TUDelft

Delft University of Technology

Factors Affecting On-Time Delivery in Large-Scale Agile Software Development

Kula, Elvan; Greuter, Eric; Van Deursen, Arie; Georgios, Gousios

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Factors Affecting On-Time Delivery in Large-Scale Agile Software Development

Elvan Kula [ID], *Member, IEEE*, Eric Greuter, Arie van Deursen [ID], *Member, IEEE*, and Georgios Gousios [ID]

Abstract—Late delivery of software projects and cost overruns have been common problems in the software industry for decades. Both problems are manifestations of deficiencies in effort estimation during project planning. With software projects being complex socio-technical systems, a large pool of factors can affect effort estimation and on-time delivery. To identify the most relevant factors and their interactions affecting schedule deviations in large-scale agile software development, we conducted a mixed-methods case study at ING: two rounds of surveys revealed a multitude of organizational, people, process, project and technical factors which were then quantified and statistically modeled using software repository data from 185 teams. We find that factors such as requirements refinement, task dependencies, organizational alignment and organizational politics are perceived to have the greatest impact on on-time delivery, whereas proxy measures such as project size, number of dependencies, historical delivery performance and team familiarity can help explain a large degree of schedule deviations. We also discover hierarchical interactions among factors: organizational factors are perceived to interact with people factors, which in turn impact technical factors. We compose our findings in the form of a conceptual framework representing influential factors and their relationships to on-time delivery. Our results can help practitioners identify and manage delay risks in agile settings, can inform the design of automated tools to predict schedule overruns and can contribute towards the development of a relational theory of software project management.

Index Terms—Software engineering management, effort estimation, empirical studies, software companies

✦

1 INTRODUCTION

L ATE delivery and cost overruns have been common problems in the software industry for decades. On average, software projects run around 30 percent overtime [1]. This percentage does not seem to have decreased since the 1980s [2]. Even though effort estimation is at the heart of almost all industries, it is especially challenging in the software industry. This is mainly due to the fact that software development is a complex undertaking, affected by a variety of social and technical factors. The overall perceived success of a software project depends heavily on meeting the time and cost estimates [3]. Improving effort estimation is therefore a critical goal for software organizations: it can help companies reduce delays and improve customer satisfaction, while enabling them to efficiently allocate resources, reduce costs and optimize delivery [4], [5]. In spite of the availability of many estimation methods and guidelines [6], [7], on-time delivery in software development remains a major challenge. Prior research identified a large number of factors that may influence the software development effort

[8], but which factors have the most impact is not clear. We lack an understanding of the relationships between these factors and how they impact on-time delivery.

Effort estimation is also a major challenge in agile software development. Prior work [9] has found that around half of the agile projects run into effort overruns of 25 percent or more. In agile settings, software is incrementally developed through short iterations to enable a fast response to changing markets and customer demands. Agile projects leverage short-term, iterative planning in which effort estimates are progressively refined [10]. A particular challenge involves combining the flexible, short-term agile planning setting with the business needs for long term planning of availability of large pieces of functionality (often referred to as "epics" [11]). Most agile teams heavily rely on experts' subjective assessment of team- and project-related factors to arrive at an estimate [12], [13]. However, these factors remain largely unexplored [13]; further analysis is required to investigate influential factors and how they impact delays in agile projects.

By identifying and investigating influential factors, we can obtain valuable insights on what data and techniques are needed to become more predictable at delivering software in agile settings. An identification of the most influential factors can help software organizations increase the effectiveness and efficiency of scheduling strategies by concentrating measurement and risk management activities directly on those factors that have the greatest impact on on-time delivery. Such knowledge can also guide future research on building and evaluating software effort estimation techniques, methods and tools. Furthermore, a deeper understanding of the interactions between influential factors can help in identifying the root causes of delays, and developing tools and

---

- *Elvan Kula is with the Delft University of Technology, 2628, CD, Delft, Netherlands, and also with the ING Tech, 1102, MG, Amsterdam, The Netherlands. E-mail: E.Kula@tudelft.nl.*
- *Eric Greuter is with the ING Tech, 1102, MG, Amsterdam, The Netherlands. E-mail: Eric.Greuter@ing.com.*
- *Arie van Deursen and Georgios Gousios are with the Delft University of Technology, 2628 Delft, The Netherlands. E-mail: {Arie.vanDeursen, G.Gousios}@tudelft.nl.*

guidelines that can assist software organizations in improving their on-time delivery performance.

The goal of this paper is to identify the most relevant factors and their interactions that affect schedule deviations in large-scale agile software development. To do so, we conduct a case study at ING, a large Dutch internationally operating bank with more than 15,000 developers. The teams at ING develop software using an agile development process. They work with epics to manage interdependent software deliveries across multiple teams and iterations. We follow a mixed-methods approach in which we combine expert- with data-based strategies to derive, confirm and investigate factors that impact the timelines of epic deliveries. We conduct a survey with 635 software experts from ING, and we analyze historic repository data from 185 teams and 2,208 epics to corroborate the survey findings. We extract proxy measures from repository data that we map to the perceived influential factors and analyze their importance in schedule deviations. Throughout our study, the following two research questions guide our work:

RQ1. *Factor identification:* Which factors are perceived to affect the timeliness of deliveries (RQ1.1), what is their perceived level of impact (RQ1.2), and what are the perceived types of interactions between these factors and on-time delivery (RQ1.3)?

RQ2. *Factor validation:* How do the perceived influential factors impact schedule deviation in deliveries?

Our survey results show that requirements refinement, task dependencies, organizational alignment, organizational politics and the geographic distribution of teams are the factors that are perceived to have the greatest impact on timely delivery. We find that factors interact hierarchically: organizational factors interact with people factors, which in turn impact the technical factors. The technical factors are perceived to have a direct impact on the timeliness of software delivery. Our data analysis reveals that the project size, number of task dependencies, historical delivery performance, team familiarity and developer experience are the most important proxy measures that explain the schedule deviations in deliveries.

By answering the research questions, we create a conceptual framework representing 25 factors and their interactions that are perceived to affect the timely delivery of software at ING. The main contributions of this paper are:

- A set of *factors and their interactions* affecting the timely delivery of software in large-scale agile development. We order the factors by their relevance.
- A *conceptual framework* of on-time delivery that represents influential factors and their interactions. This framework suggests multiple paths for action that may improve the timeliness of software deliveries.

2 BACKGROUND

On-time delivery in software development remains a major challenge and important topic of interest since improving it has a large economic benefit. Considerable research has been directed at identifying factors that cause schedule
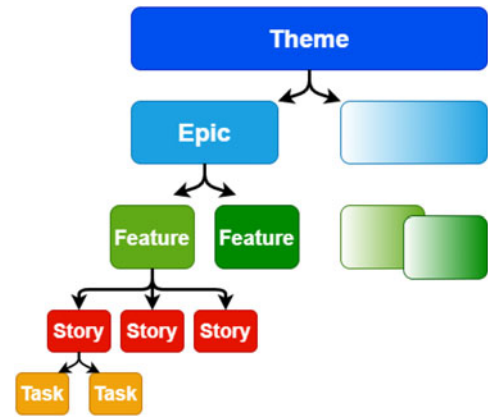

Fig. 1. Agile work breakdown structure.

overruns in software projects. This work can be found in the research areas of effort estimation and software project risk management. In this section, we provide background on large-scale agile software development, and we discuss research on effort drivers and software project risks that lead to schedule overruns. We also provide details on the development context of the case company.

2.1 Large-Scale Agile Software Development

A common way for agile companies to express user requirements is based on a five-level hierarchy introduced by Leffingwell [11] (shown in Fig. 1). Within strategic focus areas called *strategic themes*, *epics* form high-level functional goals for the product(s) [10]. Epics represent a large body of work that can be split into *features*, which in turn can be split into *user stories*. Stories are short requirements or requests written from the perspective of an end user [14]. Finally, user stories are refined into development *tasks*, which denote the technical work that needs to be done for a user story. Agile teams work with a product backlog to keep track of the status and business priority of work items [15].

Various agile planning frameworks for large projects (multi-team, multi-month) have been implemented successfully in large-scale software organizations [16], [17], [18]. These frameworks rely on three levels of planning: release, iteration and daily planning [10]. The release plan (usually 2-6 months) centers on epics [10], which often encompass multiple teams and span over multiple iterations (e.g., *sprints* in Scrum [19]). An iteration is a short, fixed-length period (usually 1-4 weeks) in which a single development team delivers a number of user stories.

2.2 Factors Influencing Software Development Effort

Prior work analyzed a variety of factors influencing the software development effort (so-called *effort drivers*). Trendowicz *et al.* [20] reviewed and divided the most commonly used effort drivers into four categories: personnel factors (i.e., team capabilities and experience [5], [9], [21], [22]), process factors (i.e., quality of methods, tools and technologies applied [23], [24], [25]), project factors (i.e., project resources and management activities [26], [27]) and product factors (i.e., effort for requirements analysis, design and coding [28]). Personnel factors and project factors are the top mentioned effort drivers in

agile projects [12]. As observed in previous studies [20], [29], the accuracy of effort estimation methods depends on the selection of relevant factors and the elimination of irrelevant and misleading factors.

Existing estimation methods can generally be classified into expert-based and model-based approaches [5], [30]. Expert-based methods rely on human expertise to select relevant factors, and are the most popular technique in both agile and traditional (waterfall-like) projects [5], [31]. Model-based methods leverage data from past projects covering a certain initial set of factors in order to identify a subset of factors that are relevant. Both methods have significant practical limitations when applied individually. Experts decisions heavily rely on subjective assessment, which may lead to inaccuracy and inconsistencies between estimates. The effectiveness of data-based methods, on the other hand, largely depends on the quantity and quality of available data. It is therefore likely that no single strategy will be the best performer for all settings [5]. Hence, recent works [32], [33] propose to combine expert-based and data-based methods – similar to this work.

Machine learning models have gained popularity as an alternative to model-based estimation methods. They have achieved promising results in estimating effort for software projects [34], [35], [36], [37], [38], and predicting the elapsed time required for bug-fixing or resolving an issue [39], [40], [41], [42], [43], [44]. Our study of factors affecting schedule deviation in epics can provide further insights into important effort drivers that can contribute to the improvement of estimation methods.

2.3 Software Project Risk Factors

Risk factors are uncertain events that can pose a serious threat to the successful completion of a software project [45]. Several studies point out that ineffective risk management is one of the main reasons for schedule overrun in software projects [46], [47], [48], [49], [50]. Risk management consists of two main activities: risk assessment and risk control. Our current work focuses on risk assessment, which is the process of identifying risks, estimating the likelihood of their occurrence and evaluating their potential effects [51]. In this work, we identify risk factors that lead to schedule overrun in software deliveries and we evaluate their relative effects.

Seminal work in the area of risk assessment has been carried out by Boehm [51] ('Top 10 Software Risks") and the Software Engineering Institute [52] ("Taxonomy-Based Risk Identification"). Examples of the most common risk factors are: unclear or changing requirements (e.g., [51], [53], [54], [55]), underestimated project complexity (e.g., [56], [57]), an unstable organizational environment (e.g., [58], [59]), lack of management support (e.g., [57], [60]), bad commitment of the user ([57], [61]) and personnel shortfalls (e.g., [51], [62]). Other studies focused on measurement of (the level of) risk [56], understanding the nature and types of risks [57] and mitigation of risk components [63]. Wallace *et al.* [64] reviewed literature in the field and identified six dimensions of software project risks: organizational environment, user, requirements, project complexity, planning & control and team risk. They analyzed the relationships between risk dimensions and overall project performance. More recently, Menezes *et al.* [65] identified 148 risk
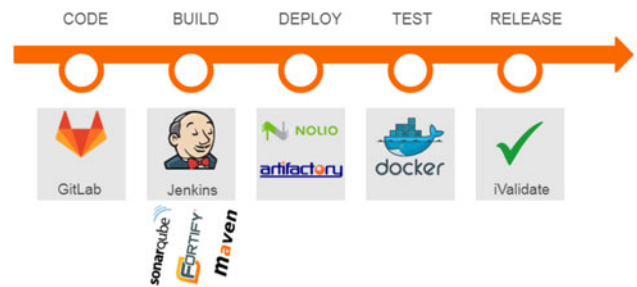

Fig. 2. Continuous delivery pipeline at ING.

factors through a systematic literature review and classified them according to the SEI Taxonomy [52]. Related research [66], [67], [68] has also been done in using statistical analysis for evaluating risk factors in software projects.

While various risk checklists and frameworks have been proposed in earlier work, the relationships between risk factors and project delay remain largely unexplored. Our study complements prior work by providing further insights into the relationships between factors and their implications for on-time delivery performance.

2.4 The Case Company

To address our research questions, we conducted a case study at ING TECH, the largest IT department within ING that consists of 295 development teams distributed over Europe, Asia and North America. TECH is responsible for the development of ING's main banking platforms and advisory services that are being used by millions of customers worldwide. The department has significant variety in terms of the products developed, the size and application domain, as well as the programming languages used.

In recent years, ING has reinvented its organisational structure, moving from traditional functional departments to a completely agile organisational structure based on Spotify's '*Squads, Tribes and Chapters*' model [69]. The main purpose of this model is to scale agile to hundreds of development teams. Each development team consists of 5 to 9 members, and has a Scrum master and a product owner. The teams at ING TECH practice DevOps and follow Scrum as agile methodology. They work in sprints of 1 to 4 weeks. To ensure that code can be deployed fast to production, ING put a fully automated software delivery pipeline in place. All development teams make use of the pipeline. The delivery pipeline contains several specialized tools, as depicted in Fig. 2.

At ING TECH, epics are usually delivered in a time span of one to four quarters, i.e., three to 12 months. Epics should either be delivered as a whole within a quarterly cycle or, in case of larger epics, in incremental software releases. The planning activities are led by tribe leads with active engagement of product owners to manage inter-team dependencies.

3 RESEARCH METHOD

Our research method consists of an exploratory and confirmatory phase. In the exploratory phase, we developed and distributed a survey to software experts to identify factors that are perceived to affect the on-time delivery of epics (RQ1.1) and types of factor interactions (RQ1.3). In the confirmatory phase, we applied data triangulation to corroborate

the respondents' perceptions and to extract more detailed insights into the effects of influential factors. We conducted a second survey with a different sample of software experts to order influential factors by their perceived level of impact (RQ1.2), and we performed regression analysis using repository data to validate the impact of factors (RQ2).

3.1 Collecting and Analyzing Survey Data

The main goal of the surveys was to gather the perceptions of software experts at ING on factors affecting the timeliness of epic deliveries and how much of an impact they have. To design and execute our surveys, we followed methodological guidelines from Kitchenham and Pfleeger [70], and Kasunic [71], for survey research in software engineering.

3.1.1 Survey Design

We developed two self-administered online surveys, which were composed of a mix of closed and open-ended questions. The first survey's purpose was to *identify influential factors and their interactions*, and the second survey was used to *assess the perceived level of impact of each of the identified factors* from the first survey. The surveys were organized into two sections: a section aimed at gathering demographic information and a section targeting the research questions.[1] We kept the number of survey questions to a minimum as shorter questionnaires have been found to receive higher response rates [70].

The demographic sections of the surveys consisted of multiple-choice questions on the respondent's role, overall experience in software development and experience within ING. These demographic characteristics are important to assess the representativeness of participants [71] and they have been shown to influence the reasons given for effort estimation errors in related work [73]. The research related section of the first survey contained open-ended questions to gather unbounded and detailed responses on influential factors and types of factor interactions. For RQ1.1, we asked experts *which* factors affect the timeliness of their teams' epic deliveries. For RQ1.3, we included a follow-up open-ended question asking *how* the reported factors influence the timeliness of epic deliveries. In the second survey, we asked experts to rate the impact level of each identified factor from the first survey (RQ1.2). We used four-point Likert scale questions from "no impact" to "large impact" to get specific responses. We also provided a separate "not applicable" optional response in case a factor was not relevant to respondents. Here we also provided respondents with a write-in question to probe for additional factors in case any new ones might appear; we received 109 responses to that question. We reviewed the responses manually and found that they were rephrasing one of the 25 factors or identifying a subcase of one of the factors.

We have taken multiple measures to make the survey questions understandable by the respondents [70]. We included a brief paragraph on the survey's start page featuring the purpose of the survey and an overview of the types of questions presented in each section. Furthermore, we

provided definitions of factors to participants in the second survey. The first two authors have considerable experience working in the company, and they were able to explain the factor definitions in vocabulary understood by the participants. Together with the last author, they also made sure that the survey questions were coherent and consistent [70]. To avoid leading questions and biasing respondents, we phrased and ordered the questions in a sequential order of activities [71]. The factors in the second survey were presented in random order to the participants to reduce ordering bias [74].

3.1.2 Survey Validation

After design, the survey instrument should be evaluated to display areas for improvement [70], [71]. We piloted both surveys with 25 randomly selected employees from ING TECH to refine the survey questions. The pilot versions included an additional open-ended question at the end of the survey asking respondents for feedback on the survey contents. The respondents' feedback allowed us to refine the survey questions. As part of the pilot run, we received 6 responses (24 percent response rate) for the first survey and 5 responses (20 percent response rate) for the second survey. No reminder emails were sent. The pilot of the first survey revealed that the wording of the survey question aimed at RQ1.3 was unclear. The question asked respondents about the types of relationships between factors, which respondents interpreted in different ways (e.g., sign of impact, causality versus correlation, direct or indirect link). Since we wanted to collect descriptive information about factor interactions and do the classification of relationships ourselves, we rephrased it as a more open-ended question in the final version of the survey. The initial version of the second survey disclosed that the names of some factors (e.g., task dependencies versus technical dependencies) were ambiguous to respondents. This prompted us to provide a list of factor definitions in the final version of the second survey.

3.1.3 Survey Execution and Sampling Strategy

Our target population was composed of the 2,850 employees that belong to the 295 development teams at ING TECH. All these teams work with epic deliveries and are therefore relevant for our study. We received access to a mailing list containing all team members, which became our sampling frame. We were able to identify participants based on their email address and we had an overview of teams (team names) they belong to. This enabled us to determine members' participation and link survey responses to teams' repository data for triangulation in RQ2.

As recommended in survey guidelines [70], [71], [75], we performed simple random sampling to obtain representative samples from our population. For our final surveys, we excluded the 50 employees solicited in the earlier pilot surveys from our sampling frame. The final version of the first survey was distributed to 1,400 employees (one half of the population) in October 2019. These employees were sampled uniformly at random across all teams at ING TECH. We received 298 responses (representing 237 teams), corresponding to a response rate of 21 percent. A majority (79 percent) of teams had one respondent, 16 percent had two

---

1. The final survey instruments can be found in the supplemental material [72].

respondents and remaining 5 percent had three respondents. The final version of the second survey was distributed to the other half of the population (another 1,400 employees) in November 2019. This second group did not include employees solicited in the first survey. We obtained 337 responses (representing 241 teams), corresponding to a response rate of 24 percent. A majority (72 percent) of teams had one respondent, 18 percent had two respondents, 9 percent had three respondents and the remaining 1 percent had four respondents.

As per our sampling plan for the surveys, the participants were invited using a personal invitation mail featuring the purpose of the survey and how its results can enable us to gain new knowledge of delay factors in epic deliveries. Participants had a total of two weeks to participate in the surveys. To follow up on non-responders [70], we sent reminder emails to those who did not participate yet at the beginning of the second week.

3.1.4   Survey Data Analysis

The data we analyzed in this paper comes exclusively from the responses to the final surveys (i.e., the first survey deployed in October 2019 and the second survey deployed in November 2019). The "not applicable" responses were omitted from the analysis set. For the analysis of RQ1.2, we used descriptive statistics to order factors by their perceived level of impact. For the analysis of RQ1.1 and RQ1.3, we performed *inductive coding* to summarize the results of the open-ended questions. Coding samples are provided as examples in the supplemental material [72].

Identifying Influential Factors. A common approach for transforming qualitative data into quantitative data is coding [76], [77]. For RQ1.1, we applied inductive coding (i.e., *inductive content analysis*) during two integration rounds to derive influential factors from the open-ended survey responses. Each code in our coding scheme represents an influential factor. We coded by statement and codes continued to emerge till the end of the process. In the first round, the first and the last author used an online spreadsheet to code a 10 percent sample (30 mutually exclusive responses) each. They assigned at least one and up to four codes to each response. Next, the first and last author met in person to integrate the obtained codes, meaning that similar codes were combined or merged, and related ones were generalized or specialized if needed. When new codes emerged, they were integrated in the set of codes. The first author then applied the integrated set of codes to 90 percent of the answers and the last author did this for the remaining 10 percent of the responses. In the second round, the two authors had another integration meeting which resulted into the final set of codes. The final set contained three (13 percent) more codes than the set resulting from the first integration round. We computed percent agreement and Cohen's kappa [78] to assess inter-coder reliability on the final coding scheme. We measured substantial agreement between the coders: percent agreement = 86 percent and $\kappa = 0.72$. Then, together, the two authors grouped the factors into the five categories identified in the work of Chow *et al.* [3]. The resulting 25 codes and five categories are summarized in Table 1.

Classifying Types of Factor Relationships. For RQ1.3, we analyzed open-ended survey responses to investigate the perceived types of relationships between influential factors and on-time delivery. We took the same approach as Jorgensen and Molokken-Ostvold [73]. We focused on direct, indirect and contributory relationships to make a distinction between simple, complex and condition-dependent types of interactions between factors. More discussions on types and interpretations of reasoning models can be found in the work of Pearl [79]. Possible interpretations of a factor X being a reason for schedule deviation are:

- There is a *direct* link between X and schedule deviation (i.e., X is a direct reason for deviation). We classified a factor as having a *direct relationship* with on-time epic delivery if it is explained to be an immediate reason for schedule deviation. For example, "unmanaged dependencies" is a reason that may immediately lead to unplanned waiting time and thus delay in an epic.

- There is an *indirect* relationship between X and schedule deviation (i.e., X leads to events that, in turn, lead to deviation). We classified a factor as having an *indirect relationship* with on-time delivery if it is explained to affect schedule deviation through other factors or events. For example, "lack of organizational trust" may lead to "errors during handoffs", which in turn may result in "unmanaged dependencies". "Lack of organizational trust" and "errors during handoffs" are both indirect reasons of different distance to the direct reason "unmanaged dependencies".

- The events leading to schedule deviation would have been harmless without X (i.e., X is a *contributory reason* for schedule deviation). We classified a factor as having a *contributory relationship* with on-time delivery if it is described as a necessary condition for schedule deviation rather than a direct or indirect reason. Assuming that "unmanaged dependencies" is a direct reason for schedule deviation, a contributory reason could be a "lack of a dependency management tool". That is, delays caused by "unmanaged dependencies" could have been prevented or reduced by effective dependency management.

For RQ1.3, we applied inductive coding during one integration round to derive a combination of (intervening) factors and their types of relationships to on-time epic delivery. We classified each reported relationship as a 'direct', 'indirect' or 'contributory' relationship using a separate code. Our interpretation of these relationships was based on the explanation of the respondent. For indirect and contributory relationships, we also coded the intervening factors that were mentioned. The first author performed the coding and classification for all answers. The last author did this for 20 percent of the answers. The resulting codes, including the intervening factors that followed from indirect and contributory relationships, matched with codes that were identified from open-ended responses to RQ1.1. No more new codes emerged in this process.

To evaluate inter-coder reliability, the first and last author met in person to compare the types of relationships identified. There were a few borderline cases in which a reported relationship would fit the indirect category as well as the contributory category. In such cases we tried to stay close to the formulation of the respondent. We classified a relationship as

a contributory relationship only if an intervening factor was formulated as a necessary condition for the occurrence of another factor (e.g., using an if-then statement). If it was not phrased as a conditional statement, then we marked the relationship as an indirect one. Using Cohen's kappa [78], we measured substantial agreement between the coders: $\kappa = 0.69$ and percent agreement = 83 percent.

3.1.5   Survey Demographics

As mentioned earlier, the surveys contained a section aimed at gathering demographic information of the respondents, namely, their role within ING, total work experience at ING, total work experience in the software industry. A majority (66 percent) of the respondents self-identified as software engineer, while the rest identified themselves as manager or team lead (19 percent), product owner (7 percent), software architect (6 percent) or other (2 percent). The experience of the respondents at ING ranged from one year (24 percent) to more than 20 years (12 percent) with a median of between one and five years (41 percent). The experience of the respondents in the software industry ranged from one year (4 percent) to more than 20 years (24 percent) with a median of between 10 and 20 years (32 percent).

3.2   Collecting and Analyzing Repository Data

To quantitatively assess the impact of the perceived influential factors presented in Table 1, we extracted proxy measures from multiple data sources at ING that capture the respondents' intended meaning of the factors. In this section, we describe the datasets used and the linking process that we applied to the datasets. The primary goal of our regression model is to explain, rather than predict; we want to understand which proxy variables have a meaningful relationship with schedule deviations in epics. Therefore, we collected proxy variables that can be measured *before* and *after* an epic has been delivered. The mapping of proxy measures to perceived influential factors is shown in Table 3 and will be explained in Section 4.4.

3.2.1   Backlog Management Data

We extracted log data from *ServiceNow*, a backlog management tool used by a majority of teams at ING TECH.[2] The dataset consists of 3,771 epics delivered by 273 teams at ING TECH between January 01, 2017 and December 31, 2019. The dataset contains the following variables for epics: *identification number*, *creation date*, *planned start date*, *actual start date*, *planned delivery date*, *actual delivery date* and a textual *description*. We acknowledge that the planned delivery date of a delivery might change before actual development of the delivery is started. Therefore, we consider only the planned delivery date as scheduled on the day that the development phase is started. An overview of all variables and their descriptions is provided in the supplemental material [72].

3.2.2   Code Quality Measurements

We extracted code quality metrics from *SonarQube*, a static code analysis tool in the software delivery pipeline at ING.[3] It

is being used by most of the development teams at ING TECH. The tool offers a wide range of metrics related to code quality and unit tests execution. We extracted snapshots of SonarQube data of 190 teams that actively use the tool as part of their software delivery pipeline. Each snapshot is linked to a delivery in ServiceNow based on *team ID* and *time stamp* of when the measurement was done in SonarQube. If the time stamp of a SonarQube snapshot falls between the actual start date and actual end date of an epic in ServiceNow, the snapshot is considered to belong to the corresponding epic. Although SonarQube offers a wide range of metrics, we only consider the subset of metrics that are collected by all teams at ING TECH. These metrics allow us to measure and compare the code quality of epics in our analysis set. For each snapshot, we extracted the metrics that teams at ING TECH measure to assess their coding performance:

- *Coding standard violations*: the number of times the source code violates a coding rule.
- *Cyclomatic code complexity*: measured average cyclomatic complexity of all files contained in a snapshot.
- *Branch coverage:* the average coverage by tests of branches in all files contained in a snapshot.
- *Failed test ratio*: the number of failed tests divided by the total number of tests executed during the development phase of an epic.
- *Comment density*: the percentage of comment lines in the source code.

To account for differences in project size, we divided the metrics *Coding standard violations* and *Cyclomatic code complexity* by *Source lines of code*: the total number of lines of source code contained in a snapshot.

3.2.3   Data Cleaning Process

To eliminate noise and missing values, we keep only the epics that meet the following conditions:

1) The *planned delivery date* and *actual delivery date* have been set.
2) The epic has been assigned to a team or group of teams.
3) The *description* field has been set.
4) The epic has been completed (i.e., if its *status* is set to *completed*).

We also removed outliers that exceed two standard deviations from the mean. For example, we removed six epics that had lasted longer than two years and that were, therefore, not representative for the rest of the dataset. The original dataset contained 3,771 epics. After linking and cleaning the data, the final dataset decreased to 2,208 epics from 185 teams for which all data are present. This group of teams overlaps with a majority (68 percent) of the teams that responded to the surveys.

3.2.4   Schedule Deviation Measures

There are a range of error measures used in effort estimation. Most of them are based on the Absolute Error (AE). Mean of Magnitude of Relative Error and Prediction at level l [80] have also been used in effort estimation. However, a number of studies [81], [82], [83] have found that those measures bias towards underestimation and are not stable when comparing effort estimation models. The *Balanced Relative Error* (BRE) [84]

2. https://www.servicenow.com/
3. https://www.sonarqube.org/

has been recommended as an alternative estimation accuracy measure. BRE is defined as

$$\text{If Act - Est} \geq 0, \text{then BRE} = \frac{Act - Est}{\text{Planned duration}}$$

$$\text{If Act - Est} < 0, \text{then BRE} = \frac{Act - Est}{\text{Actual duration}},$$

where Act is the actual delivery date and Est is the planned delivery date of an epic (as reported on the start date of the development phase). $Act - Est$ calculates the difference in days between the actual delivery date and planned delivery date: a positive difference corresponds to underestimation (Act is later than Est), while a negative value corresponds to overestimation (Act is before Est). $Actual\ duration$ is the time interval (in days) between the actual delivery date and start date of the development phase of an epic. $Planned\ duration$ is the time interval (in days) between the planned delivery date and start date of the development phase of an epic. We assess the relative and absolute schedule deviation in epics using BRE and AE (measured in days), respectively.

3.2.5 Regression Analysis

A common approach for measuring the impact of a number of factors on estimation error is to use regression analysis. For RQ2, we used regression analysis to quantitatively assess the impact of combinations of perceived influential factors on the schedule deviation in epics. We extracted 35 proxy measures from backlog management data and code quality measurements that can be mapped to the perceived influential factors. The proxy measures and their mapping to the influential factors are given in Table 3. An analysis of the proxy data revealed that it does not meet the assumptions for linear regression. Considering the need for an interpretable model in our explanatory study, we decided to go for a non-linear, spline-based regression modeling approach. We applied MARS [85]; a multivariate, piecewise regression technique that can be used to model complex relationships between a set of predictors and a dependent variable. We used the proxy measures as predictors and the measured BRE as dependent variable. MARS divides the space of predictors into multiple knots, i.e., the points where the behavior of the modeled function changes. This notion of knots makes MARS particularly suitable for problems with high input dimensions. The optimal MARS model is built using a backwards elimination feature selection routine that looks at reductions in the generalized cross-validation (GCV) criterion as each predictor is added to the model. This procedure makes it possible to rank variables in terms of their contribution to the GCV. We used the default MARS setting provided by the EARTH package in R.

4 RESULTS

This section presents results on factors affecting delays in epic deliveries, derived from survey responses and repository data at ING. Example quotes from the survey are marked with a [rX] notation, in which X refers to the corresponding respondent's identification number. The perceived influential factors resulting from our manual coding process are underlined. The proxy measures that we map to the perceived influential factors are dashed underlined.

4.1 (RQ1.1) Factor Identification

From the open-ended survey responses, we identified 25 factors that are perceived to affect the on-time delivery of epic deliveries. A list of these factors is shown in the left-hand column of Table 1. The factors are organized along the five dimensions identified in the work of Chow et al. [3]; organizational, process, project, people and technical.

4.1.1 Organizational Factors

This category of factors concerns the uncertainty surrounding the organizational environment in which an epic delivery takes place. Many respondents report the importance of organizational alignment for the on-time delivery of epics. A shared vision and mission are essential to ensure alignment between the implementation of an epic and its business strategy: "*A clear management vision creates focus and helps us align on business priorities and timelines across the company.*" [r216]

Another factor that is perceived to contribute to timely delivery is strong executive support. This includes the active involvement of management in strategy execution and the commitment of required resources. Respondent 285 explains that: "*It motivates us if management sufficiently participates in the preparation and performance review of delivery performance*".

In a related manner, respondents report delays related to organizational politics. Bureaucratic structures in the organization can hinder on-time delivery due to side steering: "*Management should trust teams to come up with realistic timelines instead of pushing deadlines. This will prevent last-minute side steering and ad-hoc work.*" [r39] Other factors in this category that are perceived to hamper the on-time delivery of epics are the geographic distribution of teams and a lack of organizational stability (i.e., impact of organizational restructuring).

4.1.2 People Factors

People factors refer to qualities associated with a software development team that can affect the timeliness of deliveries. Factors that are perceived to contribute to the on-time delivery of epics are team stability (i.e., low team member turnover), strong skills and knowledge, team familiarity (i.e., the amount of experience individuals have working with one another) and team commitment to on-time delivery (i.e., motivation to deliver on-time). Teams that are more stable, skilled, familiar and committed to delivering epics on-time are perceived to deliver more often on-time. Moreover, respondents point to the importance of effective communication between teams, management and customers when it comes to technical problems and project delays.

4.1.3 Process Factors

This category of factors refers to the effectiveness and maturity of a software development team's way of working. The

TABLE 1
Overview of 25 Factors That are Perceived to Affect the On-Time Delivery of Epics

Dimension	Factor (RQ1.1)	Perc. resp.	Level of impact (RQ1.2) Distribution 1 2 3 4	Top 2	WA	Order
Organization	Organizational alignment	15%		90%	3.47	#3
	Organizational politics	2%		86%	3.41	#4
	Geographic distribution	2%		83%	3.38	#5
	Executive support	2%		77%	3.18	#14
	Organizational stability	4%		66%	2.91	#20
Process	Requirements refinement	25%		91%	3.55	#1
	Agile maturity	8%		84%	3.34	#7
	Regular delivery	22%		87%	3.32	#8
	Work in progress	6%		75%	3.05	#16
	User involvement	9%		71%	2.95	#19
Project	Task dependencies	16%		92%	3.49	#2
	Project size	9%		84%	3.25	#11
	Project newness	3%		83%	3.22	#13
	Project security	4%		65%	2.83	#22
People	Team stability	4%		85%	3.30	#9
	Skills and knowledge	10%		83%	3.26	#10
	Team familiarity	6%		76%	3.14	#15
	Team commitment	4%		69%	2.97	#18
	Communication	3%		47%	2.49	#25
Technical	Technical dependencies	19%		89%	3.36	#6
	Poor code documentation	3%		82%	3.23	#12
	Unreliable infrastructure	7%		70%	3.03	#17
	Bugs or incidents	6%		68%	2.89	#21
	Lack of code quality	3%		65%	2.82	#23
	Insufficient testing	5%		62%	2.73	#24

The factors are organized along five dimensions: organizational, people, process, technical and project. Percentage of respondents is the percentage of survey respondents that mentioned the corresponding factor. The factors' Level of impact on delays was rated as 4 (large impact), 3 (moderate impact), 2 (small impact) and 1 (no impact). Top 2 is the percentage of respondents that answered 4 or 3 for Impact level. WA is the weighted average of the Likert scale scores for Impact level. The factors' Order numbers are based on the ordering of the weighted averages.

overall top mentioned factor is <u>requirements refinement</u>, which refers to the process of defining epics and dividing them into user stories. Missing or lacking details in the requirements is one of the main reasons for delay: "*Most of the time when we do not make the deadline, the team missed important information during refinement, which surfaced during the sprint.*" [r123] Here respondents also report the importance of frequent <u>user involvement</u> to manage user expectations and avoid delays caused by scope creep.

Another prominent factor featured in this category is <u>regular delivery</u>. Respondents explain the importance of having a short cadence for on-time delivery: teams that regularly deliver production ready software are perceived to be more predictable. Respondent 143 explains that "*Delivering software in shorter cycles enables our team to manage more complex projects and better predict our delivery capacity*".

Some respondents indicate to feel more focused and effective at work when they limit the amount of <u>work in progress</u> at any given time.

Another prominent factor in this category is <u>agile maturity</u>, which stands for the ability of a team to become more agile over time. Respondents explain that they are able to improve their agility, and thereby, on-time delivery, over time through experience. Respondent 163 states that "*It helps to hold Scrum retrospectives and actually following up on their outcome. This allows teams to come up with ways to avoid, mitigate, or better handle impediments and other causes that impact delivery*".

4.1.4 Technical Factors

The technical category represents factors related to the quality of the source code artifact, and the effectiveness of technology and tools used to produce that artifact. Technical factors that are perceived to hamper timely delivery are <u>poor code documentation</u>, <u>lack of code quality</u>, <u>bugs or incidents</u> and <u>insufficient testing</u>. Well-defined coding standards are perceived to make teams more predictable in their deliveries: "*Higher quality standards will result in less incidents and less time spent on code refactoring in the future. This will, in turn, make our team more productive and predictable.*" [r334] Regarding testing, respondents mention to often fall behind schedule when preparing and executing time/resource-intensive types of tests, such as integration tests and performance tests. Such tests can lead to delays due to delayed availability of required infrastructure, unidentified dependencies and late identification of defects.

Another factor that is perceived to delay epic deliveries is a specific type of dependency — <u>technical dependencies</u>. Technical dependencies can occur among different software artifacts, e.g., source-code, architecture, hardware and tools. Teams at ING work with project-specific repositories and share codebases across teams within one application. As a result, teams at ING are hampered by source code dependencies across projects. Moreover, respondents perceive to be delayed by the unavailability or instability of technology and tools used for software development and software testing (<u>unreliable infrastructure</u>).

4.1.5 Project Factors

This category represents the inherent complexity and uncertainty of a software project. <u>Task dependencies</u> constitute a top mentioned factor that is perceived to delay epic deliveries. Task dependencies refer to dependencies among activities in the workflow of collaborating software development teams. Issues such as inconsistent schedules and unaligned priorities can cause delays for teams that collaborate in the same delivery chain: "*Task dependencies occur when teams are not end-to-end responsible for bringing software to production. They create the need for a significant amount of hand-offs.*" [r79]

Our respondents report several project characteristics that can affect on-time delivery: <u>project size</u> (i.e., the size of software, the size of development teams and project duration), degree of <u>project newness</u> (i.e., the innovative nature of a project) and <u>project security</u> (i.e., whether the project needs to go through resource-intensive security tests). Regarding the latter, respondents explain that business-

TABLE 2
Types of Perceived Relationships Between Perceived Influential Factors and the Timeliness of Epic Deliveries (RQ1.3): *Direct* Relationship ↑ , *Indirect* Relationship → , *Contributory* Link * or Mentioned, but *No Explicit Relationship*

Dimension	Factor	Direct ↑			Indirect →	Contributory ★	No explicit relationship
		NR	WT	TE			
Organization	Organizational alignment					Task dependencies, 12%	3%
	Organizational politics	2%					
	Geographic distribution				Communication, 2%		
	Executive support				Team commitment, 2% Team stability, 1%		
	Organizational stability	3%					1%
Process	Requirements refinement	25%			Task dependencies, 20%		
	Regular delivery				Work in progress, 20%		2%
	Agile maturity				Requirements refinement, 7%		1%
	Work in progress			5%			1%
	User involvement				Requirements refinement, 7%		2%
Project	Task dependencies		15%				1%
	Project size				Work in progress, 3% Technical dependencies, 2% Communication, 2% Task dependencies, 1% Insufficient testing, 1% Requirements refinement, 1%		1%
	Project newness			3%			
	Project security					Insufficient testing, 3% Lack of code quality, 1%	1%
People	Team stability				Skills and knowledge, 3% Team commitment, 1%		1%
	Skills and knowledge				Lack of code quality, 6% Bugs or incidents, 4%		1%
	Team familiarity				Communication, 4%		2%
	Team commitment			3%			1%
	Communication			1%		Task dependencies, 2%	
Technical	Technical dependencies	13%			Lack of code quality, 6%		
	Poor code documentation			3%			
	Unreliable infrastructure		6%				1%
	Bugs or incidents	5%					1%
	Lack of code quality	3%					
	Insufficient testing	4%			Bugs or incidents, 1%		

Factors can have three types of direct relationships with on-time delivery; they can lead to necessary rework (NR), unplanned waiting time (WT) or deviations in team effectiveness (TE). For indirect and contributory links, intervening factors are mentioned in the table. The percentages indicate the percentages of survey responses that mentioned the corresponding relationship.

and safety-critical applications are generally built and tested to much higher security standards, which may lead to unexpected delays in the quality assurance and security testing process.

4.2 (RQ1.2) Perceived Level of Impact

For each factor, respondents were asked to rate the level of impact using Likert-type choices of "no impact", "small impact", "moderate impact" and "large impact". The right-hand column in Table 1 shows the perceived level of impact of the factors as rated by the survey respondents. The *Top 2* percentage indicates the percentage of responses that rated the factor as having "large impact" or "moderate impact". The *Order* represents the order of factors by the weighted average of their impact level scores. Close to 60 percent of the respondents felt the factors are all moderately influencing their on-time delivery. Task dependencies, requirements refinement, organizational alignment, technical dependencies

and regular delivery are the top 5 cited factors. They received more than 86 percent responses in the large and moderate impact category. Based on the weighted average of impact scores, requirements refinement (order #1), task dependencies (order #2), organizational alignment (order #3), organizational politics (order #4) and geographic distribution (order #5) are the top 5 rated factors. They received a weighted average impact score of 3.38 or higher. The impacts of the top 15 rated factors are perceived to have large or moderate impact by over 76 percent of the respondents. Communication (order #25) has lowest perceived impact but 47 percent of the respondents still rated it to have large or moderate impact.

Further analysis shows that respondents were quite consistent in their high ratings of most factors. The ratings for all factors have a standard deviation (SD) lower than 0.80, except for unreliable infrastructure (SD = 1.01), project security (SD = 0.99), lack of code quality (SD = 0.97), insufficient testing (SD = 0.96) and team commitment (SD = 0.96).

4.3 (RQ1.3) Perceived Types of Factor Relationships

We investigated open-ended survey responses to identify direct, indirect and contributory relationships between 25 perceived influential factors (from Table 1) and on-time epic delivery. As explained in Section 3.1.4, we focused on three types of relationships to distinguish between simple, complex and condition-dependent types of interactions between factors. An overview of the perceived types of relationships between factors and on-time epic delivery is shown in Table 2. Respondents mentioned three ways in which factors can have a direct impact on the timeliness of epics; factors can lead to unplanned *waiting time (WT)*, *necessary rework (NR)* or changes in *team effectiveness (TE)*. These are orthogonal types of effects. WT and NR are described to lead to delays, while increased (perceived) TE (i.e., a team's capacity to achieve its goals and objectives) is reported to help with on-time delivery. Table 2 shows the relevant type (NR, WT or TE) for each perceived direct relationship.

We found that the *organizational factors*, *people factors* and *technical factors* are perceived to interact *hierarchically*. The organizational factors have an impact on the people factors, which in turn affect the technical factors. From the organizational factors, executive support is perceived to have an indirect impact on timely delivery through team stability and team commitment. Respondents believe that strong executive support leads to more stable and highly motivated teams. Geographic distribution is associated with communication challenges and, thereby, reduced team effectiveness and delay. From the people factors, team stability is observed to be positively related to skills and knowledge and team commitment. Respondents explain that stable teams are more likely to develop competences over time and to take ownership of their work. Team commitment and communication are perceived to increase team effectiveness, and thereby, help with timely delivery. Skills and knowledge is reported to have an indirect impact through bugs or incidents and lack of code quality. Respondents point out that teams with more experienced members are faster at finding faults in software, and resolving unforeseen bugs and incidents. From the technical category, all factors are perceived to have a direct impact on on-time epic delivery. Respondents explain that problems related to technical dependencies, lack of code quality, bugs or incidents and insufficient testing can introduce necessary rework. Technical dependencies are also perceived to have an indirect impact on on-time delivery through *lack of code quality*. Respondents indicate that technical dependencies can introduce dependency problems, resulting in more complex and less maintainable source code.

In general, the *process factors* are not perceived to have relationships with factors from other categories. One exception is the link between requirements refinement and task dependencies: respondents believe that an effective refinement process should reveal task dependencies, thereby enabling teams to minimize the likelihood of delays caused by dependencies. Moreover, the most often reported relationship is the direct impact of requirements refinement on on-time epic delivery. Respondents report that refinement can prevent rework that is caused by unclear requirements. Regular delivery is perceived to lead to more consistency in

the amount of work in progress, which in turn has a positive impact on team effectiveness and timely delivery.

The *project factors* are reported to have direct, indirect and contributory relationships with on-time epic delivery. Unresolved *task dependencies* are perceived to immediately result in delays through hand-offs and waiting times between dependent teams. *Organizational alignment* and *communication* are perceived to contribute negatively to the relationship between task dependencies and on-time epic delivery. Respondents explain that task dependencies can only stay unresolved and lead to delays in environments characterized by misaligned priorities and communication issues. Project size is perceived to be linked with factors across all dimensions. Larger epics are associated with more dependencies, parallel work, communication overhead, and increased testing and refinement effort. Innovative epics (project newness) are reported to involve time-intensive exploration activities and to be hampered by unforeseen obstacles, which can decrease team effectiveness and lead to delays. A higher level of project security is perceived to contribute positively to delays due to rework introduced by insufficient testing and lack of code quality.

> *Key findings from RQ1:* We identified 25 factors that are perceived to affect the timeliness of epic deliveries. Requirements refinement, task dependencies, organizational alignment, organizational politics and the geographic distribution of teams are perceived to have the greatest impact on on-time epic delivery. The factors interact hierarchically; the organizational factors interact with people factors, which in turn impact the technical factors. The technical factors are perceived to have a direct impact on timely epic delivery.

4.4 Studied Proxy Measures

We formulate 35 proxy measures to map to the perceived influential factors. We extract the proxy measures from the datasets described in Section 3.2 to fit a regression model to quantitatively assess the impact of the perceived influential factors. Table 3 provides definitions for the proxy measures and their mapping to the perceived factors. Some of the proxy measures are self-descriptive and for others we explain the measure below. The proxies of the technical factors are described in Section 3.2.2. We take the median across teams or individual team members to produce measures that are representative of the group as a whole. For individual data, such as developer age, we take the median of team members working on an epic. For team data, such as team stability, we take the median of teams working on an epic.

We were not able to measure proxies for organizational alignment, organizational politics, executive support, (quality of) communication and technical dependencies. ING is not collecting quantitative data on these factors. It was also not possible to collect such data ex post facto.

4.4.1 Motivation for Mapping

Organizational Factors. To quantify the global distance between teams, we calculate global-distance based on the

TABLE 3
Mapping of Proxy Measures to the Perceived Influential Factors (From Table 1)

Perceived factor	Proxy measure	Description
Task dependencies	out-degree	Number of outgoing dependencies of an epic on other epics
Geographic distribution	global-distance	The maximum *Global Distance Metric* [86] (combining geographical, temporal, cultural distance) measured across teams working on an epic
Organizational stability	nr-changed-leads	Number of changed tribe leads during the current epic and previous epic
Team stability	stability-ratio	Median of the ratio of team members that did not change during the current epic and previous epic
Skills and knowledge	dev-age-team	Median of the number of years the developers working on an epic have been part of their team
	dev-age-ing	Median of the number of years the developers working on an epic have been working at ING
	dev-seniority	Number of senior developers working on an epic
Team familiarity	team-existence	Median of the number of years teams have existed in their current composition of team members
Team commitment	hist-performance	Median of the ratio of on-time delivered epics over all teams working on an epic
Requirements refinement	nr-updates	Number of times the epic's description field was updated before the start of the development phase
	state-ready	Whether refinement of an epic was completed before the start of its planning (binary)
Regular delivery	cycle-time	Sprint duration of teams working on an epic
	deviation	Median relative standard deviation from the regular cycle time during an epic
Agile maturity	team-point	Median of the total number of story points delivered by teams so far
	velocity	Median of the number of delivered story points per sprint during the current epic
Work in progress	parallel-epics	Median of the number of other epics that teams worked on simultaneously
	dev-workload-stories	Median of the number of stories assigned to a developer per sprint
	dev-workload-points	Median of the number of story points assigned to a developer per sprint
User involvement	acc-criteria	Whether acceptance criteria are specified for an epic (binary)
Poor code documentation	comment-density	The *comment density* measured at the start of an epic
Unreliable infrastructure	env-incidents	Number of environmental incidents that occurred during the development phase of an epic
Bugs or incidents	nr-incidents	Number of incidents that occurred during the development phase of an epic
	nr-unplanned-stories	Number of unplanned stories related to bug fixes or incidents that were added after the start of an epic's development phase
Lack of code quality	coding-violations	Number of *coding standard violations* measured at the start of an epic
	code-complexity	The *cyclomatic code complexity* measured at the start of an epic
Insufficient testing	branch-coverage	The *branch coverage* measured at the start of an epic
	failed-test-ratio	Number of tests that failed during the development phase of an epic
Project size	nr-stories	Number of planned stories assigned to an epic
	nr-sprints	Number of sprints assigned to an epic
	nr-points	Total number of planned story points assigned to an epic
	nr-teams	Number of teams working on an epic
	team-size	Median team size
	loc	Total *source lines of code* measured at the start of an epic
Project newness	novelty	Whether an epic contributes to the company's business transformation ambition (binary)
Project security	security-level	Whether an epic needs to pass a resource-intensive security testing process (binary)
Organizational politics		
Organizational alignment		
Executive support		No data available
Communication		
Technical dependencies		

The Description *column provides a description of the proxy measure.*

Global Distance Metric proposed in related work [86]. We calculate the metric for pair-wise combinations of teams and take the maximum value. To assess organizational stability, we calculate <u>nr-changed-leads</u> ; this has been shown to influence the success of software projects in earlier work [57].

Process Factors. To assess the refinement quality of an epic, we calculate <u>state-ready</u> to determine whether the 'status' field of an epic was set to 'refinement ready' before the start of its planning phase. The intuition here is that epics that are not clearly defined will miss important information that is needed for planning and that could potentially result in delay. We also calculate <u>nr-updates</u> ; the intuition here is that problematic epics will raise more

comments and questions that could result in more updates and potentially a delay. To assess agile maturity, we calculate <u>velocity</u> and <u>team-point</u> ; both have been shown to be representative of maturity in related work [10], [87]. We extract <u>acc-criteria</u> to measure user involvement in the definition of an epic. The backlog management data contains a special 'acceptance criteria' field that indicates whether teams consulted their customer(s) to define acceptance criteria.

Project Factors. We quantify task dependencies as the outgoing degree (<u>out-degree</u>) of dependencies of an epic; this has been shown to predict delay in related work [67]. To assess project newness, we determine the <u>novelty</u> of an

epic based on a special 'business goal' field in backlog management data. This field indicates whether an epic contributes to a 'business transformation', 'business continuity' or 'compliance-related obligation'. Epics that contribute to a business transformation are marked as novel. To measure project security, we determine security-level , which indicates whether an epics needs to go through a mandatory, resource-intensive security testing procedure.

People Factors. To calculate the number of senior developers working on an epic (dev seniority), we retrieve the expertise levels of developers as specified in backlog management data. At ING, the five-stage Dreyfus model [88] is employed to evaluate the expertise of developers based on their experience in the software industry. We marked developers with a Dreyfus skill level of 3 ('competent') and higher as senior. To assess a team's commitment to on-time delivery, we calculate hist-performance; team commitment has been shown to be positively related to team performance and estimation accuracy in related work [64], [89], [90], [91].

4.5 (RQ2) Factor Validation

How do perceived influential factors impact schedule deviation in epic deliveries?

To answer this research question, we applied MARS to quantitatively assess the impact of combinations of proxy measures on the BRE values in epic deliveries. Table 4 presents the optimal MARS model for BRE values based on all proxy measures presented in Table 3. The columns show, from left to right, the beta factor coefficients β_m denoted as BF_m, the basis functions selected as significant covariates in the model, the coefficient values estimated and a visualization of the relationship between the independent variable and BRE value. The value of the beta factor implies the magnitude of effect of the basis function (i.e., variable effect) on the BRE value. For the effect of each basis function, $max(0, x - t)$ is equal to $(x - t)$ when x is greater than t (the knot value); otherwise the basis function is equal to zero.

As shown in Table 4, the MARS model contains 20 basis functions and 13 proxy measures. The selected proxy measures represent all five of the factor dimensions. They are effective in explaining 67 percent of the variation in the BRE values of epic deliveries (adjusted R^2: 0.672.). Fig. 3 provides a ranking of the proxy measures by order of importance. Proxies that have no impact on BRE are not shown. The importance is calculated as the relative importance of proxies in terms of reductions in the GCV estimate of the prediction error as each proxy measure is included. From this figure, we observe that nr-sprints, out-degree, hist-performance, dev-age-ing, team-existence and team-size have the greatest impact on schedule deviation in epics. Their importance values range from 15 to 21 percent.

Factors associated with delay show a rising relationship in the rightmost column of Table 4. From Table 4, beta factors BF1 and BF2 account for the nonlinear delay effect of nr-sprints, the most important variable in the MARS model. The number of sprints in an epic is positively related to the BRE values with a knot at $t = 8$. The effect of nr-sprints can

TABLE 4
The MARS Model for the BRE Values of Epic Deliveries (RQ2)

Beta factor	Basis function	Coefficient	Relationship
BF0		0.1014	
BF1	Max(0, 8 - *nr-sprints*)	−0.0142	
BF2	Max(0, *nr-sprints* - 8)	0.0339	
BF3	Max(0, 4 - *out-degree*)	−0.0138	
BF4	Max(0, *out-degree* - 4)	0.04507	
BF5	Max(0, 0.61 - *hist-performance*)	0.1843	
BF6	Max(0, *hist-performance* - 0.61)	−0.2643	
BF7	Max(0, 2.66 - *dev-age-ing*)	0.0984	
BF8	Max(0, 0.93 - *team-existence*)	0.0242	
BF9	Max(0, *team-existence* - 0.93)	−0.0099	
BF10	Max(0, 5 - *team-size*)	−0.0066	
BF11	Max(0, *team-size* - 5)	0.0248	
BF12	Max(0, *security-level* - 0.85)	0.1288	
BF13	Max(0, 8 - *nr-unplanned-stories*)	−0.0043	
BF14	Max(0, *nr-unplanned-stories* - 8)	0.0138	
BF15	Max(0, *changed-leads* - 2)	0.0344	
BF16	Max(0, 0.74 - *stability-ratio*)	0.3091	
BF17	Max(0, 19 - *nr-stories*)	−0.0038	
BF18	Max(0, *nr-stories* - 19)	0.0105	
BF19	Max(0, *nr-incidents* - 10)	0.0145	
BF20	Max(0, *dev-workload-points* - 12)	0.0129	

Adjusted R^2: 0.672. The variables in the models are ordered by importance (see Fig. 3).

be explained as follows. The delays in epics tend to increase with a higher number of sprints. If the number of sprints in an epic is lower than 8, the schedule deviation will increase by 0.0142 per sprint (indicated by BF1). If the number of sprints exceeds 8, then the schedule deviation value will increase faster by 0.0339 per sprint (indicated by BF2). Other proxy measures that contribute to delay in epics are out-degree, team-size, nr-unplanned-stories and nr-stories. They have a two-sided, positive relationship with BRE with a single knot. This indicates that delays in epics tend to increase with larger teams and with more outgoing dependencies, unplanned stories and planned stories. The proxies security-level, nr-changed-leads, nr-incidents and dev-workload-points can also contribute to delays: they have a right-sided, positive relationship with BRE with a single knot. This means that when these proxies exceed their corresponding knot value, the delay in epics tends to increase. For example, the beta factor BF15 shows the nonlinear effect of nr-changed-leads on BRE which can be described as follows. If the teams' tribe lead changed less than two times during the current and previous epics, it has a negligible effect on the BRE value (indicated by BF0). However, if the number of changed tribe leads exceeds two,
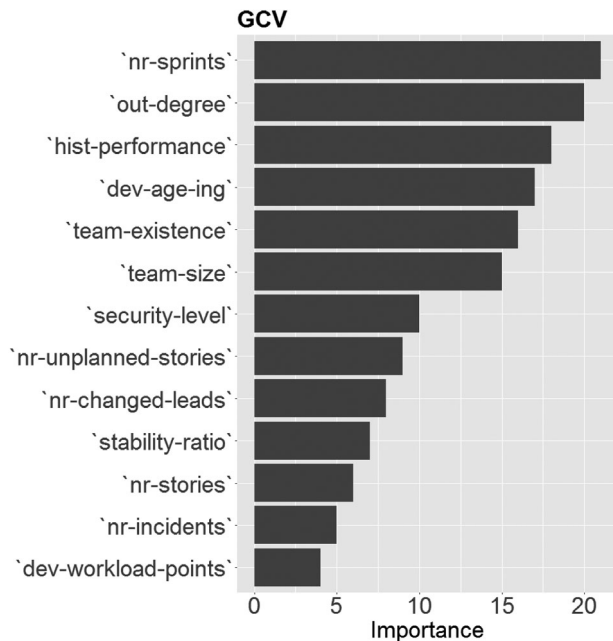
Fig. 3. Importance of proxy measures in MARS model for BRE.

then the BRE value will increase by 0.0344 for every additional change in tribe lead. The effects of security-level are similar. If the security level is higher than 0.85, then the BRE increases by 0.1288. Given that security-level is a binary variable, this means that in practice the BRE values of epics that need to pass the mandatory security testing procedure are 0.1288 higher than the BRE values of epics that do not need to go through this procedure.

Factors associated with on-time delivery show a downward relationship in the rightmost column of Table 4. As indicated by BF5-9 and BF16, the proxies (hist-performance, dev-age-ing, team-existence and stability-ratio help with the timely delivery of epics. The delay in epics tends to decrease for teams that were less often involved with delay epics in the past and teams that have existed longer in their current composition. The delay can also decrease with a higher developer experience at ING and higher team stability up to the corresponding knot values.

Absolute Deviation. Further analysis of the absolute deviations in epics showed that an overlapping set of 10 variables is effective in explaining 61 percent of the variation in the AE values. The proxy measure nr-teams emerged as statistically significant, while nr-unplanned-stories, nr-changed-leads, nr-incidents and dev-workload-points were not selected in the model for AE values. Moreover, the proxies have slightly different importance values: out-degree (15), nr-sprints (14), (team-existence (13), dev-age-ing (12), hist-performance (11) and (nr-stories (10) have the greatest explanatory power for the absolute deviation.

> *Key findings from RQ2:* A set of 13 proxy measures is effective in explaining 67 percent of the variation in the BRE values of epics. The project size, number of task dependencies, historical delivery performance, team familiarity and developer experience at ING have the greatest explanatory power for schedule deviations in epics.

5 A CONCEPTUAL FRAMEWORK OF ON-TIME DELIVERY

We organized the findings from our survey and regression analysis into a conceptual framework, presented in Fig. 4. The framework captures the 25 perceived influential factors (from Table 1) and how they relate to on-time delivery. The connections between factors are derived from the reported types of relationships in Table 2. The directions are derived from the descriptions of factors in Section 4.1 and the relationships in Section 4.3.

Practitioners can use our conceptual framework to identify and manage the risks associated with on-time software delivery. For example, in our regression model, historical delivery performance is one of the most important proxy measures that affect schedule deviations in epics. Epics assigned to teams having a high percentage of delayed epics are at a high risk of being delayed. We recommend project managers to identify the teams that are involved with many delayed epics in the past, and consider allocating more time for their work or training them to successfully estimate effort and deliver on-time.

Our study also provides practitioners with a comprehensive list of factors and proxy measures that should be collected and analyzed to derive useful models that can be applied to improve on-time delivery. An assessment of the most significant factors would be a good starting point for further exploring influential factors in other settings. By weighing and analyzing the qualitative and quantitative attributes of factors, practitioners can choose the most important factors that influence their on-time delivery. Moreover, our analysis shows that expert- and data-based selection methods identified different (only partially overlapping) sets of relevant factors. Therefore, we recommend software practitioners to combine selection methods to extract more detailed insights and gain a better understanding of their software development processes.

The design of an effective strategy to improve on-time delivery must recognize the relationships between influential factors. The relationships in our conceptual framework are operationalized by reported relationships. We can therefore not reason about causal links between factors. However, our framework does enable us to form hypotheses that could lead to actionable insights and may suggest corrective actions to address the root causes of delay in similar development contexts. Our results suggest that addressing factors that have a direct relationship with on-time delivery should directly lead to an improvement in on-time delivery. For example, our survey respondents believe that a lack of code quality leads to necessary rework, and thereby, delay in epics. We therefore hypothesize that code quality improvements may reduce the likelihood of rework and improve the on-time delivery performance. Moreover, our results suggest that improvements in indirectly influential factors lead to improvements in intervening factors, which, in turn, improve the timeliness of deliveries. For example, executive support is perceived to have a positive impact on team stability and team commitment, which in turn leads to improved team skills, high-quality code and less bugs or incidents. We hypothesize that establishing stronger executive support may lead to more stable, committed and highly
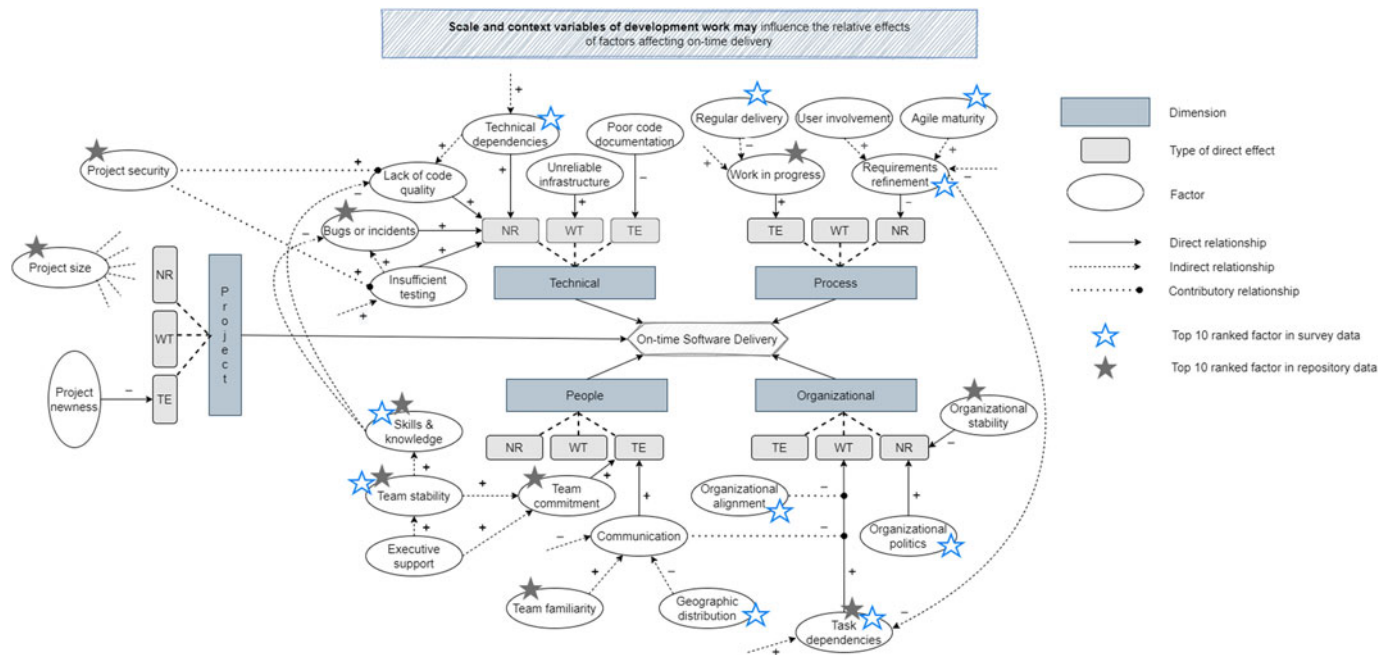
Fig. 4. A conceptual framework of on-time delivery (based on survey responses and quantitative data from the case company). The dimensions/categories of factors are shown in blue rectangles. The 25 perceived influential factors are visualized as ellipses connected by directional arrows indicating their perceived relationships. For readability purposes, the arrows starting from project size are partially omitted. For each relationship, the direction (*positive +* or *negative -*) is shown. Factors are found to have three types of direct relationships with the timeliness of deliveries; they can lead to *necessary rework (NR)*, unplanned *waiting time (WT)* or changes in *team effectiveness (TE)*. The star symbols indicate factors that are rated among the top 10 most relevant factors in survey data or repository data.

skilled teams that are better able to maintain code quality and resolve delays caused by bugs and incidents. The paths for action that can be inferred from our conceptual framework indicate that on-time software delivery requires attention across many factors, and that both social and technical factors may need to be addressed to enable continuous improvement.

6 DISCUSSION

New Influential Factors. Our study has identified additional factors that influence on-time delivery and which, to the best of our knowledge, have not been covered in the current literature. We found that team familiarity is associated with timely software delivery. While prior research [92] has shown that familiarity is beneficial to team performance, it has not been investigated in the context of on-time delivery before. Our respondents believe that familiarity between team members improves the team coordination and helps in adapting to environmental changes. This indicates that for a better on-time delivery performance project managers should not only focus on keeping teams stable, they should also track and support teams to build familiarity over the long term. Moreover, agile maturity emerged as a new factor that is perceived to affect on-time delivery. The survey respondents rated this factor among the top 10 most influential factors. The survey responses point out that a growing agile maturity enables teams and ultimately the organization to continuously improve their on-time delivery performance.

Relevance of Factors. The 25 influential factors we presented as part of RQ1 relate to previous research in effort estimation [20] and software project risk management [65].

Our results provide new insights into the relative effects of these factors. Our respondents perceive requirements refinement, task dependencies and organizational alignment to have the greatest impact on the timing of their deliveries. While requirements-related issues are top-cited risk factors in literature [65], task dependencies and organizational alignment have not received much attention. They have only been investigated in the context of scaling agile methods [93]. Further research is required to investigate the importance of these factors in the context of on-time delivery.

Prior work [12] has shown that team- and project-related factors are the most often mentioned effort drivers by agile practitioners. However, the relative importance of these factors has not been investigated before. The regression analysis we presented as part of RQ2 confirmed that the proxy measures of project factors (i.e., task dependencies, project size) and team factors (i.e., team commitment, team familiarity and skills & knowledge) have the greatest impact on schedule deviations. The project factors are found to have a slightly greater impact than task factors. We did not find a significant relationship between the code quality measurements and schedule deviations in epics. Poor code quality and documentation have also emerged as perceived influential factors in other studies [94], [95]. More in-depth studies are needed to corroborate the perceived impact of source code quality on on-time delivery.

Role of Organizational Environment. Our respondents reported the importance of several organizational factors for on-time delivery. The factors organizational alignment, organizational politics and geographic distribution were rated among the top 5 most influential factors in our survey. Among these, organizational politics and organizational

alignment have been shown to impact project performance in related work [57], [64], [96] but they have not been identified as top influential factors before. Hence, our results suggest that different environmental aspects may play a larger role in on-time delivery than previously thought. Further research is required to investigate the impact of the organizational environment on on-time delivery in different settings.

Coordination Challenges in Large-Scale Agile. Our survey respondents indicated that several factors affecting on-time delivery are related to the challenges of adopting agile methods at the large scale of the case company. These factors include task dependencies, technical dependencies, geographic distribution and organizational alignment. Our respondents explained that their teams often depend on other teams and external parties for testing and deploying new software. This resonates with the findings by earlier work [93], [97], [98] that large-scale agile projects are more likely to be hampered by communication and coordination challenges. Further research is required to investigate the characteristics and impacts of different types of dependencies and task relationships in large-scale settings. Software organizations would benefit from mechanisms that make teams aware of inter-team dependencies, blockers and external dependencies that have the largest impact on their delivery time. Future research should study how existing coordination mechanisms are used in large-scale agile companies and how they can be improved to better support agile teams in delivering on time.

Incident Management Workflows. In line with earlier work [60], [99], [100], [101], [102], we found that software deliveries are delayed by unexpected bugs and technical incidents. In our regression model, the number of unplanned stories and the number of incidents have a strong relationship with schedule deviation in epics. The disruptive nature of bugs and incidents calls for streamlined incident management processes and automated incident handling. Promising research in this direction has been carried out by Gupta *et al.* [103]. They used information integration techniques and machine learning to automatically link incoming incidents with configuration items. An interesting extension would be to leverage probabilistic modeling to predict the impact of an incoming incident on the time estimate of a delivery.

Multi-Objective Optimization for Software Delivery. We found that the security level of a software delivery is positively related to delay. Our respondents indicated that there is no tolerance for failure in some of the business-critical systems at ING. In highly regulated projects, engineers may need to decide to delay a delivery to increase time available for quality assurance and security testing. This alludes to a tension between delivery speed and the constraints imposed by regulations. New methods for rapid security verification and vulnerability identification could help organizations maintain agility. Related work [104], [105], [106] has focused on integrating security into agile methods and the challenges which this presents. Fitzgerald *et al.* [104] looked into the concept of *continuous compliance* and end-to-end traceability to support agile development processes in large-scale regulated environments.

The trade-off between timely delivery and security highlights a broader theme that predictable delivery is not the only factor that development teams in software organizations

are trying to optimize. In reality, organizations deal with multiple objectives and look for optimal trade-off solutions that balance several criteria. Future research should investigate how the value of on-time delivery is measured and weighed in trade-offs in software industry.

A Sociotechnical Approach to Improving On-Time Delivery. The perceived indirect and contributory relationships between influential factors (see Table 2) show that changing one factor may impact another, and that there are many non-technical factors that may also have an impact on on-time delivery. The results lead to some hypotheses that we discussed in Section 5, that may suggest corrective actions to improve the timeliness of software deliveries. The hierarchical interactions between factors indicate that the interplay between technical factors and on-time delivery is influenced by the social context of development work, as determined by organizational and people factors. This suggests that a healthy team culture and organizational environment can be used as intervening factors to resolve potential harmful effects of technical issues on on-time delivery.

Predicting Delays in Software Deliveries. An interesting opportunity for future work is to incorporate the influential factors from our study into predictive models for delays in software deliveries. Currently, most software organizations rely on experts' subjective assessment to arrive at a time estimate for their deliveries. This may lead to inaccuracy and more importantly inconsistencies between estimates. Therefore, software organizations can benefit from predictive models that provide automated support for project managers in predicting the delivery time or delay of software deliveries. Existing models [41], [67], [67], [107] learn based on metadata (e.g., type, priority) and/or textual features of the software task. Our results show that the predictive power of these models can be enhanced by incorporating the significant variables from our regression analysis. This will enable models to capture information about the software task as well as the environment in which the delivery takes place.

Our findings also suggest that an incremental learning approach might be beneficial for predicting delays in software deliveries. As unexpected events related to bugs, incidents, and a team members' departure can occur during the development phase (after a time estimate has been made), the prediction model should learn over time and adjust predictions based on newly acquired knowledge. A sliding window-setting could be used to model team dynamics and external changes. This boosts the ability of the model to learn and predict based on a team's recent delivery performance, and to forget older, irrelevant data. Initial work in this direction has been carried out by Abrahamsson *et al.* [34].

A Relational Theory of On-Time Delivery. A relational theory of on-time delivery would provide insightful and actionable information for project managers to design effective risk management strategies that improve on-time delivery performance. Our framework contributes in the advancement of a *relational theory* of on-time software delivery [108]. We identified and categorized influential factors (*descriptive theory*), and aim to specify the relations between factors to start to explain how they interact with each other. One potential research direction is therefore to investigate

why factors are related, e.g., through causality. This could be assessed by causal inference on time-series data, or through observations collected by experiments with development teams.

Guiding Future Research. We anticipate that our conceptual framework, and the set of factors we identified, can be useful for other researchers that study on-time software delivery. Our survey instrument and mixed-methods approach can be replicated to reveal how specific factors impact on-time delivery in other settings. A consideration across organizations of different scale and context could lead to quite different factors that influence the results. Moreover, other social contextual settings, such as organizations with different levels of management support and organizational stability, could be considered to explore how the social setting affects the interplay between technical and social factors, and their impact on on-time epic delivery.

7   THREATS TO VALIDITY

In this section, we discuss the threats to validity of our study and limitations with our conceptual framework. We used the checklist of Molléri *et al.* [109] to assess our surveys and identify threats to validity. The resulting scores from our assessment using the checklist can be found in the supplemental material [72].

External Validity. As with any single-case empirical study, external threats are concerned with our ability to generalize our results [110]. The company we studied employs thousands of software engineers and has significant variety in terms of the products developed, the size and application domain (banking applications, cloud software, software tools). We performed random sampling and captured a range of roles and experiences, which may improve generalizability [109]. Even though our sample of survey participants is diverse, it is unlikely to be representative of software managers and engineers in general.

We conducted self-administered surveys, which may suffer from non-response bias [111], [112]. Our surveys were advertised as an "On-time Software Delivery Survey" and therefore could have led to over-representation of teams that deliver on-time. Developers from teams that are often delayed might have been less comfortable about participating in the surveys. Moreover, some of the factors presented in the second survey were geared at technical aspects of software development. This might have caused participants in non-technical roles to drop out, resulting in a bias toward software engineers. A limitation from our survey tool is that it does not record partially completed surveys. Hence, we do not know the dropout rate and cannot determine drop-out questions. This introduces a possible threat to external validity [109]. Our survey may have also been subject to self-selection bias, e.g., participants with strong opinions about delivery deadlines might have been more likely to participate in the survey. To mitigate non-response and self-selection bias, we sent personal invitations, kept the survey as short as possible and were transparent about the survey length. We also sent reminders to non-responders to increase the response rate and reduce the possibility of bias.

Although we control for variations using a large number of participants and projects, we cannot generalize our conclusions to other organizations. Replication of this work in different development settings is required to determine how work context influences (the perceptions of) factors affecting on-time delivery. Our findings indicate a trade-off between timely delivery and increased (security) testing and code refactoring effort. In a financial organization like ING there is no tolerance for failure in some of their business-critical systems. This may have influenced the factors we identified, making our findings likely to generalize more to software organizations with similar security regulations. Moreover, our results show several factors related to organizational fragmentation (e.g., dependencies, organizational alignment) which may be more common in large-scale organizations. In an effort to increase external validity and encourage replication, we have made our survey instrument available so that others can deploy it in different organizations and contexts [72].

Internal Validity. We recognize that surveys can introduce biases and may contain ambiguous questions [109]. To mitigate these issues, we used terminology familiar to the target population and piloted the survey. We updated the survey instruments based on the validation results. In addition, we sought support from the second (confirmatory) survey and performed data triangulation. In the second survey (also piloted), no new factors emerged from the open-ended responses we solicited. In our survey design, we phrased and ordered the questions sequentially to avoid leading questions. We also randomized the order of factors to address order effects.

Our survey was not anonymous and therefore might have been subject to social desirability bias (i.e., a respondent's possible tendency to appear in a positive light) [113]. To mitigate this risk, we let participants know that the responses would be kept confidential and evaluated in aggregated form.

A factor that might have influenced our qualitative analysis is the bias induced by the involvement of the authors with the studied organization [109]. To counter the biases which might have been introduced by the first two authors, the last author (from Delft University of Technology) helped in designing survey questions and the manual coding of survey responses. In addition, we formally checked for reliability by computing inter-coder reliability. Another risk of the coding process is the loss of accuracy of the original response due to an increased level of categorization. To mitigate this risk, we allowed multiple codes to be assigned to the same answer.

Construct Validity. The goal of our first survey was to elicit influential factors and their interactions as experienced by engineers themselves. As the factors and types of relationships come from open-ended responses, and we rigorously assessed the expressed mechanisms through manual coding, we argue that they accurately represent the respondents' views. However, it should be noted that we did not verify whether participants can distinguish between affects. In the second survey we asked participants about the perceived impact of factors. To keep the survey short, each factor was measured by a single response item. Therefore, we could not test the reliability of participants' responses.

In our analysis of repository data, we consider data variables as constructs to meaningfully measure perceived influential factors. This introduces possible threats to construct validity due to measurement errors [114]. The proxy variables we measured may not capture the respondents' intended meaning of the concepts or constructs. Many factors, such as team commitment and organizational alignment, are quantifiable in principle but not directly measurable. For example, we measured the historical on-time delivery performance of a team as a reflective indicator of their commitment to on-time delivery. However, the commitment level of team members might not be reflected in their past on-time delivery performance. A more common and direct way of measuring commitment is through psychological attachment instruments but it was not possible to collect such data ex post facto.

For the mapping of proxy measures to the perceived influential factors we had to find acceptable trade-offs between the preciseness of proxy measures and the availability of data at ING. We acknowledge that for some perceived factors more precise alternatives can be found in related work. However, the repository data available did not cover equally precise data on all factors. Furthermore, it is possible that the data variables do not accurately represent reality. For example, we calculated time-related information on epics based on their planned and actual delivery dates in backlog management data. However, it might happen that teams close their deliveries too early or too late. We cannot account for the impact of poor record keeping on our results.

Transferability and Credibility of Our Framework. We believe that the influential factors in the conceptual framework are likely transferable, to some extent, to other settings as they relate to previous research in effort estimation and project risk management. However, the specific results regarding the ordering of factors, factor relationships and regression analysis are bounded to the scale and context of ING. How factors impact on-time software delivery may vary according to scale and context factors of development work. It is noteworthy that we were not able to validate all factors as the repository data available did not cover all perceived influential factors. Our regression analysis does not exclude the importance of the non-included variables. Additional variables would probably have been included in our regression model if we had more data. We were not able to triangulate the relationships for RQ1.3. Replication of this work is required to validate the findings and reach more general conclusions. This might help enrich the conceptual framework.

8 CONCLUSION

Improving the timeliness of software deliveries is a challenge that is faced by many software organizations. In this paper, we identified and investigated the most relevant factors affecting delay in large-scale agile software development. We composed our findings in the form of a conceptual framework (Fig. 4) representing these factors and their interactions. The key findings of this study are:

1) Requirements refinement, task dependencies, organizational alignment, organizational politics and the geographic distribution of teams are perceived to have the greatest impact on timely software delivery.
2) Project size, number of dependencies, historic delivery performance, team familiarity and developer experience are the most important variables that explain schedule deviations in software deliveries.
3) Factors are found to interact hierarchically: organizational factors are perceived to interact with people factors, which in turn impact the technical factors. Technical factors are perceived to have a direct impact on timely software delivery.

Our conceptual framework suggests multiple paths for action that may improve the timeliness of software deliveries. Based on our findings, we identified challenging areas calling for further attention, related to the scalability of agile methods, inter-team dependencies, security concerns, the role of organizational culture, team stability and incident management. Progress in these areas is crucial in order to become more predictable at delivering software in agile settings.

ACKNOWLEDGMENTS

REFERENCES

[1] T. Halkjelsvik and M. Jørgensen, "From origami to software development: A review of studies on judgment-based predictions of performance time," *Psychol. Bull.*, vol. 138, no. 2, 2012, Art. no. 238.
[2] M. Jørgensen, "What we do and don't know about software development effort estimation," *IEEE Softw.*, vol. 31, no. 2, pp. 37–40, Mar./Apr. 2014.
[3] T. Chow and D.-B. Cao, "A survey study of critical success factors in agile software projects," *J. Syst. Softw.*, vol. 81, no. 6, pp. 961–971, 2008.
[4] F. J. Heemstra, "Software cost estimation," *Inf. Softw. Technol.*, vol. 34, no. 10, pp. 627–639, 1992.
[5] M. Jørgensen, "A review of studies on expert estimation of software development effort," *J. Syst. Softw.*, vol. 70, no. 1/2, pp. 37–60, 2004.
[6] B. W. Boehm and P. N. Papaccio, "Understanding and controlling software costs," *IEEE Trans. Softw. Eng.*, vol. 14, no. 10, pp. 1462–1477, Oct. 1988.
[7] A. Trendowicz, J. Münch, and R. Jeffery, "State of the practice in software effort estimation: A survey and literature review," in *Proc. IFIP Central East Eur. Conf. Softw. Eng. Techn.*, 2008, pp. 232–245.
[8] A. Trendowicz and J. Münch, "Factors influencing software development productivity–state-of-the-art and industrial experiences," *Advances Comput.*, vol. 77, pp. 185–241, 2009.
[9] M. Usman, E. Mendes, F. Weidt, and R. Britto, "Effort estimation in agile software development: A systematic literature review," in *Proc. 10th Int. Conf. Predictive Models Softw. Eng.*, 2014, pp. 82–91.
[10] M. Cohn, *Agile Estimating and Planning*. London, U.K.: Pearson Education, 2005.
[11] D. Leffingwell, *Scaling Software Agility: Best Practices for Large Enterprises*. London, U.K.: Pearson Education, 2007.

[12] M. Usman, E. Mendes, and J. Börstler, "Effort estimation in agile software development: A survey on the state of the practice," in *Proc. 19th Int. Conf. Eval. Assessment Softw. Eng.*, 2015, pp. 1–10.

[13] E. Dantas, M. Perkusich, E. Dilorenzo, D. F. Santos, H. Almeida, and A. Perkusich, "Effort estimation in agile software development: An updated review," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 28, no. 11n12, pp. 1811–1831, 2018.

[14] M. Cohn, *User Stories Applied: For Agile Software Development*. Boston, MA, USA: Addison-Wesley Professional, 2004.

[15] K. Schwaber and M. Beedle, *Agile Software Development With Scrum*, vol. 1. Upper Saddle River, NJ, USA: Prentice Hall, 2002.

[16] K. Conboy and N. Carroll, "Implementing large-scale agile frameworks: Challenges and recommendations," *IEEE Softw.*, vol. 36, no. 2, pp. 44–50, Mar./Apr. 2019.

[17] M. Paasivaara, "Adopting safe to scale agile in a globally distributed organization," in *Proc. IEEE 12th Int. Conf. Global Softw. Eng.*, 2017, pp. 36–40.

[18] M. Paasivaara, B. Behm, C. Lassenius, and M. Hallikainen, "Large-scale agile transformation at ericsson: A case study," *Empir. Softw. Eng.*, vol. 23, no. 5, pp. 2550–2596, 2018.

[19] H. F. Cervone, "Understanding agile project management methods using scrum," *OCLC Syst. Serv.: Int. Digit. Library Perspectives*, vol. 27, no. 1, pp. 18–22, 2011.

[20] A. Trendowicz and R. Jeffery, *Software Project Effort Estimation: Foundations and Best Practice Guidelines for Success*. Berlin, Germany: Springer, 2014, pp. 277–293.

[21] V. Mahnič and T. Hovelja, "On using planning poker for estimating user stories," *J. Syst. Softw.*, vol. 85, no. 9, pp. 2086–2095, 2012.

[22] C. J. Torrecilla-Salinas, J. Sedeño, M. Escalona, and M. Mejías, "Estimating, planning and managing agile web development projects under a value-based perspective," *Inf. Softw. Technol.*, vol. 61, pp. 124–144, 2015.

[23] P. Abrahamsson, I. Fronza, R. Moser, J. Vlasenko, and W. Pedrycz, "Predicting development effort from user stories," in *Proc. Int. Symp. Empir. Softw. Eng. Meas.*, 2011, pp. 400–403.

[24] D. Nguyen-Cong and D. Tran-Cao, "A review of effort estimation studies in agile, iterative and incremental software development," in *Proc. RIVF Int. Conf. Comput. Commun. Technol.-Res. Innov. Vis. Future*, 2013, pp. 27–30.

[25] S. Grapenthin, S. Poggel, M. Book, and V. Gruhn, "Facilitating task breakdown in sprint planning meeting 2 with an interaction room: An experience report," in *Proc. 40th EUROMICRO Conf. Softw. Eng. Adv. Appl.*, 2014, pp. 1–8.

[26] S. Kang, O. Choi, and J. Baik, "Model-based dynamic cost estimation and tracking method for agile software development," in *Proc. IEEE/ACIS 9th Int. Conf. Comput. Inf. Sci.*, 2010, pp. 743–748.

[27] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband, "A systematic literature review on agile requirements engineering practices and challenges," *Comput. Hum. Behav.*, vol. 51, pp. 915–929, 2015.

[28] M. Agrawal and K. Chari, "Software effort, quality, and cycle time: A study of CMM level 5 projects," *IEEE Trans. Softw. Eng.*, vol. 33, no. 3, pp. 145–156, Mar. 2007.

[29] M. Jørgensen and S. Grimstad, "Avoiding irrelevant and misleading information when estimating development effort," *IEEE Softw.*, vol. 25, no. 3, pp. 78–83, May/Jun. 2008.

[30] T. Menzies, Z. Chen, J. Hihn, and K. Lum, "Selecting best practices for effort estimation," *IEEE Trans. Softw. Eng.*, vol. 32, no. 11, pp. 883–895, Nov. 2006.

[31] M. Jørgensen and T. M. Gruschke, "The impact of lessons-learned sessions on effort estimation and uncertainty assessments," *IEEE Trans. Softw. Eng.*, vol. 35, no. 3, pp. 368–383, May/Jun. 2009.

[32] A. Trendowicz, M. Ochs, A. Wickenkamp, J. Münch, Y. Ishigai, and T. Kawaguchi, "Integrating human judgment and data analysis to identify factors influencing software development productivity," *e-Informatica*, vol. 2, no. 1, pp. 47–69, 2008.

[33] E. Kocaguneli, T. Menzies, and J. W. Keung, "On the value of ensemble effort estimation," *IEEE Trans. Softw. Eng.*, vol. 38, no. 6, pp. 1403–1416, Nov./Dec. 2012.

[34] P. Abrahamsson, R. Moser, W. Pedrycz, A. Sillitti, and G. Succi, "Effort prediction in iterative software development processes–incremental versus global prediction models," in *Proc. 1st Int. Symp. Empir. Softw. Eng. Meas.*, 2007, pp. 344–353.

[35] P. Hearty, N. Fenton, D. Marquez, and M. Neil, "Predicting project velocity in XP using a learning dynamic Bayesian network model," *IEEE Trans. Softw. Eng.*, vol. 35, no. 1, pp. 124–137, Jan./Feb. 2009.

[36] M. Choetkiertikul, H. K. Dam, T. Tran, T. T. M. Pham, A. Ghose, and T. Menzies, "A deep learning model for estimating story points," *IEEE Trans. Softw. Eng.*, vol. 45, no. 7, pp. 637–656, Jul. 2019.

[37] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Inf. Softw. Technol.*, vol. 54, no. 1, pp. 41–59, 2012.

[38] P. Sharma and J. Singh, "Systematic literature review on software effort estimation using machine learning approaches," in *Proc. Int. Conf. Next Gener. Comput. Inf. Syst.*, 2017, pp. 43–47.

[39] H. Zhang, L. Gong, and S. Versteeg, "Predicting bug-fixing time: An empirical study of commercial software projects," in *Proc. 35th Int. Conf. Softw. Eng.*, 2013, pp. 1042–1051.

[40] L. D. Panjer, "Predicting eclipse bug lifetimes," in *Proc. 4th Int. Workshop Mining Softw. Repositories*, 2007, pp. 29–29.

[41] P. Bhattacharya and I. Neamtiu, "Bug-fix time prediction models: Can we do better?," in *Proc. 8th Work. Conf. Mining Softw. Repositories*, 2011, pp. 207–210.

[42] Q. Song, M. Shepperd, M. Cartwright, and C. Mair, "Software defect association mining and defect correction effort prediction," *IEEE Trans. Softw. Eng.*, vol. 32, no. 2, pp. 69–82, Feb. 2006.

[43] S. Assar, M. Borg, and D. Pfahl, "Using text clustering to predict defect resolution time: A conceptual replication and an evaluation of prediction accuracy," *Empir. Softw. Eng.*, vol. 21, no. 4, pp. 1437–1475, 2016.

[44] C. Maddila, C. Bansal, and N. Nagappan, "Predicting pull request completion time: A case study on large scale cloud services," in *Proc. 27th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2019, pp. 874–882.

[45] J. G. March and Z. Shapira, "Managerial perspectives on risk and risk taking," *Manage. Sci.*, vol. 33, no. 11, pp. 1404–1418, 1987.

[46] K. De Bakker, A. Boonstra, and H. Wortmann, "Does risk management contribute to it project success? A meta-analysis of empirical evidence," *Int. J. Project Manage.*, vol. 28, no. 5, pp. 493–503, 2010.

[47] J. J. Jiang, G. Klein, and T. L. Means, "Project risk impact on software development team performance," *Project Manage. J.*, vol. 31, no. 4, pp. 19–26, 2000.

[48] J. J. Jiang, G. Klein, and R. Discenza, "Information system success as impacted by risks and development strategies," *IEEE Trans. Eng. Manage.*, vol. 48, no. 1, pp. 46–55, Feb. 2001.

[49] W.-M. Han and S.-J. Huang, "An empirical analysis of risk components and performance on software projects," *J. Syst. Softw.*, vol. 80, no. 1, pp. 42–50, 2007.

[50] L. Wallace, M. Keil, and A. Rai, "Understanding software project risk: A cluster analysis," *Inf. Manage.*, vol. 42, no. 1, pp. 115–125, 2004.

[51] B. W. Boehm, "Software risk management: Principles and practices," *IEEE Softw.*, vol. 8, no. 1, pp. 32–41, Jan. 1991.

[52] M. J. Carr, S. L. Konda, I. Monarch, F. C. Ulrich, and C. F. Walker, "Taxonomy-based risk identification," Carnegie-Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU/SEI-93-TR-006, 1993.

[53] R. N. Charette, *Software Engineering Risk Analysis and Management*. New York, NY, USA: Intertext Publications, 1989.

[54] C. Jones, *Assessment and Control of Software Risks*. Englewood Cliffs, NJ, USA: Yourdon Press, 1994.

[55] T. Addison and S. Vallabh, "Controlling software project risks: An empirical study of methods used by experienced project managers," in *Proc. Annu. Res. Conf. South African Inst. Comput. Scientists Inf. Technol. Enablement Through Technol.*, 2002, pp. 128–140.

[56] H. Barki, S. Rivard, and J. Talbot, "Toward an assessment of software development risk," *J. Manage. Inf. Syst.*, vol. 10, no. 2, pp. 203–225, 1993.

[57] R. Schmidt, K. Lyytinen, M. Keil, and P. Cule, "Identifying software project risks: An international delphi study," *J. Manage. Inf. Syst.*, vol. 17, no. 4, pp. 5–36, 2001.

[58] K. Ewusi-Mensah, *Software Development Failures*. Cambridge, MA, USA: MIT Press, 2003.

[59] S. L. Jarvenpaa and B. Ives, "Executive involvement and participation in the management of information technology," *MIS Quart.*, vol. 15, pp. 205–227, 1991.

[60] M. Van Genuchten, "Why is software late? An empirical study of reasons for delay in software development," *IEEE Trans. Softw. Eng.*, vol. 17, no. 6, pp. 582–590, Jan. 1991.

[61] H. Barki and J. Hartwick, "Rethinking the concept of user involvement," *MIS Quart.*, vol. 13, pp. 53–63, 1989.

[62] H. Hoodat and H. Rashidi, "Classification and analysis of risks in software engineering," *World Acad. Sci. Eng. Technol.*, vol. 56, no. 32, pp. 446–452, 2009.

[63] J. Ropponen and K. Lyytinen, "Components of software development risk: How to address them? A project manager survey," *IEEE Trans. Softw. Eng.*, vol. 26, no. 2, pp. 98–112, Feb. 2000.

[64] L. Wallace, M. Keil, and A. Rai, "How software project risk affects project performance: An investigation of the dimensions of risk and an exploratory model," *Decis. Sci.*, vol. 35, no. 2, pp. 289–321, 2004.

[65] J. Menezes, C. Gusmão, and H. Moura, "Risk factors in software development projects: A systematic literature review," *Softw. Qual. J.*, vol. 27, no. 3, pp. 1149–1174, 2019.

[66] E. Letier, D. Stefan, and E. T. Barr, "Uncertainty, risk, and information value in software requirements and architecture," in *Proc. 36th Int. Conf. Softw. Eng.*, 2014, pp. 883–894.

[67] M. Choetkiertikul, H. K. Dam, T. Tran, and A. Ghose, "Predicting delays in software projects using networked classification (t)," in *Proc. 30th IEEE/ACM Int. Conf. Autom. Softw. Eng.*, 2015, pp. 353–364.

[68] H. R. Joseph, "Poster: Software development risk management: Using machine learning for generating risk prompts," in *Proc. IEEE/ACM 37th Int. Conf. Softw. Eng.*, 2015, vol. 2, pp. 833–834.

[69] H. Kniberg and A. Ivarsson, "Scaling agile@ spotify with tribes, squads, chapters & guilds," *Entry Posted November*, vol. 12, pp. 1–14, 2012.

[70] B. A. Kitchenham and S. L. Pfleeger, "Principles of survey research: Parts 1– 6," *ACM SIGSOFT Softw. Eng. Notes*, vol. 28, pp. 24–27, 2003.

[71] M. Kasunic, "Designing an effective survey," Carnegie-Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU/SEI-2005-HB-004, 2005.

[72] E. Kula, E. Greuter, A. van Deursen, and G. Gousios, "Supplemental material for factors affecting on-time delivery in large-scale agile software development," 2020. [Online]. Available: https://tinyurl.com/yxkylqpb

[73] M. Jorgensen and K. Molokken-Ostvold, "Reasons for software effort estimation error: Impact of respondent role, information collection approach, and data analysis method," *IEEE Trans. Softw. Eng.*, vol. 30, no. 12, pp. 993–1007, Dec. 2004.

[74] F. Strack, ""Order effects" in survey research: Activation and information functions of preceding questions," in *Context Effects in Social and Psychological Research*. Berlin, Germany: Springer, 1992, pp. 23–34.

[75] J. S. Molléri, K. Petersen, and E. Mendes, "Survey guidelines in software engineering: An annotated review," in *Proc. 10th ACM/IEEE Int. Symp. Empir. Softw. Eng. Meas.*, 2016, pp. 1–6.

[76] J. Y. Cho and E.-H. Lee, "Reducing confusion about grounded theory and qualitative content analysis: Similarities and differences," *Qualitative Rep.*, vol. 19, no. 32, pp. 1–20, 2014.

[77] J. F. DeFranco and P. A. Laplante, "A content analysis process for qualitative software engineering research," *Innovations Syst. Softw. Eng.*, vol. 13, no. 2, pp. 129–141, 2017.

[78] J. Cohen, "A coefficient of agreement for nominal scales," *Educ. Psychol. Meas.*, vol. 20, no. 1, pp. 37–46, 1960.

[79] P. Judea, "Causality: Models, reasoning, and inference," *Cambridge University Press. ISBN 0*, vol. 521, no. 77362, 2000, Art. no. 8.

[80] S. Conte, H. Dunsmore, and V. Shen, *Software Engineering Metrics and Models*. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc., 1986.

[81] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, "A simulation study of the model evaluation criterion MMRE," *IEEE Trans. Softw. Eng.*, vol. 29, no. 11, pp. 985–995, Nov. 2003.

[82] B. A. Kitchenham, L. M. Pickard, S. G. MacDonell, and M. J. Shepperd, "What accuracy statistics really measure," *IEE Proc.-Softw.*, vol. 148, no. 3, pp. 81–85, 2001.

[83] D. Port and M. Korte, "Comparative studies of the model evaluation criterions MMRE and PRED in software cost estimation research," in *Proc. 2nd ACM-IEEE Int. Symp. Empir. Softw. Eng. Meas.*, 2008, pp. 51–60.

[84] Y. Miyazaki, M. Terakado, K. Ozaki, and H. Nozaki, "Robust regression for developing software estimation models," *J. Syst. Softw.*, vol. 27, no. 1, pp. 3–16, 1994.

[85] J. H. Friedman, "Multivariate adaptive regression splines," *The Ann. Statist.*, vol. 19, pp. 1–67, 1991.

[86] J. Noll and S. Beecham, "Measuring global distance: A survey of distance factors and interventions," in *Proc. Int. Conf. Softw. Process Improvement Capability Determination*, 2016, pp. 227–240.

[87] D. Hartmann and R. Dymond, "Appropriate agile measurement: Using metrics and diagnostics to deliver business value," in *Proc. AGILE*, 2006, pp. 6–134.

[88] S. E. Dreyfus and H. L. Dreyfus, "A five-stage model of the mental activities involved in directed skill acquisition," *Distribution*, Univ. California, Berkeley, Berkeley, CA, p. 22, Feb. 1980.

[89] C. Aube and V. Rousseau, "Team goal commitment and team effectiveness: The role of task interdependence and supportive behaviors," *Group Dynamics: Theory Res. Pract.*, vol. 9, no. 3, 2005, Art. no. 189.

[90] K. M. Chudoba, E. Wynn, M. Lu, and M. B. Watson-Manheim, "How virtual are we? Measuring virtuality and understanding its impact in a global organization," *Inf. Syst. J.*, vol. 15, no. 4, pp. 279–306, 2005.

[91] O. Morgenshtern, T. Raz, and D. Dvir, "Factors affecting duration and effort estimation errors in software development projects," *Inf. Softw. Technol.*, vol. 49, no. 8, pp. 827–837, 2007.

[92] R. S. Huckman, B. R. Staats, and D. M. Upton, "Team familiarity, role experience, and performance: Evidence from indian software services," *Manage. Sci.*, vol. 55, no. 1, pp. 85–100, 2009.

[93] K. Dikert, M. Paasivaara, and C. Lassenius, "Challenges and success factors for large-scale agile transformations: A systematic literature review," *J. Syst. Softw.*, vol. 119, pp. 87–108, 2016.

[94] H. Huang and E. M. Trauth, "Cultural influences and globally distributed information systems development: Experiences from chinese it professionals," in *Proc. ACM SIGMIS CPR Conf. Comput. Personnel Res.: The Global Inf. Technol. Workforce*, 2007, pp. 36–45.

[95] M. Kajko-Mattsson, "Problems in agile trenches," in *Proc. 2nd ACM-IEEE Int. Symp. Empir. Softw. Eng. Meas.*, 2008, pp. 111–119.

[96] J. Kontio, M. Hoglund, J. Ryden, and P. Abrahamsson, "Managing commitments and risks: Challenges in distributed agile development," in *Proc. 26th Int. Conf. Softw. Eng.*, 2004, pp. 732–733.

[97] N. Sekitoleko, F. Evbota, E. Knauss, A. Sandberg, M. Chaudron, and H. H. Olsson, "Technical dependency challenges in large-scale agile software development," in *Proc. Int. Conf. Agile Softw. Develop.*, 2014, pp. 46–61.

[98] D. E. Strode, S. L. Huff, B. Hope, and S. Link, "Coordination in co-located agile software development projects," *J. Syst. Softw.*, vol. 85, no. 6, pp. 1222–1238, 2012.

[99] A. Elbanna and S. Sarker, "The risks of agile software development: Learning from adopters," *IEEE Softw.*, vol. 33, no. 5, pp. 72–79, Sep./Oct. 2016.

[100] A. L. Lederer and J. Prasad, "Causes of inaccurate software development cost estimates," *J. Syst. Softw.*, vol. 31, no. 2, pp. 125–134, 1995.

[101] S. Grimstad, M. Jorgensen, and K. Molokken-Ostvold, "The clients' impact on effort estimation accuracy in software development projects," in *Proc. 11th IEEE Int. Softw. Metrics Symp.*, 2005, pp. 10–pp.

[102] K. M. Furulund and K. Molkken-stvold, "Increasing software effort estimation accuracy using experience data, estimation models and checklists," in *Proc. 7th Int. Conf. Qual. Softw.*, 2007, pp. 342–347.

[103] R. Gupta, K. H. Prasad, and M. Mohania, "Automating ITSM incident management process," in *Proc. Int. Conf. Auton. Comput.*, 2008, pp. 141–150.

[104] B. Fitzgerald, K.-J. Stol, R. O'Sullivan, and D. O'Brien, "Scaling agile methods to regulated environments: An industry case study," in *Proc. 35th Int. Conf. Softw. Eng.*, 2013, pp. 863–872.

[105] F. Moyon, K. Beckers, S. Klepper, P. Lachberger, and B. Bruegge, "Towards continuous security compliance in agile software development at scale," in *Proc. IEEE/ACM 4th Int. Workshop Rapid Continuous Softw. Eng.*, 2018, pp. 31–34.

[106] L. ben Othmane, P. Angin, H. Weffers, and B. Bhargava, "Extending the agile development process to develop acceptably secure software," *IEEE Trans. Dependable Secure Comput.*, vol. 11, no. 6, pp. 497–509, Nov./Dec. 2014.

[107] E. Giger, M. Pinzger, and H. Gall, "Predicting the fix time of bugs," in *Proc. 2nd Int. Workshop Recommendation Syst. Softw. Eng.*, 2010, pp. 52–56.

[108] F. S. Downs and J. Fawcett, *The Relationship of Theory and Research*. London, U.K.: McGraw-Hill/Appleton & Lange, 1986.

[109] J. S. Molléri, K. Petersen, and E. Mendes, "An empirically evaluated checklist for surveys in software engineering," *Inf. Softw. Technol.*, vol. 119, 2020, Art. no. 106240.

[110] R. Donmoyer, "Generalizability and the single-case study," in *Case Study Method: Key Issues, Key Texts*. Thousand Oaks, CA, USA: SAGE Publications, 2000, pp. 45–68.

[111] E. D. De Leeuw, *Data Quality in Mail, Telephone and Face to Face Surveys*. San Jose, CA, USA: ERIC, 1992.

[112] R. M. Groves, R. B. Cialdini, and M. P. Couper, "Understanding the decision to participate in a survey," *Public Opin. Quart.*, vol. 56, no. 4, pp. 475–495, 1992.

[113] A. Furnham, "Response bias, social desirability and dissimulation," *Pers. Indiv. Differ.*, vol. 7, no. 3, pp. 385–400, 1986.

[114] P. Ralph and E. Tempero, "Construct validity in software engineering research and software metrics," in *Proc. 22nd Int. Conf. Eval. Assessment Softw. Eng.*, 2018, pp. 13–23.

Elvan Kula (Member, IEEE) is currently working toward the doctoral degree at the Delft University of Technology, The Netherlands. She focuses on using automated techniques to both understand and improve software development processes in terms of efficiency and predictability. Her research interests include effort estimation, software analytics, and machine learning for software engineering. She is the manager of AI for Fintech Research, a five year research collaboration between ING and TU Delft.

Eric Greuter is chief product owner at the IT Infrastructure Data & Analytics Department of ING TECH, Netherlands. He leads several engineering and research teams, and is responsible for the backlog management data warehouse at ING, Netherlands. His research interests include software processes, data integration, and cloud infrastructure services.

Arie van Deursen (Member, IEEE) is a full professor in software engineering at the Delft University of Technology, The Netherlands, where he heads the Software Engineering Research Group and chairs the Department of Software Technology. His research interests include human aspects of software engineering, software architecture, and software testing. He serves on the advisory board of the Innovation Center for AI (ICAI). He is founder and scientific director of AI for Fintech Research (AFR), and one of the currently 20 ICAI labs. He served as program co-chair for ESEC/FSE 2017 and for ICSE 2021. He serves on the advisory board of the *Empirical Software Engineering* and on the editorial board of the *PeerJ Computer Science*.

Georgios Gousios is a research engineer at Facebook and an associate professor at the Delft University of Technology, The Netherlands (on leave). He works in the fields of software analytics, software ecosystems, software processes, and machine learning for software engineering. He is the main author of the GHTorrent data collection and curation framework and various widely used tools and datasets.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.