveness can be improved further by making the V_o depend on relevant context variables, resulting in our best options $\mathcal{A}_8$ for V_D and $\mathcal{A}_6$ for V_P, achieving good average recall/precision tuples of 0.54, 0.83 and 0.73, 0.84, respectively. The accuracy of the produced rankings appears reasonable as well, although the ranking precision of the best diagnostic model is still disappointingly low (0.38).

Depending on a practitioner's needs, these metrics can be tuned by trading recall for precision and vice-versa. As visible in Figures 2 and 3, a wide range of values is achievable. Choosing the correct trade-off for a given application has been shown to be difficult [27], and further investigation with a systematic study would be necessary before suggesting a specific point on these curves.

Interestingly, architectures whose edges point towards V_o perform better in many cases than architectures whose edges point away from V_o: For V_D, an inverse model consistently achieves better metrics than its respective non-inverted model. This is not entirely true for V_P, where the Survey and Kalinowski architectures show diverging behaviour. We cannot provide a sound reason for this behaviour and whether it transfers to other applications remains to be seen.

Furthermore, the baseline algorithm exhibits interesting behavior: unlike expected, recall and precision fall in unison, meaning that no reasonable prediction can be achieved by simply naming the generally most probable problems or causes. Taking into account the specific circumstances of a project is thus of paramount importance for risk management.

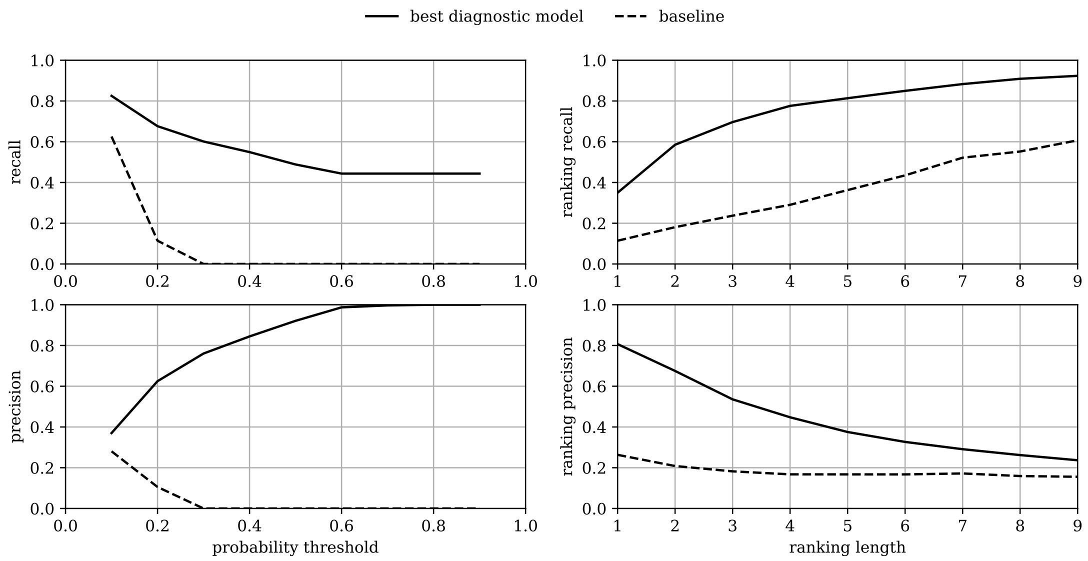


Fig. 2. Metrics of the best diagnostic model $\mathcal{A}_8$ and the baseline model.

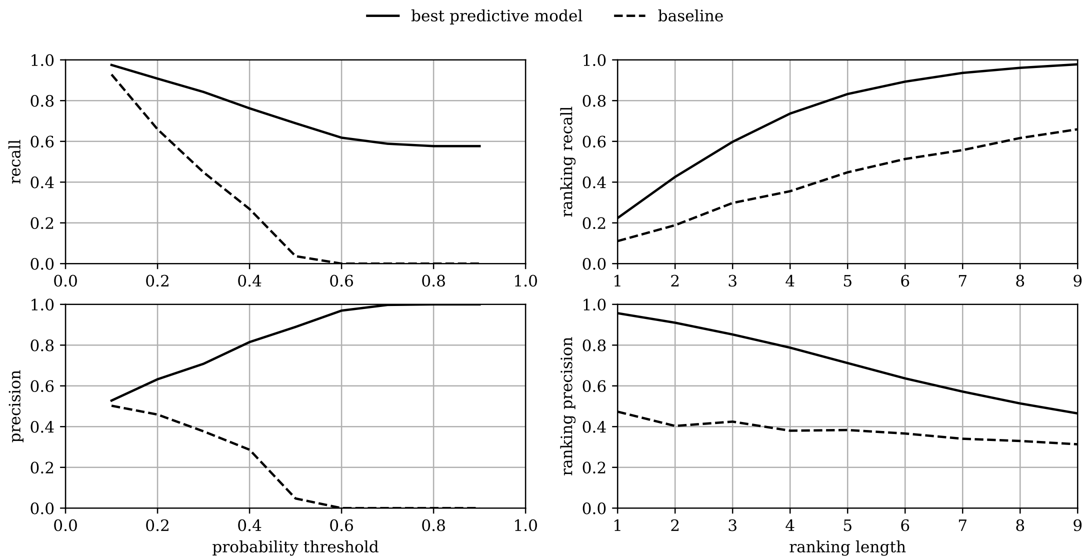


Fig. 3. Metrics of the best predictive model $\mathcal{A}_6$ and the baseline model.

4) Threats to validity: The above results rely on a series of assumptions that need to be challenged. Our selection of architectures is by its very nature limited and does not cover all possibilities to connect the variable types. Even our definition of an architecture limits the number of Bayesian Networks which were evaluated because the basic definition does not force variables of the same type to be connected homogeneously. Furthermore, the choice of filter values by experience to reduce the computational load is, to a certain degree, arbitrary. Better or worse results could possibly be achieved by simply using different values.

Due to combinatorial effects, however, it is unlikely that an extensive evaluation will ever be possible and such computational trade-offs are a necessity. By starting with very simplistic architectures ($\mathcal{A}_5$–$\mathcal{A}_8$) whose filters we only tuned as far as necessary to be able to yield actual results, we established at least a lower bound on what should be possible to achieve on the given data sets. Adding the architectures known in literature ($\mathcal{A}_1$–$\mathcal{A}_4$) helps to validate previously unchallenged assumptions.

A last threat emerges from the data set itself. There are no guarantees that the pre-defined problems and manually coded effects and causes are an internally coherent representation.

Indeed, a simple random inspection of these items yielded a number of overlapping causes such as *Lack of project management*, *Poor project management*, *Missing project management*; clearly, if one of them is present, the other causes should be present as well. In addition, there is not always a clear causality: *Difficulties in project management*, which is basically synonymous to the above causes, is listed as an effect. There are other similar ambiguities in the data set whose effect is difficult to assess.

A minor point is that the discretization of certain context variables naturally resulted in a reduction of the precision of our input data. The break points were chosen to be equiprobable, but there is no strong evidence that these break points are an inherently good choice.

B. External validity

1) Research questions: To evaluate the external validity, i.e. to assess the validity and perceived usefulness of our approach when applied in a realistic context, we are interested in the following research questions:

- **RQ1:** How do the outputs of our tool compare with the assessment of an RE expert for a specific project?
- **RQ2:** How do RE experts assess the usefulness of the tool and the style of presentation?
- **RQ3:** In which contexts and for which ends would RE experts apply the tool?

2) Study design: To answer the research questions, we conducted a case study in the context of a German software consulting company. We contacted an RE expert of the company and asked him whether he is interested in giving feedback on a tool for RE risk prediction. After he agreed, we asked him to think of a current or past RE project and offered him support for one of the two addressed use cases.

The data collection for the case study was conducted as a semi-structured interview. Two authors and the RE expert participated in the interview that lasted around 60 minutes. The whole interview was recorded to support a detailed analysis. The interview comprised the following three parts:

Part 1: We asked the RE expert to describe the case project, name the major problems he encountered and list their causes according to his opinion.

Part 2: Only now did we introduce the RE expert to our tool (as depicted in Figure 4), backed by the model performing best in the internal validation (Section V-A). We went through the problems and effects offered by the tool and asked the RE expert whether any of them were present in the project. Afterwards, the tool returned a list of the 5 most likely causes together with their predicted probabilities. As additional information, the tool offered a visualization of all performance metrics and a graphical representation of the underlying graph. We asked the expert to assess the results of the tool in terms of precision, completeness, and level of abstraction.

Part 3: Finally, we asked the RE expert to discuss following questions about the approach in general and the tool's particular result presentation:

1) Which information presented in the user interface of our tool does the expert consider to be important, what information is missing to make informed decisions?
2) What is more relevant to the RE expert, precision of the presented results or recall? Does the expert favor a bounded list of top-X results or does he prefer a variable-length list of results above a certain probability threshold?
3) How does the RE expert assess the impact of such data-driven predictions on his personal decisions?
4) What is the most relevant target group for the approach? Which target groups may not benefit from the approach?

3) Case Description: Our study participant is an RE consultant with 20 years of experience in general IT projects. During his early years, he worked on domain and business process modeling before focusing on testing and quality engineering. For three years, he has been working exclusively as an RE consultant.

During the case study, he was interested in analyzing a recently finished project. His role was to coach the client company's product owners for one year. The company had recently decided to move towards more agile practices.

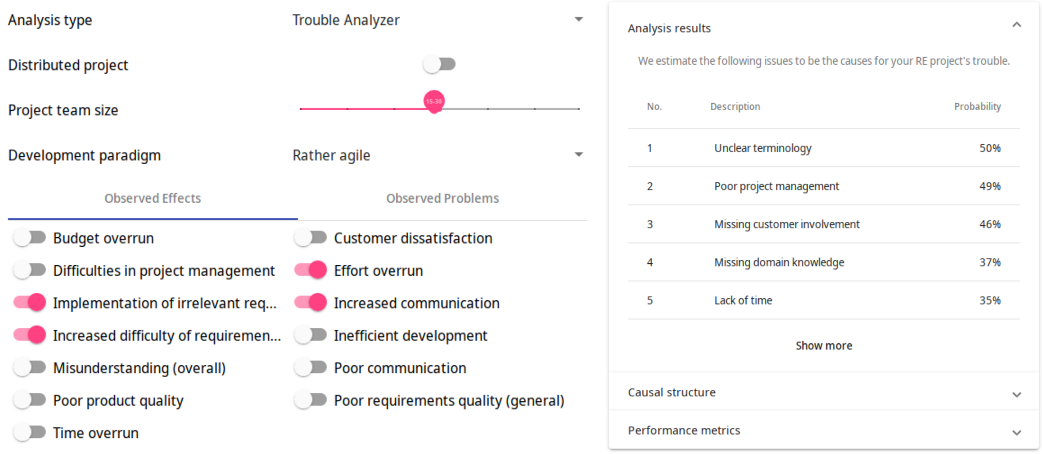
The project itself was concerned with the enhancement of an automation portal for a "digital factory" with mainly automotive products. 250 team members distributed over several locations (Germany, Eastern Europe, India) were involved, a large team compared to the size in the NaPiRE dataset. The applied development process was Scrum and our study participant assessed his relation to the customer as mostly neutral with better and worse moments.

According to him, the project lends itself to a *Post-Mortem-Analysis* (diagnostic reasoning) to identify causes of several RE problems he experienced in the course of the project.

4) Study results: In the following, we present the results and relate them to our research questions.

RQ1 – performance: Table V lists the causes and the related problems named by the interviewee. We entered the evidence reported in Table VI into the tool. Of the predicted top-10 causes (Table VII), our expert confirmed 7 and rejected 3. These false positives also include the cause with the highest probability (*missing domain knowledge*), which was not an issue at all in the project. Besides the false positives, the expert assessed the mentioned causes as good matches. Especially, *Lack of a well-defined RE process* was a top match. He also confirmed the cause *Poor project management*. However, he considered this cause to be too coarse-grained without a more detailed definition, which is not provided by the data set.

RQ2 – usefulness: The general usage of the tool was considered straightforward and the presentation of the results as a ranking was perceived to be very appropriate. The expert judges the presentation of five items on the list as a good choice because it is long enough to allow a variety of possible causes to be presented while still being tractable in group discussions. For this reason, the precision of these five predictions is of high importance, much more than achieving high recall and covering the majority of all causes that might be present.



Analysis type	Trouble Analyzer
Distributed project	○
Project team size	15-30
Development paradigm	Rather agile

Observed Effects	Observed Problems
○ Budget overrun	○ Customer dissatisfaction
○ Difficulties in project management	● Effort overrun
● Implementation of irrelevant req...	● Increased communication
● Increased difficulty of requiremen...	○ Inefficient development
○ Misunderstanding (overall)	○ Poor communication
○ Poor product quality	○ Poor requirements quality (general)
○ Time overrun	

Analysis results

We estimate the following issues to be the causes for your RE project's trouble.

No.	Description	Probability
1	Unclear terminology	50%
2	Poor project management	49%
3	Missing customer involvement	46%
4	Missing domain knowledge	37%
5	Lack of time	35%

Fig. 4. Data input form and output presentation (cause analysis use case)

TABLE V
PROBLEMS AND CAUSES EXPERIENCED BY THE RE EXPERT

Problem	Causes
Bad team communication	Top-down implementation of Scrum No culture of failure Traditional company culture
Product owners were not responsible	Agile teams setup according to system components, not features. Team setup was immutable
Poor requirements quality	Poor knowledge about RE No QA for requirements
Very technical user stories	Poor knowledge about agile methodology Silent rejection of the agile methodology

TABLE VI
EVIDENCE PROVIDED BY THE RE EXPERT

Evidence
Problems
Poor communication
Poor product quality
Difficulties in project management
Misunderstandings (overall)
Poor requirements quality (general)
Effects
Underspecified requirements
Weak relationship between customer and project team
Communication flaws within the project team
Insufficient support by customer
Weak access to customer needs
Incomplete or hidden requirements
Stakeholders with difficulties in separating requirements from solutions
Unclear/unmeasurable non-functional requirements

TABLE VII
TOOL PREDICTIONS AND EXPERT CONFIRMATION

Rank	Cause	Conf
1 (53%)	Missing domain knowledge	✗
2 (52%)	Missing customer involvement	✓
3 (52%)	Lack of a well-defined RE process	✓
4 (52%)	Poor project management	✓
5 (52%)	Lack of time	✗
6 (51%)	Lack of requirements management	✓
7 (48%)	Lack of experience of RE team members	✓
8 (44%)	Communication flaws between team and customer	✓
9 (31%)	Poor requirements elicitation techniques	✓
10 (29%)	Lack of communication channels	✗

The presentation of probabilities along the ranking was perceived as a good means to communicate the approximate risk of following the tool's suggestions, although we should have stated more clearly that such a device cannot replace a fully-fledged RE process assessment and only provides very general hints at what might be going wrong in a project. When presenting the option to show the predicted probabilities for all causes, we observed an interesting effect: every item on this long list was interpreted as prediction of the tool by the expert, regardless of the accompanying probability.

We concluded that, while a ranking of a given length is the preferable way to display the inference results, a hybrid presentation approach is more suitable: the list should be cut off at a given probability threshold so users are not tempted to consider items with an evidently low probability.

RQ3 – applicability: The RE expert proposed three primary target groups using the tool for the following purposes.

- Classic projects with a project lead: Discussion input to improve the development process,
- Agile teams: Discussion input to improve the development process, e.g., during the Scrum Retrospective,
- Teams without or with inexperienced requirements engineer: Highlighting of low-hanging fruits to develop a more sophisticated RE process.

These scenarios align with his perception that the tool's result were most likely to influence group decisions, and are less likely to be able to have a tangible effect on an individual's judgements.

5) Threats to validity: The above paragraphs must be seen in the light of the chosen study design: the results of case studies are inherently difficult to generalize. Despite carefully choosing an experienced participant bringing insights from a variety of projects in Software and Requirements Engineering, we conclude from the case description that the findings are probably biased towards agile methodology in larger companies and that experiences in other contexts may differ.

Another issue of case study designs are psychological biases, whose abundance forces us to focus on a selected few. Courtesy bias is hard to exclude during an interview, so most probably the above results judge our tool more positively than justifiable. Consistency bias cannot be ruled out, either. We separated the manual analysis of the situation (*part 1* of the interview) from the tool introduction (*part 2*), but participants would still strive to interpret the tool's predictions in a way that would produce a consistent description of the situation. In particular, generic items like *Poor project management* lend themselves to such adjustments. We estimate that groupthink is less of an issue due to the participant's relatively independent position as a consultant in the project.

In the future, these concerns should be addressed by (1) interviewing RE experts from other backgrounds to increase our coverage of different contexts, and (2) making the tool generally available online and combining it with a questionnaire to allow anonymous feedback without a human interviewer to alleviate psychological biases.

VI. Conclusions

Based on the NaPiRE data set, we trained a series of Bayesian Networks to model cause-effect relationships in RE projects with different contextual characteristics. These models were firstly used to conduct a post-mortem analysis, deriving probable causes of sub-optimal RE performance, and secondly to conduct a preventive analysis, predicting probable issues a young project might encounter. The method was subject to a rigorous cross-validation procedure for both use cases before assessing its applicability to real-world scenarios with a case study.

Generally, the results are promising. For both use cases, we achieve good recall/precision values with simple network architectures neglecting the causal structure implied by the underlying data set. The same is true for the quality of probability-based rankings of predicted items produced by the networks, except for the precision of the rankings for the post-mortem analysis.

The case study involving a user-friendly interface to these models is equally supportive. The predicted causes generally matched the causes predicted by the interviewed RE expert and the presentation as a ranking was perceived as useful, although minor improvements remain. Precision was determined to be the driving performance metric in this context; unfortunately, this is the one metric in which our models perform sub-optimally. We identified a number of applications for our tool: In both, classic and agile projects its predictions can serve as valuable discussion input to improve the RE process. Moreover, it can help inexperienced teams to focus on the most worthwhile RE process and technique enhancements.

The internal validation questions the causality assumptions behind the design of the problems/causes/effects section in the NaPiRE survey: the fact that neglecting them yields considerably better performance puts into question whether what survey participants qualify as a cause actually is a cause, and whether what they qualify as an effect actually is an effect. Given that distinguishing these is a notoriously difficult task even for scientists (with a large number of theoretical approaches and a variety of pitfalls such as spurious correlation), it might be too much to ask survey participants to always identify the causal chain correctly.

Loosening these assumptions in the survey, i.e. asking for correlations only, possibly supplying a pre-defined list of causes and effects instead of the manual coding effort, and using undirected graphical models with a less strict structure should help improve prediction quality.

References

[1] M. Glinz, "A glossary of requirements engineering terminology: Version 1.7," 2017. [Online]. Available: https://www.ireb.org/content/downloads/1-cpre-glossary/ireb_cpre_glossary_17.pdf

[2] D. Méndez Fernández, S. Wagner, M. Kalinowski, M. Felderer, P. Mafra, A. Vetrò, T. Conte, M.-T. Christiansson, D. Greer, C. Lassenius, T. Männistö, M. Nayabi, M. Oivo, B. Penzenstadler, D. Pfahl, R. Prikladnicki, G. Ruhe, A. Schekelmann, S. Sen, R. Spinola, A. Tuzcu, J. L. de la Vara, and R. Wieringa, "Naming the pain in requirements engineering," *Empirical Software Engineering*, vol. 22, no. 5, pp. 2298–2338, 2017.

[3] I. M. del Águila and J. del Sagrado, "Bayesian networks for enhancement of requirements engineering: A literature review," *Requirements Engineering*, vol. 21, no. 4, pp. 461–480, 2016.

[4] D. Méndez Fernández, "Supporting Requirements-Engineering Research That Industry Needs: The NaPiRE Initiative," *IEEE Software*, vol. 35, no. 1, pp. 112–116, Jan. 2018.

[5] S. Wagner, D. Méndez Fernández, M. Felderer, A. Vetrò, M. Kalinowski, R. Wieringa, D. Pfahl, T. Conte, M.-T. Christiansson, D. Greer, C. Lassenius, T. Männistö, M. Nayebi, M. Oivo, B. Penzenstadler, R. Prikladnicki, G. Ruhe, A. Schekelmann, S. Sen, R. Spínola, A. Tuzcu, J. L. D. L. Vara, and D. Winkler, "Status quo in requirements engineering: A theory and a global family of surveys," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 28, no. 2, pp. 9:1–9:48, Feb. 2019.

[6] D. Méndez Fernández, S. Wagner, M. Kalinowski, A. Schekelmann, A. Tuzcu, T. Conte, R. Spinola, and R. Prikladnicki, "Naming the pain in requirements engineering: Comparing practices in Brazil and Germany," *IEEE Software*, vol. 32, no. 5, pp. 16–23, 2015.

[7] M. Kalinowski, P. Curty, A. Paes, A. Ferreira, R. Spinola, D. Méndez Fernández, M. Felderer, and S. Wagner, "Supporting defect causal analysis in practice with cross-company data on causes of requirements engineering problems," in *39th International Conference on Software Engineering (ICSE)*, 2017, pp. 223–232.

[8] D. Méndez Fernández, M. Tießler, M. Kalinowski, M. Felderer, and M. Kuhrmann, "On Evidence-Based Risk Management in Requirements Engineering," in *Software Quality: Methods and Tools for Better Software and Systems*, ser. Lecture Notes in Business Information Processing, D. Winkler, S. Biffl, and J. Bergsmann, Eds. Springer International Publishing, 2018, pp. 39–59.

[9] A. T. Misirli and A. B. Bener, "Bayesian networks for evidence-based decision-making in software engineering," *IEEE Transactions on Software Engineering (TSE)*, vol. 40, no. 6, pp. 533–554, 2014.

[10] A. Tosun, A. B. Bener, and S. Akbarinasaji, "A systematic literature review on the applications of Bayesian networks to predict software quality," *Software Quality Journal*, vol. 25, no. 1, pp. 273–305, 2017.

[11] Y. Tang, K. Feng, K. Cooper, and J. Cangussu, "Requirement engineering techniques selection and modeling an expert system based approach," in *International Conference on Machine Learning and Applications*, 2009, pp. 705–709.

[12] A. Nagy, M. Njima, and L. Mkrtchyan, "A Bayesian based method for agile software development release planning and project health monitoring," in *International Conference on Intelligent Networking and Collaborative Systems*, 2010, pp. 192–199.

[13] D. N. Card, "Defect-causal analysis drives down error rates," *IEEE Software*, vol. 10, no. 4, pp. 98–99, 1993.

[14] P. F. Wilson, *Root Cause Analysis: A Tool for Total Quality Management*. ASQ Quality Press, 1993.

[15] M. Solé, V. Muntés-Mulero, A. I. Rana, and G. Estrada, "Survey on models and techniques for root-cause analysis," 2017.

[16] N. A. Ernst and G. C. Murphy, "Case studies in just-in-time requirements analysis," in *2nd IEEE International Workshop on Empirical Requirements Engineering (EmpiRE)*, 2012, pp. 25–32.

[17] A. M. Davis, *Just Enough Requirements Management: Where Software Development Meets Marketing*. New York, NY, USA: Dorset House Publishing Co., Inc., 2005.

[18] M. Glinz, "A risk-based, value-oriented approach to quality requirements," *IEEE Software*, vol. 25, no. 2, pp. 34–41, 2008.

[19] Y. Asnar, P. Giorgini, and J. Mylopoulos, "Goal-driven risk assessment in requirements engineering," *Requirements Engineering*, vol. 16, no. 2, pp. 101–116, 2011.

[20] L. Grunske and D. Joyce, "Quantitative risk-based security prediction for component-based systems with explicitly modeled attack profiles," *Journal of Systems and Software (JSS)*, vol. 81, no. 8, pp. 1327–1345, 2008.

[21] T. Koski and J. Noble, *Bayesian Networks: An Introduction*, 1st ed., ser. Wiley Series in Probability and Statistics. Chichester, West Sussex, UK: Wiley, Sep. 2009.

[22] D. Mendez, S. Wagner, M. Kalinowski, M. Felderer, P. Mafra, A. Vetro, T. Conte, M.-T. Christiansson, D. Greer, C. Lassenius, T. Männistö, M. Nayebi, M. Oivo, B. Penzenstadler, D. Pfahl, R. Prikladnicki, G. Ruhe, A. Schekelmann, S. Sen, R. Spinola, J.-L. de la Vara, A. Tuzcu, and R. Wieringa, "NaPiRE data set 2014," 2018. [Online]. Available: https://figshare.com/articles/NaPiRE_Data_Set_2014/5845083

[23] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, 2017.

[24] T. Wheeler and M. Kochenderfer, "Sisl/BayesNets.jl: Bayesian networks for Julia," Stanford Intelligent Systems Laboratory, 2019. [Online]. Available: https://github.com/sisl/BayesNets.jl

[25] E. R. Gansner and S. C. North, "An open graph visualization system and its applications to software engineering," *Software: Practice and Experience*, vol. 30, no. 11, pp. 1203–1233, 2000.

[26] T.-Y. Liu, "Learning to rank for information retrieval," *Foundations and Trends in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.

[27] J. P. Winkler, J. Grönberg, and A. Vogelsang, "Optimizing for recall in automatic requirements classification: An empirical study," in *27th IEEE International Requirements Engineering Conference (RE)*, 2019.