Questions for Data Scientists in Software Engineering: A Replication

Huijgens, Hennie; Rastogi, Ayushi; Mulders, Ernst; Gousios, Georgios; Deursen, Arie van

DOI
10.1145/3368089.3409717

Publication date
2020

Document Version
Accepted author manuscript

Published in
Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering

Citation (APA)
Huijgens, H., Rastogi, A., Mulders, E., Gousios, G., & Deursen, A. V. (2020). Questions for Data Scientists in Software Engineering: A Replication. In P. Devanbu, M. Cohen, & T. Zimmermann (Eds.), *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 568–579). (ESEC/FSE 2020). ACM. https://doi.org/10.1145/3368089.3409717

Questions for Data Scientists in Software Engineering: A Replication

Hennie Huijgens
Delft University of Technology
Delft, The Netherlands
h.k.m.huijgens@tudelft.nl

Ayushi Rastogi
Ernst Mulders*
Delft University of Technology
Delft, The Netherlands
a.rastogi@tudelft.nl
ernst@mulde.rs

Georgios Gousios
Arie van Deursen
Delft University of Technology
Delft, The Netherlands
g.gousios@tudelft.nl
arie.vandeursen@tudelft.nl

ABSTRACT

In 2014, a Microsoft study investigated the sort of questions that data science applied to software engineering should answer. This resulted in 145 questions that developers considered relevant for data scientists to answer, thus providing a research agenda to the community. Fast forward to five years, no further studies investigated whether the questions from the software engineers at Microsoft hold for other software companies, including software-intensive companies with different primary focus (to which we refer as *software-defined enterprises*). Furthermore, it is not evident that the problems identified five years ago are still applicable, given the technological advances in software engineering.

This paper presents a study at ING, a software-defined enterprise in banking in which over 15,000 IT staff provides in-house software solutions. This paper presents a comprehensive guide of questions for data scientists selected from the previous study at Microsoft along with our current work at ING. We replicated the original Microsoft study at ING, looking for questions that impact both software companies and software-defined enterprises and continue to impact software engineering. We also add new questions that emerged from differences in the context of the two companies and the five years gap in between. Our results show that software engineering questions for data scientists in the software-defined enterprise are largely similar to the software company, albeit with exceptions. We hope that the software engineering research community builds on the new list of questions to create a useful body of knowledge.

CCS CONCEPTS

• **General and reference** → *Surveys and overviews.*

KEYWORDS

Data Science, Software Engineering, Software Analytics.

*Work completed during an internship at ING.

ESEC/FSE '20, November 8–13, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7043-1/20/11...$15.00
https://doi.org/10.1145/3368089.3409717

ACM Reference Format:
Hennie Huijgens, Ayushi Rastogi, Ernst Mulders, Georgios Gousios, and Arie van Deursen. 2020. Questions for Data Scientists in Software Engineering: A Replication. In *Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '20), November 8–13, 2020, Virtual Event, USA*. ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3368089.3409717

1 INTRODUCTION

Software engineering researchers try solving problems that are relevant to software developers, teams, and organizations. Historically, researchers identified these problems from their experience, connections in industry and/or prior research. In 2014, however, a study at Microsoft [7] systematically analyzed software engineering questions that data scientists can answer and made it accessible to a wider audience.

Switching context, in the past few years ING transformed itself from a finance-oriented company to a software-defined, data-driven enterprise. From a software engineering perspective, this includes the implementation of fully automated release engineering pipelines for software development activities in more than 600 teams performing 2,500+ deployments per month for 750+ applications. These activities leave a trove of data, suggesting that data scientists using, e.g., modern machine learning techniques could offer valuable and actionable insights to ING.

To that end, ING needs questions that are relevant for their engineers which their data scientists can answer. As we started looking for existing resources, we came across the 145 software engineering questions for data scientists presented in the Microsoft study [7]. However, before adopting the list, we wanted to know:

RQ: To what extent do software engineering questions relevant for Microsoft apply to ING, five years later?

Microsoft is a large software company, while ING that is a FinTech company using software to improve its banking solutions (software-defined enterprise). Moreover, the two companies are at different scale. In 2014, Microsoft had more than 30,000 engineers while even today ING is almost half its size with approximately 15,000 IT employees (on a total of 45,000). More details on the differences in the context of the two companies are available in Table 1. We try to understand whether the questions relevant for a software company extend to a software-defined enterprise. We compare the results of the original Microsoft study [7] with our results at ING to understand the relevance of the questions beyond Microsoft but also as a guide for other software-defined enterprises that are undergoing their digital transformation. We further explore whether the technological advances in the last five years changed the way we develop software. To answer this question, we carried out a replication of the original Microsoft study at ING. Similar to the original study, we conducted two surveys: one, to find data science problems in software engineering, and second, to rank the questions in the order of their relevance (see Figure 1). For the first survey, we randomly sampled 1,002 ING engineers and received 116 responses with 336 questions. We grouped the 336 questions on similarities resulting in 171 descriptive questions. We shared subsets of these 171 descriptive questions with another random sample of 1,296 ING engineers for ranking. In the end, we received 21,888 rankings from 128 ING engineers. These ranked 171 questions are the questions that engineers at ING would like data scientists to solve. Further, we compare our list of 171 questions to the original list of 145 questions to answer our research question. Our study shows that the core software development problems, relating to *code* (e.g. understanding code, testing, and quality), *developer productivity* (both individuals and team) and *customer* are same for the software company and the software-defined enterprise. Nonetheless, subtle differences in the type of questions point to changes in market as well as differences in the context of the two companies.

2 IMPACT OF THE MICROSOFT 2014 STUDY

In order to gain a good insight into the further course of the Microsoft 2014 study after it was published, including any implications for research, we conducted a citation analysis. In addition, we looked at studies that have not quoted the Microsoft study, but that are relevant to our study. Hence this section also serves as our discussion of related work. We investigated the 136 studies that, according to Google Scholar, quote the Microsoft study. First of all, we looked at the number of times that the 136 studies themselves were cited by other studies; we limited the further analysis to 70 studies with a citation per year greater than 1.00. We then characterized studies into empirical approach, reference characterization, SE topic, and machine learning (ML) topic (see Table 2). Note that one paper can belong to multiple topics. We made the following observations:

Microsoft itself is building on its study. 11% of the citations come from Microsoft studies itself, mostly highly cited studies on SE culture, such as [18, 41, 51]. we notice that all citing Microsoft studies use a survey among a large number of SE practitioners (ranging from 16 to 793 respondents with a median of 311), whereas other studies based on a survey generally reach substantially lower numbers of participants.

Table 1: Context of Microsoft in 2014 and ING in 2019.

	Microsoft 2014	ING 2019
Branch	Software Company	Banking (FinTech)
Organization Size	Approx. 100,000 (in 2014), about 30,000 engineers	45,000 employees of which 15,000 IT
Team Structure	Typically size 5 ± 2	600 teams of size 9 ± 2
Development Model	Agile/Scrum (60%+)	Agile (Scrum / Kanban)
Pipeline automation	Every team is different. Continuous Integration in many teams	Continuous Delivery as a Service
Development Practice	DevOps	(Biz)DevOps

Table 2: Characterizations of Citing Studies.

Empirical Approach (n = 70)	Number of studies	Percentage
Analysis of SE process data (e.g. IDE)	30	43%
Survey SE practitioners	17	24%
Interview SE practitioners	7	10%
Literature review	5	7%
Experiment, case, or field study	5	7%

Reference characterization (n = 70)	Number of studies	Percentage
Plain reference in related work	38	54%
Reference as example for study setup	27	39%
Study partly answers MS question	9	13%
Study explicitly answers MS question	3	4%

Software Engineering Topic (n = 70)	Number of studies	Percentage
Software analytics, data science	20	29%
Testing, debugging, quality, code review	15	21%
Software engineering process	12	17%
Software engineering culture	9	13%
Mobile apps	3	4%

Machine Learning Topic (n = 24)	Number of studies	Percentage
Examples of Machine Learning applications	8	11%
Natural Language Processing	5	7%
Ensemble Algorithms	3	4%
Instance-based Algorithms	2	3%
Deep Learning Algorithms	2	3%
Other	4	5%

Half of the citing studies analyze SE process data, and 24% uses a survey. Looking at the empirical approach (see the first sub-table in Table 2), indicates that 43% of the studies contain a quantitative component, in which analysis of SE process data in particular is part of the study. Good examples are [9, 28]. Furthermore, 24% of the citing studies uses a survey among SE practitioners, for example [18, 22, 45, 69, 75]. Ten percent is based on interviews with SE practitioners, such as [20, 41, 42, 50]. Seven percent contains a literature review, for example [12, 45, 73]. Another 7% conducts an experiment [33, 62], case study [49, 59], or field study [9, 10].

Only three out of 70 studies explicitly answer a question from the initial Microsoft study. The second sub-table in Table 2 shows that only 3 studies (4%) explicitly refer their research question to an initial Microsoft one: [16, 28, 33]. Nine studies (13%) partly try to answer a MS question: [8–10, 30, 52, 62, 64, 65, 70]. 29 studies (39%) refer to the original Microsoft study because they used it as an example for their own study [17, 59], either with regard to the study design [20, 22, 29, 37, 46, 48, 67], the rating approach (Kano) [51, 61], or the card sorting technique [19, 54, 60, 63]. Furthermore, a large part (38 studies, 54%) of the citing studies simply refers to the original Microsoft study in a simple related work way.

A majority of citing studies is about Software Analytics, Testing related studies, and SE Process. The third sub-table shows that most cited studies are about software analytics, often combined with a focus on the role of the software engineer and its perceptions, e.g. [42, 51]. In other cases the emphasis on software analytics is combined with a more technical focus on machine learning, e.g. [21, 48]. Other studies within the topic software analytics are about a variety of methods, tools, and techniques [2, 3, 11, 14, 15, 27, 38, 47, 55, 71–73]. Many of the studies that cite the Microsoft study—and which are often quoted themselves—relate to testing or test automation. Fifteen studies (21%) are about testing [8–10, 13, 23, 24, 33, 45, 66], debugging [80] and code review [25, 46].

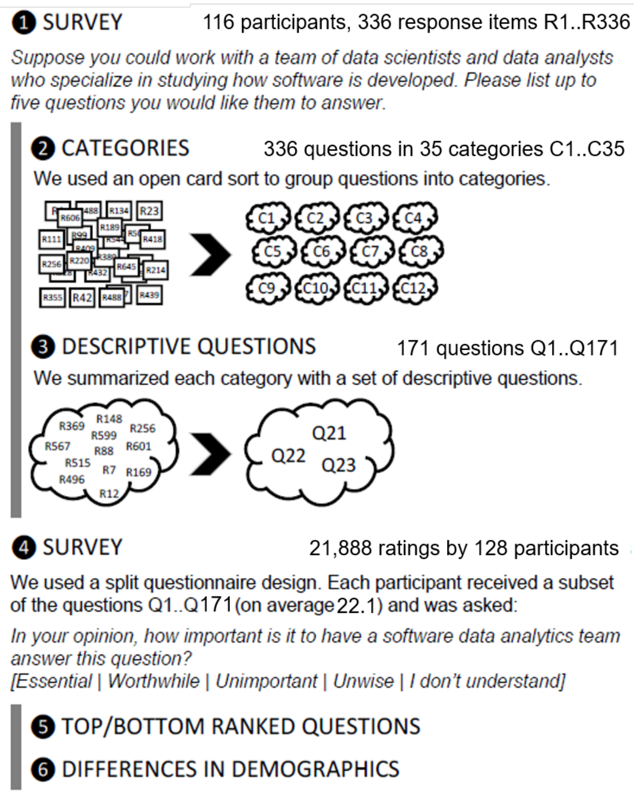
12 studies (17%) handle SE process related topics, such as productivity of software engineers [52], visualization [6, 31], and continuous delivery [74, 76]. In addition, studies also relate to continuous delivery pipelines and pipeline automation [74, 78]. Another frequent topic in citing studies is data and models, including aspects of cloud development [32, 49, 55]. Driven by a tendency toward automation of pipelines, software generates a large amount of data. Many different data sources—such as version control systems, peer code review systems, issue tracking systems, mail archives—are available for mining purposes [29, 79].

34% of the cited studies includes some form of Machine Learning. One third of the citing papers do include some form of machine learning (ML), ranging from applying a ML technique for analysis purposes to coming up with examples of the application of ML in practice. As the fourth sub-table in Table 2 shows, 8 studies include examples of applications of ML in practice, e.g. [11, 41, 55]. Text related techniques such as NLP occur 5 times, e.g. [23, 61], ensemble techniques 3 times [30, 37, 60], and instance-based and deep learning both 2 times [14, 21, 27, 48]. Four other techniques—neural networks, clustering, decision trees, and regression—occur one time. Perhaps this finding supports a trend that is visible in SE research, where more and more machine learning techniques are being used in SE analyzes and vice versa, also called *AI-for-Software-Engineering* [1, 40, 53].

13% are about the cultural aspects of software engineering. Software analytics is an area of extensive growth [56]. The original Microsoft 2014 study influenced ongoing research, looking at the 136 papers citing it gives the impression that it certainly did inspire other researchers and practitioners in setting up studies on software developers needs. Nine studies (13%) of the citing studies are about cultural aspects of software engineering, such as topic selection in experiments [58], characteristics of software engineers [20, 50, 67], causes for frustration [19], or challenges for software engineers [29, 63, 69].

3 STUDY DESIGN

Our study design comprises of two parts. In part one, we replicate the original Microsoft study at ING. We follow the step-by-step procedure prescribed in the original study, with slight modifications appropriate for our contextFigure 1 depicts the research methodology we followed; the figure is an exact copy of the approach used in the original Microsoft 2014 study with numbers from our study. In the next step, we compare the questions identified in the Microsoft study to ours for similarities and differences including addition of new questions and removal of previous questions to answer our research questions.



This figure is a copy from the original Microsoft 2014 study, with numbers from our study. The figure was re-used with permission of the Microsoft 2014 study authors.

Figure 1: Overview of the research methodology

3.1 The Initial Survey

We sent the initial survey to 1,002 ING software engineers randomly chosen from a group of 2,342 employees working within the IT department of ING in May 2018. Unlike the Microsoft study, we did not offer any reward to increase the participation. This is a deviation from the original study but aligns with the policy of ING. Out of the 1,002 engineers 387 engineers started the survey, 271 of them even filled the demographics but stopped when asked to write questions. In the end, we received 336 questions from 116 responses for a response rate of 11.6%. Table 3 shows the distribution of responses across discipline and role.

3.2 Coding and Categorization

Next we did an open card sort to group 336 questions into categories. Our card sort was open, meaning that we coded independently from the Microsoft study. To create independent codes, the first author who did a majority of the coding did not study the Microsoft paper before or during the replication. The other authors knew the paper from before and merely skimmed the methodology section for replication.

We let the groups emerge and evolve during the sorting process. This process comprised of three phases. In *preparation phase*, we created a card for each question. Questions 1 to 40 were tagged by the second author. Questions 41 to 80 were tagged by the fourth author. Questions 81 to 90 were tagged by both the second and the fourth author. The tags of questions 1 to 90 were discussed by both the second and fourth author and based on their discussion final tags were prepared. The remaining questions 91 to 336 were then tagged by the first author, based on the tags from the previous step. We discarded cards that made general comments on software development and did not inquire any specific topic.

In the *execution phase*, cards were sorted into meaningful groups and were assigned a descriptive title. Similar to the Microsoft study, the questions were not easy to work with; many questions were same or similar to one another, most were quite verbose while others were overly specific. We distilled them into a set of so-called descriptive questions that more concisely describe each category (and sub-category). In this step, out of the 336 questions, 49 questions were discarded and the remaining 287 questions were divided into 35 sub-categories. An example of reaching descriptive question is presented below[1]:

🗩 *'What factors affect the composition of DevOps teams?'*

from the following respondents' questions:

🗨*"Would it be better to create specialized development teams instead of DevOps teams?"*

🗨*"What is your idea of an ideal team that should develop software? How many and what kind of people should be part of it?"*

Finally, in the *analysis phase*, we created abstract hierarchies to deduce general categories and themes. In total, we created 171 descriptive questions, a full list of which is available in the appendix.

3.3 The Rating Survey

We created a second survey to rate the 171 descriptive questions. We split the questionnaire into eight component blocks (similar to the Microsoft study) and sent component blocks to potential respondents. The idea behind using the *split questionnaire survey design* is to avoid low response rate. Each participant received a block of questions along with a text *"In your opinion, how important is it to have a software data analytics team answer this question?"* with possible answers as *"Essential"*, *"Worthwhile"*, *"Unimportant"*, *"Unwise"*, and *"I don't understand"* [39].

[1] A closed balloon indicates a respondent question; an open balloon indicates a descriptive question.

Table 3: Distribution of responses based on discipline and role in the initial survey as well as rating survey.

Discipline	Initial Survey	Rating Survey
Development & Testing	62.0%	68.8%
Project Management	2.0%	3.9%
Other Engineering (e.g. architect)	28.0%	19.5%
Non-Engineering	8.0%	7.8%

Current Role	Initial Survey	Rating Survey
Developer	51.1%	20.0%
Lead	14.3%	18.7%
Architect	9.0%	11.8%
Manager & Executive	8.3%	20.0%
Other	17.3%	29.6%

The rating survey was sent to the remaining 1,296 software engineers at ING. Here too, 360 engineers started the survey (28%), but many of them did not complete it (36% drop-out rate). Finally, we received 128 responses, for a somewhat low response rate of 10%. On an average each question received 21,888/177=123 ratings making the resulting ranks stable. Table 3 shows the distribution of responses for the rating survey based on discipline and current role.

3.3.1 *Top-Rated/Bottom-Rated Questions.*

Finally, to rank each question, we dichotomized the ordinal Kano scale avoiding any scale violations [44]. We computed the following percentages for each descriptive question:

- Percentage of '**Essential**' responses among all the responses:

$$\frac{Essential}{Essential + Worthwhile + Unimportant + Unwise}$$

- Percentage of 'Essential' and 'Worthwhile' responses among all the responses (to which we refer as **Worthwhile+**):

$$\frac{Essential + Worthwhile}{Essential + Worthwhile + Unimportant + Unwise}$$

- Percentage of '**Unwise**' responses among all the responses:

$$\frac{Unwise}{Essential + Worthwhile + Unimportant + Unwise}$$

We rank each question based on the above percentages, with the top rank (#1) having the highest percentage in a dimension (Essential, Worthwhile+, or Unwise). Table 5 and Table 6 presents the most desired (Top 10 Essential, Top 10 Worthwhile+) and the most undesired (Top 10 Unwise) descriptive questions. For all 171 questions and their rank, see the appendix.

3.3.2 *Rating by Demographics.*

Unlike the Microsoft study, we did not have employee database to rank responses based on demographics, and privacy regulations prevented us from asking people-related aspects such as years of experience (another deviation from the original study). Nonetheless, in both the initial and the rating survey, we asked the following professional background data from the participants:

- *Discipline:* Participants were asked to indicate their primary working area: *Development*, *Test*, *Project Management*, *Other Engineer* (e.g. architect, lead), or *Other Non-Engineer* (only one selection was possible).
- *Current Role:* Participants were asked to indicate their current role: *Individual Contributor*, *Lead*, *Architect*, *Manager*, *Executive*, or *Other* (more selections were possible).

To investigate the relations of descriptive questions to professional background (discipline or current role), we built stepwise logistic regression models. We build our own models since the referenced study did not share scripts to run statistical tests although we did follow their procedure as is. Stepwise regression eliminated professional backgrounds that did not improve the model for a given question and a response. In addition, we removed professional backgrounds for which the coefficient in the model was not statistically significant at p-value < 0.01. For each of the 171 questions, we built a model with Essential response (yes/no) as a dependent variable and professional background as independent variable. We built similar models for Worthwhile+ and Unwise responses. In total, we built 513 models, three for each of the 171 descriptive questions.

3.4 Comparison of Questions

As a preliminary analysis, we start by looking at the similarities and differences in the broader themes or categories in both the studies. Then for each theme, we see how the prominent questions in ING compare against the prominent questions at Microsoft.

To make the comparison systematic, we followed a two-step approach. First, we ran word counts on the questions from both the companies presenting a text-based comparison to identify broad differences. Further, the first two authors manually analyzed top 100 essential questions from the two companies in detail. The authors drew affinity diagrams using Microsoft questions (see Figure 2) and appended related questions from ING to it. In case no cluster fits a question, a new cluster is created. This resulted in three types of clusters: match and no match (addition of ING questions and deletion of Microsoft questions). Analyses of the three clusters and the frequency distribution of questions (in addition to the previous three analyses) present insights into our research question.

4 RESULTS

The original Microsoft study came up with 145 questions that software engineers want data scientists to answer. Replicating the original study at ING, we identified 171 data science questions.

This section presents a comparison of the two sets of questions based on category, type of questions within categories, top-rated questions, bottom-rated questions, and questions relevant for different demographics. Next, we compare the questions from the two companies using word count and affinity diagrams to answer our research question.

4.1 Categories

We noticed that some of our categories directly match the Microsoft study. Other categories, however, can be mapped to one or more categories of the Microsoft study. No new emergent category in our study indicates that broadly there are no differences between the questions for a software-defined enterprise from a software company. For further analysis, we map our categories on to theirs, details on which are available in Table 4.

Next, we explore the essential questions at ING and their distinguishing link to the questions from the Microsoft study.

4.1.1 Bug Measures (BUG). The essential questions at ING relate to the effort spent on bugs, methods to prevent security-related vulnerabilities, and the relationship between bugs and specific ING-related development platforms.

🗨*"How does the effort spent on fixing vulnerabilities and bugs relate to effort spent on writing software correctly from the start?"*

🗨*"What methods are most effective in preventing security-related vulnerabilities or bugs from being introduced in software code?"*

4.1.2 Development Practices (DP). The performance and productivity of DevOps teams was found in a number of questions including team happiness and work pleasure (# 1 question), ways of decision making, non-overlapping development activities in the same environment, product ownership and business responsibilities, licensing of tools, and the choice of a data modeling approach.

🗨*"What factors affect the performance and productivity of DevOps teams with regard to team happiness and pleasure in your work?"*

🗨*"What factors affect the performance and productivity of DevOps teams with regard to evidence-based decision-making versus decision-making based on expert opinions?"*

🗨*"What factors affect the performance and productivity of DevOps teams with regard to simultaneous slow and fast developments at the same time in the same environment?"*

4.1.3 Development Best Practices (BEST). This category emphasized best (or worst) development practices relating to technology selection, effectiveness, and choice of tools.

🗨*"How can we make sure that we build for re-usability and scalability?"*

🗨*"What factors affect high performance teams?"*

🗨*"When do you remove an old module that you think is not being used anymore?"*

4.1.4 Testing Practices (TP). Questions here ranged from automated test data generation, on-demand provisioning of test environments, testing of high volumes, to question like "should we let loose Chaos Monkey" [35] [5]

🗨*"To what extent does on-demand provisioning of development and test environments, including up-to-date data affect delivery of software solutions?"*

🗨*"What factors affect performance testing on high data volumes?"*

🗨*"How can a system for (semi) automated CRUD test data generation improve delivery of software solutions?"*

🗨*"Should we let loose Chaos Monkey, like Netflix?*

4.1.5 Evaluating Quality (EQ). This category included questions on code analysis, ways to assess quality of software code, and effectiveness of testing practices.

🗨*"What methods can be applied to analyze whether software code is working as expected?"*

🗨*"To what extent does testability of software code affect the quality of code?"*

4.1.6 Customers and Requirements (CR). The essential questions related to measure customer value, requirement validation, and the use of formal models. Notably, questions relating to development trade-offs such as backward compatibility or the impact of testing in production appeared in the Microsoft study but not ours.

🗨*"How to measure the customer value of a software product?"*

🗨*"How can requirements be validated before starting actual software development?"*

🗨*"How can user feedback be integrated in an efficient and effective way into software code?"*

4.1.7 Software Development Lifecycle (SL). Questions in this category related to the effectiveness and performance in lead time,

Table 4: ING categories and questions mapped on to the 12 Microsoft categories

		ING 2019 Study				Microsoft 2014 Study				
Category		Cards		Subcategories	Descriptive Questions	Cards		Subcategories	Descriptive Questions	Difference ING 2019 compared to MS 2014
Teams and Collaboration	TC	14	4%	5	7	73	10%	7	11	$\downarrow$ 6%
Testing Practices	TP	32	9%	3	15	101	14%	5	20	$\downarrow$ 5%
Services	SVC	3	1%	2	1	42	6%	2	8	$\downarrow$ 5%
Reuse and Shared Components	RSC	5	1%	3	2	31	4%	1	3	$\downarrow$ 3%
Customers and Requirements	CR	9	3%	2	7	44	6%	5	9	$\downarrow$ 3%
Software Development Lifecycle	SL	6	2%	4	4	32	4%	3	7	$\downarrow$ 2%
Development Practices	DP	51	15%	14	38	117	16%	13	28	$\downarrow$ 1%
Bug Measurements	BUG	6	2%	3	5	23	3%	4	7	$\downarrow$ 1%
Productivity	PROD	29	9%	8	20	57	8%	5	13	$\uparrow$ 1%
Evaluating Quality	EQ	38	11%	6	11	47	6%	6	16	$\uparrow$ 5%
Development Best Practices	BEST	49	15%	7	36	65	9%	6	9	$\uparrow$ 6%
Software Development Process	PROC	46	14%	7	25	47	6%	3	14	$\uparrow$ 8%
Discarded Cards		*49*	*15%*			*49*	*7%*			$\uparrow$ 8%
Total Cards		337	100%	64	171	728	100%	60	145	

Table sorted on the percentage difference in the number of questions in the ING study compared to the Microsoft study.

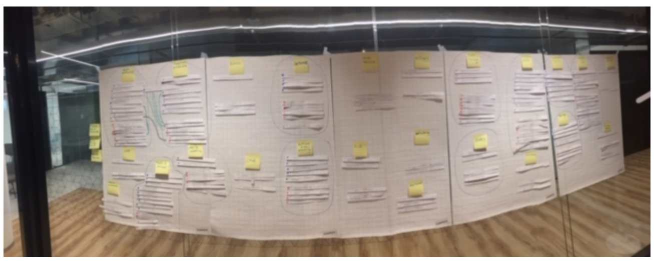


Figure 2: Analysis of ING 2019 and MS 2014 questions.

cost, and quality (same as the Microsoft study) but also questions relating to security and risk from a management perspective.

"What factors affect providing new technologies to consumers, and can implementations of new technology be internally and externally benchmarked?"

"What factors affect estimation of lead time, cost, and quality of software deliveries?"

4.1.8 Software Development Process (PROC). Our questions related to development processes, technology selection, and deployment of software solutions. At Microsoft, in contrast, questions related to the choice of software methodology (e.g. ways in which agile is better than waterfall? and benefits of pair programming). We also noticed that at ING topics like the effects of automated continuous delivery pipeline popped up which were not seen in the Microsoft study.

"How can we improve the deployment process in DevOps teams?"

"Does a focus on quick release of features and bug fixes into production help to achieve confidence and agility?"

4.1.9 Productivity (PROD). This category had questions on the productivity of DevOps teams - but also individual developers, ranked essential. Notably, questions related to the measurement of individual developers (e.g. the questions mentioned below regarding "great coder" and "open spaces") were often ranked "Unwise". Quite unlike the Microsoft study, where respondents considered these questions as unwise, engineers at ING had a mixed opinion.

"What factors affect the performance of DevOps teams and the quality of software code with regard to quantity and quality of environments?"

"Are developers working in an open space with several teams more effective or less than developers working in a room with just their team?"

"What makes a great coder? What aspects affect the performance of DevOps teams and the quality of software with regard to characteristics of an individual software engineer?"

4.1.10 Teams and Collaborations (TC). Essential questions here are typically about dependencies between teams, team composition, team maturity, and knowledge sharing among teams.

"To what extent do dependencies on other teams affect team performance?"

"How does team maturity affect code quality and incidents?"

"What factors affect the composition of DevOps teams?"

4.2 Top-Rated Questions

Table 5 shows top 15 "Essential" and top 10 "Worthwhile or higher" questions. Interestingly, only two out of the top 15 "Essential" questions were a part of the top 10 "Worthwhile or higher" questions and none vice-versa. This potentially means that our participants are more pronounced and opt for Essential or Worthwhile only when they feel so. Culture can be another possible reason since all participants at ING are located in one country while participants of the Microsoft study were more diverse [34].

Our top questions are on development processes, technology selection, and deployment of software solutions. The top related questions at Microsoft, in contrast, relates to the choice of software methodology (e.g. ways in which agile is better than waterfall? and benefits of pair programming). We also noticed that in our study topics like the effects of automated continuous delivery pipeline popped up which were not seen in the Microsoft study.

Notably, a large fraction of the top 20 "Essential" or "Worthwhile or higher" questions at Microsoft (9 out of 20; including top 2) relates to customers. This suggests that for Microsoft customer benefit is most important or perhaps one of the most important question. Our study, in contrast, paints a very different picture.

Table 5: Questions with the highest "Essential" and "Worthwhile or higher" percentages.

	Question	Category	Percentages			Rank		
			Essential	Worthwhile+	Unwise	Essential	Worthwhile+	Unwise
⚒ Q143	What factors affect the performance and productivity of DevOps teams with regard to team happiness and pleasure in your work?	DP	68.4%	94.7%	0.0%	1	9	68
★ Q98	How does on-demand provisoning of develop- and test environments, including up-to-date data affect delivery of software solutions?	TP	66,7%	77,8%	0,0%	2	95	68
★ Q37	How can we make sure that we build for reusability and scalability?	BEST	63.2%	89.5%	5.3%	3	42	63
⚒ Q145	What factors affect the performance of DevOps teams and the quality of software code with regard to quantity and quality of environments?	PROD	60.0%	100.0%	0.0%	4	1	68
⚒ Q114	What factors affect High Performance Teams?	BEST	58.8%	82.4%	0.0%	5	75	68
★ Q154	What factors affect understandability and readability of software code for other developers?	DP	58.3%	91.7%	8.3%	6	25	44
⚒ Q76	How can we improve the deployment process in DevOps teams?	PROC	56.3%	93.8%	0.0%	7	15	68
★ Q36	How does the effort spend on fixing vulnerabilities and bugs relate to effort spend on writing software correctly from the start?	BUG	56.3%	93.8%	0.0%	7	15	68
⚒ Q53	How does a continuous delivery pipeline with automated testing and migrating, including rollback facilities affect the performance of DevOps teams and the quality of software?	PROC	56.3%	93.8%	0.0%	7	15	68
★ Q22	How can requirements be validated before starting actual software development?	CR	55.6%	88.9%	0.0%	10	44	68
★ Q123	What factors affect performance testing on high data volumes?	TP	55.6%	88.9%	0.0%	10	44	68
👤 Q58	How to measure the customer value of a software product?	CR	55.6%	77.8%	11.1%	10	95	20
★ Q88	To what extent does testability affect the quality of software code?	EQ	52.9%	100.0%	0.0%	14	1	68
⚒ Q67	To what extent do automated checks of coding conventions, code quality, code complexity, and test-coverage affect the quality of software systems and the performance of DevOps teams?	EQ	47.1%	100.0%	0.0%	25	1	68
★ Q11	How can a system for (semi) automated CRUD test data generation improve delivery of software solutions?	TP	44.4%	100.0%	0.0%	32	1	68
⚒ Q104	What aspects affect the performance of DevOps teams and the quality of software with regard to software architecture?	PROD	40.0%	100.0%	0.0%	44	1	68
★ Q19	How can editors help software developers to document their public functions in a way that it is available for other developers?	CR	33.3%	100.0%	0.0%	73	1	68
★ Q122	What factors affect maintainability of software systems?	EQ	33.3%	100.0%	0.0%	73	1	68
★ Q80	How do automated controls within the continuous delivery pipeline affect the effort spent on risk and security?	DP	50.0%	95.0%	0.0%	15	8	68

Table is sorted on Rank Essential. The icon 👤 indicates customer related questions, ⚒ indicates questions that focus on the engineer and the effects of software development practices and processes on her work, and ★ indicates quality related questions.

Only two out of the 336 questions in the initial survey mentioned the word "customer" and only one of those questions made it to the top-20 (Q58 "How to measure the customer value of a software product" at rank 10 "Essential"). This question is, in line with the Microsoft study, marked with icon 👤, in Table 5.

Another eight "Essential" or "Worthwhile or higher" questions in the Microsoft study (marked with icon ⚒) focus on the engineer and the effects of software development practices and processes on her work. In our study, we identified nine questions with this icon. In addition to the focus on individual engineer, many of the questions in our study relates to the concept of the DevOps team. Overall, it seems that Microsoft has a big focus on customer while ING emphasizes on the engineering team itself. Finally, seven questions in the Microsoft study (marked with the icon ★) were about quality-related issues (same as ours with eleven questions).

4.3 Bottom-Rated Questions

Table 6 shows the top 10 unwise questions. The most "Unwise" question (Q27) at ING is the use of domain-specific language for use by non-experts. In the Microsoft study, the top five "Unwise" questions were all about a fear that respondents had of being rated. This effect can be seen in our study too (two of the top ten unwise questions - Q161 and Q30 - relate to measuring the performance of individual engineers), but not nearly as strongly as in the Microsoft study. Respondents in our study are torn on this topic; Q161 and Q30 are ranked as "Unwise" by respectively 22.2% and 20.0% of the respondents, but also ranked as "Essential" by another group of 44.4% and 40.0% of the respondents. Also, it was interesting to see that measuring and benchmarking time to market of software solutions (Q38) is one of the top 10 unwise questions. It indicates resistance against comparing departments based on key performance indicators like the time to market.

4.4 Rating by Demographics

Table 7 shows essential questions for different disciplines (Developer, Tester, Project Management) and roles (Manager, Individual Contributor, Architect). The complete inventory of questions for "Worthwhile or higher" and "Unwise" responses is present in the appendix.

4.4.1 Discipline. Microsoft study showed tester as a specific discipline mainly interested in test suites, bugs, and product quality. We do not see the discipline "tester" in our study. This can be seen in Table 7 in which overall scores relating to "Test" are low and highest for "Development". Software engineers in the DevOps teams at ING consider themselves to be generic developers, and testing is an integrated part of the discipline "developer". Both developers and

Table 6: Questions with the highest "Unwise" percentages (opposition).

	Question	Category	Percentages Essential	Worthwhile+	Unwise	Rank Essential	Worthwhile+	Unwise
Q27	How can software solutions in one common language be developed in a way that it is applicable to every person, regardless of ones interest in software development?	CR	22.2%	55.6%	33.3%	121	152	1
Q39	How can Windows-server images be created in order to facilitate testing within a continuous delivery pipeline?	DP	9.1%	45.5%	27.3%	162	163	2
Q170	Why do many developers focus on the newest of the newest? Why don't they leave this to a small group in order to use time and effort more efficiently?	DP	21.1%	47.4%	26.3%	128	161	3
Q161	What makes a great coder? What aspects affect the performance of DevOps teams and the quality of software with regard to characteristics of an individual software engineer?	PROD	44.4%	66.7%	22.2%	32	128	4
Q134	What factors affect TFS (Team Foundation Services) - a Microsoft product that provides source code management - with regard to working with automated pipelines?	BEST	38.9%	72.2%	22.2%	54	118	4
Q30	How can the performance of individual software engineers be benchmarked internally ING and externally with other companies?	PROD	40.0%	50.0%	20.0%	44	157	6
Q77	To what extent does changing of requirements during development affect the delivery of software solutions?	PROC	12.5%	68.8%	18.8%	150	124	7
Q21	How can PL1 software code be converted to Cobol code, while maintaining readability of the code in order to simplify an application environment?	BEST	18.2%	36.4%	18.2%	140	169	8
Q38	How can we measure the time to market of software solutions delivered within a department at ING in order to benchmark the performance of that department against others.	DP	9.1%	54.5%	18.2%	162	155	8
Q149	What factors affect the use of machine learning in software development over a period of ten years?	DP	16.7%	66.7%	16.7%	143	128	10
Q28	How can the cost of data be identified, in order to sign a price tag to data?	DP	5.6%	50.0%	16.7%	168	157	10

Table is sorted on Rank Unwise.

Table 7: Statistically significant rating differences for the response "Essential" by professional background.

	Question	Category	Response	Discipline Dev	Test	PM
Q2	Are there practices of good software teams from the perspective of releasing software solutions into production?	PROC	Essential	**66.7%**	5.6 %	11.1%
Q21	How can PL1 software code be converted to Cobol code, while maintaining readability of the code in order to simplify an application environment?	BEST	Essential	**66.7%**	4.8 %	0.0%
Q28	How can the cost of data be identified, in order to sign a price tag to data?	DP	Essential	**72.7%**	0.0 %	0.0%
Q46	How do static code analysis tools such as Fortify and Sonar influence the quality of software engineering products?	BEST	Essential	**36.6%**	0.0 %	27.3%
Q88	*To what extent does testability affect the quality of software code?*	EQ	Essential	**68.4%**	0.0 %	0.0%
Q89	How does time spent - in terms of full-time versus part-time - of a Scrum master affect the delivery of software solutions?	PROC	Essential	**66.7%**	5.6 %	11.1%
Q95	To what extent do dependencies on other teams affect team performance?	TC	Essential	**68.4%**	0.0 %	0.0%
Q97	How does documentation during software maintenance affect delivery of software solutions?	TP	Essential	**50.0%**	0.0 %	0.0%
Q110	What factors affect data analytics with regard to the use of external sources - such as market research reports and follow market trends - and let individual teams handle their local evolution?	PROC	Essential	**66.7%**	5.6 %	11.1%
Q162	What methods are most effective in preventing security related vulnerabilities or bugs from being introduced in software code?	BUG	Essential	**68.4%**	0.0 %	0.0%

	Question	Category	Response	Current Role Manager	Individual	Architect
Q2	Are there practices of good software teams from the perspective of releasing software solutions into production?	PROC	Essential	41.4%	**44.8** %	6.9%
Q46	How do static code analysis tools such as Fortify and Sonar influence the quality of software engineering products?	BEST	Essential	**69.2%**	15.4 %	0.0%
Q97	How does documentation during software maintenance affect delivery of software solutions?	TP	Essential	10.0%	**60.0** %	20.0%
Q153	What factors affect trunk-based development - a source-control branching model, where developers collaborate on code in a single branch - with regard to quality of software code?	BEST	Essential	22.6%	**54.8** %	9.7%

The professional background with the highest rating is highlighted in **bold**. Questions that are also in Table 5 are shown in *italics*. The role "Manager" includes the responses for "Manager" and "Lead".

testers are for example significantly interested in the testability of software code, and the quality of software related to an agile way of working and working in DevOps teams. Other findings relate to developers being significantly interested in team performance, e.g.

regarding practices of good software teams from the perspective of releasing software into production, the use of data analytics to improve individual teams, and dependencies on other teams.

4.4.2 Role. More individual contributors (e.g. developers) than managers are interested in good practices for software teams to release software into production. More managers than individual contributors, on the other hand, are interested in how software can help realize new policies and changes in the way of working, the relationship between documentation and maintenance of software, and to what extent the use of static code analysis tools such as Fortify and Sonar can affect the quality of software.

4.5 Comparing ING and Microsoft Questions

A comparison of the top 15 words from each company (see Table 8) shows that a majority of the popular themes are the same (e.g., code, test, software, and quality). Subtle differences, however, exist relating to rank (words in italics do not make it to top-15 in another company) and to the usage of a word in the other company (underlined).

A subset of these differences can be attributed to differences in terminology. For instance, Microsoft uses terms like employee/employees and team/teams, while their equivalents at ING are team/squad and engineer. Apart from this, Microsoft questions focused more on bugs, cost, time, customers, and tools while ING employees talked about version, problem, systems, process, and impact.

Next, we inferred 24 themes from the clusters in the affinity diagram organically merging into three broad categories: relating to code (like understanding code, testing, quality), developers (individual and team productivity) and customers (note that while customers did not make it to the top-10 essential questions, they were important in the top-100). The 24 themes are automated testing, testing, understanding code, documentation, formal methods, code review, debugging, risk, refactoring, deployment, bug fixing, legacy, software quality, requirement, release, cloud, customer, estimation, team productivity, employer productivity, cost, team awareness,

Table 8: Top 15 words from questions at ING and Microsoft

Microsoft 2014		ING 2019	
Word	Count	Word	Count
code / coding	48 (19%)	testing / debugging	92 (14%)
test / tests / testing	39 (16%)	code / coding	87 (13%)
software	31 (13%)	software	76 (11%)
employee / employees	16 (6%)	team / squad	72 (11%)
quality	13 (5%)	development	62 (9%)
bugs	13 (5%)	*version / library*	39 (6%)
development	12 (5%)	data	37 (6%)
cost	11 (4%)	incident, issue, *problem*	36 (5%)
team / teams	11 (4%)	security / risk	34 (5%)
time	10 (4%)	*system / systems*	34 (5%)
customer / customers	9 (4%)	quality	34 (5%)
impact	9 (4%)	production	21 (3%)
productivity	9 (4%)	*engineer*	14 (2%)
project	9 (4%)	*process*	14 (2%)
tools	7 (3%)	*impact*	13 (2%)

Top 15 words (sorted on count) from Microsoft 2014 and ING 2019 study. Words in the top-15 of one company and not the other are printed in italic. Words in one list and not the other are underlined.

and agile working. Investigating each theme and category in detail, we noticed that despite minor differences in the individual questions (some questions are broad in one company and specific in another), largely the key questions remain the same. For instance, employees at both the companies find questions relating to team productivity and employee productivity important, and yet assessing and comparing individual employees is undesirable. There were, however, subtle differences. For instance, in the Microsoft study, we noticed a few questions eliciting the need for agile (vs. waterfall) as well as automated testing. In the ING study, however, we do not see such questions. Rather, we see questions relating to the functional aspects of agile and automated testing. Another subtle difference between the two companies is relating to code size. While not stated explicitly, from the nature of questions, it seems that the software teams at Microsoft are dealing with a large legacy codebase. This was reflected in questions relating to team awareness, code monitoring, backward compatibility, and refactoring. Such questions, however, did not occur in ING. Other than the above, we saw cloud-related questions appearing in the Microsoft study only, while deployment-related questions appeared in ING only. In a nutshell, the core software development challenges of ING are consistent with Microsoft. There are although some nuanced differences which relate to the evolution of software market in the last five years as well as differences in the characteristics of the two companies.

5 DISCUSSION

In this section, we discuss potential explanations for the differences in the list of questions found in our study compared to the Microsoft study. We saw questions eliciting the need of agile methods in the Microsoft study while at ING the questions related to functional aspects. Our hypothesis here is that in the last five years there has been a change in the market: while in 2014, the questions on the adoption of agile and automated testing were common, in 2019 agile and automated testing became the norm.

We noticed that many questions at Microsoft deal with the scale of legacy code while no such question appeared at ING. One potential explanation for the observation can be that software systems at ING are not of the same scale as Microsoft. Nonetheless, it remains a lesson that in the next 10 years, ING can also be dealing with the complexity of large code base as Microsoft is experiencing today.

Finally, some questions appeared in only one organization. We believe that these observations have something to do with the individual practices followed at Microsoft and ING. The deployment-related questions at ING might be a result of the adoption of continuous delivery as a service. Surprisingly, we did not see any finance-related questions in the ING study. ING is a finance-based company and we expected to see some issues relating to both finance and software appear. We noticed that employees often talked about security, but no real finance-related questions appear. One explanation for this observation can be that the data science challenges relating to software development are independent of the actual field to which it is applied. Supporting this argument, 145 questions from Microsoft also did not bring up any product specific details. Another potential explanation can be that through our question we anchored our respondents into asking software development related questions only.

5.1 Implications

One of the key findings of this paper is a list of 171 questions that software engineers in a large, software-driven organization would like to see answered, in order to optimize their software development activities. From this, we see implications both in terms of practice and industry.

From a practical perspective, our study offers a new way of thinking to software development organizations who care about their development processes. The questions originally raised by Microsoft are not just relevant to one of the largest tech companies in the world, but also to large software-defined enterprises active outside the tech-sector proper. Inspired by these questions, an organization may select the most relevant ones, and seek ways to address them. While some questions are fundamentally hard to answer, organizations can make a starting point by collecting relevant data about their development processes. This, then, can help to make the development process itself more and more data-driven. This is exactly how ING intends to use the questions, and we believe companies around the world can follow suit.

From a research perspective, we have seen that the original Microsoft study has generated a series of papers that apply some form of Machine Learning to address the challenges raised in that study. In the research community, *AI-for-Software-Engineering* is an increasingly important topic, with many papers appearing that seek to apply machine learning to address software engineering problems. Our study aims to add urgency and direction to this emerging field, by highlighting not just which questions *can* be answered, but which ones *should* be answered, from a practitioner perspective.

5.2 Threats to Validity

While our study expands the external validity of the original study, the fact remains that the two lists of questions are based on just two companies, which are both large organizations with over 10,000 software developers. Our study highlights relevance to the FinTech sector, but it would be interesting to see further replications, for example in the automotive or health care sector, with different regulatory and additional safety constraints. We expect that many of the questions are also relevant to smaller organizations, especially given the agile way of working at ING. Nevertheless, it will be worthwhile to further explore this.

From a perspective of internal validity, creating codes independent of the prior study is challenging. It is possible that the similarities and differences seen compared to the Microsoft study relates to factors (e.g. researcher bias) other than the actual data. We tried mitigating it by limiting our exposure to the previous study, not involving authors from the Microsoft study, and multiple authors generating codes independently. Nonetheless, these biases are likely to exist.

For reasons of replication, we have used where possible the same survey questions, method of analysis and division into work area and discipline as in the Microsoft study [7]. Apart from positive effects, this choice also had a negative effect with regard to analysis of demographics, mainly due to the fact that ING uses a different way of working, including corresponding roles and team structure, than within Microsoft. Especially mapping the professional background "Discipline" of the original study on the demographic "Discipline" as applied within ING was challenging.

ING works with DevOps teams, where an engineer fulfills both the area of developer and that of tester. As a result, testers were under-represented in both of the surveys we conducted. As a mitigation measure we therefore opted for combining the results of developers and testers in the further analysis.

Another potential threat is sensitivity of the ranks which mostly occurs at the extreme sides of the ranking, when, e.g., none of the participants label a question as 'Unwise'. In our study, on average each question received $21,888/177 = 123$ ratings and hence sensitivity of ranks is unlikely.

The presented results are free from corporate influence including Microsoft. A number of stakeholders at ING (CIO, Corporate Communications) reviewed the submitted paper and approved it without any changes. Nevertheless, self-censorship by the authors remains a potential threat. Furthermore, researchers may have their biases which can potentially influence the results.

As also emphasized in related work on replications [77] [68] [36] [43] [57] [26], our study seeks to replicate earlier findings in a different context (e.g. other companies) and during a different time (environments and perceptions of engineers do change over time). In order to facilitate future replication of our study, we make the total set of descriptive questions and additional info on results of our tests available in our technical report.

6 CONCLUSION

Conducted at ING—a software-defined enterprise providing banking solutions—this study presents 171 questions that software engineers at ING would like data scientists to answer. This study is a replication of a similar study at software company Microsoft, which resulted in 145 questions for data scientists. Further, we went a step beyond to investigate the applicability of Microsoft's questions in ING, as well as changes in trends over the last five years.

We compared the two lists of questions and found that the core software development challenges (relating to code, developer, and customer) remain the same. Nonetheless, we observed subtle differences relating to the technology and software process developments (e.g., currently the debate about agile versus waterfall is now largely absent) and differences in the two organizations (e.g., Microsoft's focus on solving problems with a large code bases and ING's challenges with continuous deployment). We complete our analysis with a report on the impact Microsoft 2014 study generated, also indicating the impact that our study is capable to generate.

A thorough understanding of key questions software engineers have that can be answered by data scientists is of crucial importance to both the research community and modern software engineering practice. Our study aims to contribute to this understanding. We call on other companies, large and small, to conduct a similar analysis, in order to transform a software engineering into a data-driven endeavour addressing the most pressing questions.

ACKNOWLEDGMENTS

The authors thank ING and all participant engineers for sharing their experience and views with us. We thank the authors of the original Microsoft study for the permission to reuse their research design figure.

REFERENCES

[1] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 291–300.

[2] Oana Andrei, Muffy Calder, Matthew Chalmers, Alistair Morrison, and Mattias Rost. 2016. Probabilistic formal analysis of app usage to inform redesign. In *International Conference on Integrated Formal Methods*. Springer, 115–129.

[3] Timothy Arndt. 2018. Big Data and software engineering: prospects for mutual enrichment. *Iran Journal of Computer Science* 1, 1 (2018), 3–10.

[4] Anonymous Authors. 2019. *Technical Report anonymized*. Technical Report.

[5] Ali Basiri, Lorin Hochstein, Nora Jones, and Haley Tucker. 2019. Automating chaos experiments in production. *Proceedings of the 41st ACM/IEEE International Conference on Software Engineering (ICSE)* (2019).

[6] Andrea Batch and Niklas Elmqvist. 2017. The interactive visualization gap in initial exploratory data analysis. *IEEE transactions on visualization and computer graphics* 24, 1 (2017), 278–287.

[7] Andrew Begel and Thomas Zimmermann. 2014. Analyze This! 145 Questions for Data Scientists in Software Engineering. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*. ACM, New York, NY, USA, 12–23. https://doi.org/10.1145/2568225.2568233

[8] Moritz Beller, Georgios Gousios, Annibale Panichella, Sebastian Proksch, Sven Amann, and Andy Zaidman. 2017. Developer testing in the ide: Patterns, beliefs, and behavior. *IEEE Transactions on Software Engineering* 45, 3 (2017), 261–284.

[9] Moritz Beller, Georgios Gousios, Annibale Panichella, and Andy Zaidman. 2015. When, how, and why developers (do not) test in their IDEs. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ACM, 179–190.

[10] Moritz Beller, Georgios Gousios, and Andy Zaidman. 2015. How (much) do developers test?. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 2. IEEE, 559–562.

[11] Christian Bird, Tim Menzies, and Thomas Zimmermann. 2015. Chapter 1 - Past, Present, and Future of Analyzing Software Data. In *The Art and Science of Analyzing Software Data*, Christian Bird, Tim Menzies, and Thomas Zimmermann (Eds.). Morgan Kaufmann, Boston, 1 – 13. https://doi.org/10.1016/B978-0-12-411519-4.00001-X

[12] Bruno Cartaxo, Gustavo Pinto, and Sergio Soares. 2018. The role of rapid reviews in supporting decision-making in software engineering practice. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*. 24–34.

[13] Jeffrey C Carver, Oscar Dieste, Nicholas A Kraft, David Lo, and Thomas Zimmermann. 2016. How practitioners perceive the relevance of esem research. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM, 56.

[14] Di Chen, Wei Fu, Rahul Krishna, and Tim Menzies. 2018. Applications of psychological science for actionable analytics. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 456–467.

[15] Jacek Dąbrowski, Emmanuel Letier, Anna Perini, and Angelo Susi. 2019. Finding and analyzing app reviews related to specific features: A research preview. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 183–189.

[16] Boonyarit Deewattananon and Usa Sammapun. 2017. Analyzing user reviews in Thai language toward aspects in mobile applications. In *2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. IEEE, 1–6.

[17] Paul Denny, Brett A Becker, Michelle Craig, Greg Wilson, and Piotr Banaszkiewicz. 2019. Research This! Questions That Computing Educators Most Want Computing Education Researchers to Answer. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. 259–267.

[18] P. Devanbu, T. Zimmermann, and C. Bird. 2016. Belief Evidence in Empirical Software Engineering. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. 108–119. https://doi.org/10.1145/2884781.2884812

[19] Denae Ford and Chris Parnin. 2015. Exploring causes of frustration for software developers. In *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering*. IEEE, 115–116.

[20] Denae Ford, Justin Smith, Philip J Guo, and Chris Parnin. 2016. Paradise unplugged: Identifying barriers for female participation on stack overflow. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 846–857.

[21] Wei Fu and Tim Menzies. 2017. Easy over hard: A case study on deep learning. In *Proceedings of the 2017 11th joint meeting on foundations of software engineering*. 49–60.

[22] Vahid Garousi, Ahmet Coşkunçay, Aysu Betin-Can, and Onur Demirörs. 2015. A survey of software engineering practices in Turkey. *Journal of Systems and Software* 108 (2015), 148–177.

[23] Vahid Garousi and Michael Felderer. 2017. Worlds apart: industrial and academic focus areas in software testing. *IEEE Software* 34, 5 (2017), 38–45.

[24] Vahid Garousi and Kadir Herkiloglu. 2016. Selecting the right topics for industry-academia collaborations in software testing: an experience report. In *2016 IEEE International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 213–222.

[25] Daniel German, Gregorio Robles, Germán Poo-Caamaño, Xin Yang, Hajimu Iida, and Katsuro Inoue. 2018. " Was My Contribution Fairly Reviewed?" A Framework to Study the Perception of Fairness in Modern Code Reviews. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*. IEEE, 523–534.

[26] Omar S. Gómez, Natalia Juristo, and Sira Vegas. 2014. Understanding replication of experiments in software engineering: A classification. *Information and Software Technology* 56, 8 (2014), 1033 – 1048. https://doi.org/10.1016/j.infsof.2014.04.004

[27] Georgios Gousios, Dominik Safaric, and Joost Visser. 2016. Streaming software analytics. In *2016 IEEE/ACM 2nd International Workshop on Big Data Software Engineering (BIGDSE)*. IEEE, 8–11.

[28] Xiaodong Gu and Sunghun Kim. 2015. " What Parts of Your Apps are Loved by Users?"(T). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 760–770.

[29] Monika Gupta, Ashish Sureka, Srinivas Padmanabhuni, and Allahbaksh Mohammedali Asadullah. 2015. Identifying Software Process Management Challenges: Survey of Practitioners in a Large Global IT Company. In *Proceedings of the 12th Working Conference on Mining Software Repositories (MSR '15)*. IEEE Press, Piscataway, NJ, USA, 346–356.

[30] Emitza Guzman, Muhammad El-Haliby, and Bernd Bruegge. 2015. Ensemble methods for app review classification: An approach for software evolution (n). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 771–776.

[31] Sebastian Hahn, Matthias Trapp, Nikolai Wuttke, and Jürgen Döllner. 2015. Thread City: Combined Visualization of Structure and Activity for the Exploration of Multi-threaded Software Systems. In *2015 19th International Conference on Information Visualisation*. IEEE, 101–106.

[32] Ahmed E Hassan and Tao Xie. 2010. Software intelligence: the future of mining software engineering data. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*. ACM, 161–166.

[33] Michael Hilton, Nicholas Nelson, Hugh McDonald, Sean McDonald, Ron Metoyer, and Danny Dig. 2016. Tddviz: Using software changes to understand conformance to test driven development. In *International Conference on Agile Software Development*. Springer, Cham, 53–65.

[34] Geert Hofstede and Michael H. Bond. 1984. Hofstede's Culture Dimensions: An Independent Validation Using Rokeach's Value Survey. *Journal of Cross-Cultural Psychology* 15, 4 (1984), 417–433. https://doi.org/10.1177/0022002184015004003 arXiv:https://doi.org/10.1177/0022002184015004003

[35] Yury Izrailevsky and Ariel Tseitlin. 2011. The Netflix Simian Army. *Netflix Technology Blog* (2011). https://medium.com/netflix-techblog/the-netflix-simian-army-16e57fbab116

[36] Natalia Juristo and Omar S. Gómez. 2012. *Replication of Software Engineering Experiments*. Springer Berlin Heidelberg, Berlin, Heidelberg, 60–88. https://doi.org/10.1007/978-3-642-25231-0_2

[37] Shaikh Jeeshan Kabeer, Maleknaz Nayebi, Guenther Ruhe, Chris Carlson, and Francis Chew. 2017. Predicting the vector impact of change-an industrial case study at brightsquid. In *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 131–140.

[38] Verena Käfer. 2017. Summarizing software engineering communication artifacts from different sources. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. 1038–1041.

[39] Noriaki Kano, Nobuhiko Seraku, Fumio Takahashi, and Shin ichi Tsuji. 1984. Attractive Quality and Must-Be Quality. *Journal of the Japanese Society for Quality Control* 14 (1984), 39–48.

[40] Foutse Khomh, Bram Adams, Jinghui Cheng, Marios Fokaefs, and Giuliano Antoniol. 2018. Software engineering for machine-learning applications: The road ahead. *IEEE Software* 35, 5 (2018), 81–84.

[41] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. 2016. The Emerging Role of Data Scientists on Software Development Teams. In *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*. ACM, New York, NY, USA, 96–107. https://doi.org/10.1145/2884781.2884783

[42] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. 2016. The emerging role of data scientists on software development teams. In *Proceedings of the 38th International Conference on Software Engineering*. ACM, 96–107.

[43] Barbara Kitchenham. 2008. The role of replications in empirical software engineering - a word of warning. *Empirical Software Engineering* 13, 2 (2008), 219–221.

https://doi.org/10.1007/s10664-008-9061-0

[44] B. Kitchenham and S. Pfleeger. 2008. Personal Opinion Surveys. *Guide to Advanced Empirical Software Engineering* (2008).

[45] Pavneet Singh Kochhar, Xin Xia, David Lo, and Shanping Li. 2016. Practitioners' expectations on automated fault localization. In *Proceedings of the 25th International Symposium on Software Testing and Analysis*. ACM, 165–176.

[46] Oleksii Kononenko, Olga Baysal, and Michael W Godfrey. 2016. Code review quality: how developers see it. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. IEEE, 1028–1038.

[47] Rahul Krishna. 2017. Learning effective changes for software projects. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 1002–1005.

[48] Rahul Krishna and Tim Menzies. 2018. Bellwethers: A baseline method for transfer learning. *IEEE Transactions on Software Engineering* 45, 11 (2018), 1081–1105.

[49] Philipp Leitner, Jürgen Cito, and Emanuel Stöckli. 2016. Modelling and managing deployment costs of microservice-based cloud applications. In *Proceedings of the 9th International Conference on Utility and Cloud Computing*. ACM, 165–174.

[50] Paul Luo Li, Andrew J Ko, and Jiamin Zhu. 2015. What makes a great software engineer?. In *Proceedings of the 37th International Conference on Software Engineering-Volume 1*. IEEE Press, 700–710.

[51] David Lo, Nachiappan Nagappan, and Thomas Zimmermann. 2015. How practitioners perceive the relevance of software engineering research. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ACM, 415–425.

[52] Cuauhtemoc Lopez-Martin, Arturo Chavoya, and Maria Elena Meda-Campaña. 2014. A machine learning technique for predicting the productivity of practitioners from individually developed software projects. In *15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. IEEE, 1–6.

[53] Lucy Ellen Lwakatare, Aiswarya Raj, Jan Bosch, Helena Holmström Olsson, and Ivica Crnkovic. 2019. A taxonomy of software engineering challenges for machine learning systems: An empirical investigation. In *International Conference on Agile Software Development*. Springer, 227–243.

[54] Björn Mathis, Vitalii Avdiienko, Ezekiel O Soremekun, Marcel Böhme, and Andreas Zeller. 2017. Detecting information flow by mutating input data. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 263–273.

[55] Tim Menzies, Ekrem Kocaguneli, Burak Turhan, Leandro Minku, and Fayola Peters. 2014. *Sharing data and models in software engineering*. Morgan Kaufmann.

[56] Tim Menzies and Thomas Zimmermann. 2018. Software Analytics: What's Next? *IEEE Software* 35, 5 (2018), 64–70.

[57] James Miller. 2005. Replicating software engineering experiments: a poisoned chalice or the Holy Grail. *Information and Software Technology* 47, 4 (2005), 233 – 244. https://doi.org/10.1016/j.infsof.2004.08.005

[58] Ayse Tosun Misirli, Hakan Erdogmus, Natalia Juristo, and Oscar Dieste. 2014. Topic selection in industry experiments. In *Proceedings of the 2nd International Workshop on Conducting Empirical Studies in Industry*. 25–30.

[59] Maleknaz Nayebi, Yuanfang Cai, Rick Kazman, Guenther Ruhe, Qiong Feng, Chris Carlson, and Francis Chew. 2019. A longitudinal study of identifying and paying down architecture debt. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 171–180.

[60] Maleknaz Nayebi, Konstantin Kuznetsov, Paul Chen, Andreas Zeller, and Guenther Ruhe. 2018. Anatomy of functionality deletion: an exploratory study on mobile apps. In *Proceedings of the 15th International Conference on Mining Software Repositories*. 243–253.

[61] Maleknaz Nayebi, Mahshid Marbouti, Rache Quapp, Frank Maurer, and Guenther Ruhe. 2017. Crowdsourced exploration of mobile app features: A case study of the fort mcmurray wildfire. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Society Track (ICSE-SEIS)*. IEEE, 57–66.

[62] Tobias Roehm. 2015. Two user perspectives in program comprehension: end users and developer users. In *2015 IEEE 23rd International Conference on Program Comprehension*. IEEE, 129–139.

[63] Saurabh Sarkar and Chris Parnin. 2017. Characterizing and predicting mental fatigue during programming tasks. In *2017 IEEE/ACM 2nd International Workshop on Emotion Awareness in Software Engineering (SEmotion)*. IEEE, 32–37.

[64] Anand Ashok Sawant, Romain Robbes, and Alberto Bacchelli. 2016. On the reaction to deprecation of 25,357 clients of 4+ 1 popular Java APIs. In *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 400–410.

[65] Anand Ashok Sawant, Romain Robbes, and Alberto Bacchelli. 2018. On the reaction to deprecation of clients of 4+ 1 popular Java APIs and the JDK. *Empirical Software Engineering* 23, 4 (2018), 2158–2197.

[66] Gerald Schermann, Dominik Schöni, Philipp Leitner, and Harald C Gall. 2016. Bifrost: Supporting continuous deployment with automated enactment of multi-phase live testing strategies. In *Proceedings of the 17th International Middleware Conference*. 1–14.

[67] Vibhu Saujanya Sharma, Rohit Mehra, and Vikrant Kaulgud. 2017. What do developers want? an advisor approach for developer priorities. In *2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 78–81.

[68] Forrest J. Shull, Jeffrey C. Carver, Sira Vegas, and Natalia Juristo. 2008. The role of replications in Empirical Software Engineering. *Empirical Software Engineering* 13, 2 (01 Apr 2008), 211–218. https://doi.org/10.1007/s10664-008-9060-1

[69] Margaret-Anne Storey, Alexey Zagalsky, Fernando Figueira Filho, Leif Singer, and Daniel M German. 2016. How social and communication channels shape and challenge a participatory culture in software development. *IEEE Transactions on Software Engineering* 43, 2 (2016), 185–204.

[70] Sampo Suonsyrjä, Laura Hokkanen, Henri Terho, Kari Systä, and Tommi Mikkonen. 2016. Post-deployment data: A recipe for satisfying knowledge needs in software development?. In *2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA)*. IEEE, 139–147.

[71] Sampo Suonsyrjä and Tommi Mikkonen. 2015. Designing an unobtrusive analytics framework for monitoring java applications. In *Software Measurement*. Springer, 160–175.

[72] Mohammadali Tavakoli, Liping Zhao, Atefeh Heydari, and Goran Nenadić. 2018. Extracting useful software development information from mobile application reviews: A survey of intelligent mining techniques and tools. *Expert Systems with Applications* 113 (2018), 186–199.

[73] Ambika Tripathi, Savita Dabral, and Ashish Sureka. 2015. University-industry collaboration and open source software (oss) dataset in mining software repositories (msr) research. In *2015 IEEE 1st International Workshop on Software Analytics (SWAN)*. IEEE, 39–40.

[74] Carmine Vassallo, Fiorella Zampetti, Daniele Romano, Moritz Beller, Annibale Panichella, Massimiliano Di Penta, and Andy Zaidman. 2016. Continuous delivery practices in a large financial organization. In *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 519–528.

[75] Zhiyuan Wan, Xin Xia, Ahmed E Hassan, David Lo, Jianwei Yin, and Xiaohu Yang. 2018. Perceptions, expectations, and challenges in defect prediction. *IEEE Transactions on Software Engineering* (2018).

[76] David Gray Widder, Michael Hilton, Christian Kästner, and Bogdan Vasilescu. 2019. A conceptual replication of continuous integration pain points in the context of Travis CI. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 647–658.

[77] Claes Wohlin. 2014. Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE '14)*. ACM, New York, NY, USA, Article 38, 10 pages. https://doi.org/10.1145/2601248.2601268

[78] Fiorella Zampetti, Simone Scalabrino, Rocco Oliveto, Gerardo Canfora, and Massimiliano Di Penta. 2017. How open source projects use static code analysis tools in continuous integration pipelines. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE, 334–344.

[79] Thomas Zimmermann. 2017. Software Productivity Decoded: How Data Science Helps to Achieve More (Keynote). In *Proceedings of the 2017 International Conference on Software and System Process (ICSSP 2017)*. ACM, New York, NY, USA, 1–2. https://doi.org/10.1145/3084100.3087674

[80] Weiqin Zou, David Lo, Zhenyu Chen, Xin Xia, Yang Feng, and Baowen Xu. 2018. How practitioners perceive automated bug report management techniques. *IEEE Transactions on Software Engineering* (2018).

APPENDIX

This appendix contains methodological and statistical supplements for the paper 'Questions for Data Scientists in Software Engineering'. The report gives in Table 9 and overview of the mapping of categories from the ING 2019 study on the categories in the original Microsoft 2014 study. Table 10 includes an overview of all 171 descriptive questions that resulted from the initial survey, and that were used for ranking purposes in the ranking survey. The table is sorted on Percentages Worthwhile+ and Percentages Essential. Furthermore, Table 11 gives an overview of statistically significant rating differences by demographics.

Table 9: Category mapping.

Microsoft Category	Descriptive Questions	ING Category	Descriptive Questions
Development Practices	28	Estimation Architecture Knowledge sharing Dependencies	8 7 3 3
Testing Practices	20	Testing	13
Evaluating Quality	16	Code analysis Quality Effectiveness	5 5 1
Software Development Process	14	Development processes Technology selection Deployment	17 3 2
Productivity	13	Productivity Employee evaluation	8 6
Teams and Collaboration	11	Team composition Knowledge sharing	3 2
Customers and Requirements	9	Formal methods Customer value	4 3
Development Best Practices	9	Best practices Technology selection Effectiveness Tools	11 7 7 5
Services	8	Performance	1
Bug Measurements	7	Security	3
Software Development Lifecycle	7	Effectiveness Management Performance Security	1 1 1 1
Reuse and Shared Components	3	Awareness Reuse	1 1

Table 10: Overview of all descriptive questions.

	Question	Category	Percentages Essential	Percentages Worthwhile+	Percentages Unwise	Rank Essential	Rank Worthwhile+	Rank Unwise
Q143	What factors affect the performance and productivity of DevOps teams with regard to team happiness and pleasure in your work?	DP	68.4%	94.7%	0.0%	1	9	68
Q98	To what extent does on-demand provisioning of develop- and test environments, including up-to-date data affect delivery of software solutions?	TP	66,7%	77,8%	0,0%	2	95	68
Q37	How can we make sure that we build for re-usability and scalability?	BEST	63,2%	89,5%	5,3%	3	42	63
Q145	What factors affect the performance of DevOps teams and the quality of software code with regard to quantity and quality of environments?	PROD	60,0%	100,0%	0,0%	4	1	68
Q114	What factors affect High Performance Teams?	BEST	58,8%	82,4%	0,0%	5	75	68
Q154	What factors affect understand-ability and readability of software code for other developers?	DP	58,3%	91,7%	8,3%	6	25	44
Q76	To what extent affects building software solutions by using a continuous delivery pipeline with automated testing and migrating, including rollback facilities the performance of DevOps teams and the quality of software?	PROC	56,3%	93,8%	0,0%	7	15	68
Q36	How can we improve the deployment process in DevOps teams?	PROC	56,3%	93,8%	0,0%	7	15	68
Q53	How does the effort spent on fixing vulnerabilities and bugs relate to effort spent on writing software correctly from the start?	BUG	56,3%	93,8%	0,0%	7	15	68
Q22	How can requirements be validated before starting actual software development?	CR	55,6%	88,9%	0,0%	10	44	68
Q123	What factors affect performance testing on high data volumes?	TP	55,6%	88,9%	0,0%	10	44	68
Q58	How to measure the customer value of a software product?	CR	55,6%	77,8%	11,1%	10	95	20
Q163	What methods can be applied to analyze whether software code is working as expected?	EQ	53,3%	93,3%	0,0%	13	19	68
Q88	To what extent affects the test ability of software code the quality of code?	EQ	52,9%	100,0%	0,0%	14	1	68
Q80	To what extent affects implementing automated controls within the continuous delivery pipeline the effort spent on accountability with regard to risks and security?	DP	50,0%	95,0%	0,0%	15	8	68
Q125	What factors affect providing new technologies to consumers, and can implementations of new technology be internally and externally benchmarked?	SL	50,0%	90,0%	0,0%	15	30	68
Q109	What factors affect creating and maintaining software solutions yourself, versus using off-the-shelve solutions?	BEST	50,0%	90,0%	0,0%	15	30	68
Q34	How can user feedback be integrated in an efficient and effective way into software code?	CR	50,0%	87,5%	12,5%	15	56	15
Q1	Are developers working in an open space with several teams more effective or less than developers working in a room with just their team?	PROD	50,0%	83,3%	8,3%	15	73	44
Q140	What factors affect the level of competence - such as novice, advanced, competent - of software engineers in order to gain insight into tasks, time spent, and development goals per competence level.	DP	50,0%	75,0%	8,3%	15	106	44

	Question	Category	Percentages Essential	Percentages Worthwhile+	Percentages Unwise	Rank Essential	Rank Worthwhile+	Rank Unwise
Q159	What is the relation between frameworks used and programming languages one hand and time of development on the other?	DP	50,0%	75,0%	0,0%	15	106	68
Q8	Does a focus on time and effort affects the quality of software systems in terms of availability, reliability, and security?	RSC	50,0%	70,0%	0,0%	15	122	68
Q33	How can toolsets be organized into distributable packages, in order to support installation of components by software developers?	BEST	47,4%	84,2%	0,0%	23	70	68
Q166	When do you remove an old module that you think is not being used anymore?	BEST	47,4%	73,7%	0,0%	23	113	68
Q67	To what extent affects automated checks of coding conventions, code quality, code complexity, and test-coverage the quality of software systems and the performance of DevOps teams?	EQ	47,1%	100,0%	0,0%	25	1	68
Q82	To what extent affects peer review or code review the quality of code?	EQ	47,1%	94,1%	0,0%	25	12	68
Q75	To what extent affects automation of deployment scripts, with regard to single use versus re-use, the delivery of software solutions?	PROC	46,7%	86,7%	0,0%	27	59	68
Q162	What methods are most effective in preventing security related vulnerabilities or bugs from being introduced in software code?	BUG	46,7%	86,7%	0,0%	27	59	68
Q7	Does a focus on quick release of features and bug fixes into production help to achieve confidence and agility?	PROC	46,7%	80,0%	6,7%	27	84	51
Q141	What factors affect the performance and productivity of DevOps teams with regard to evidence-based decision-making versus decision-making based on expert opinions.	DP	45,5%	90,9%	0,0%	30	26	68
Q32	How can the productivity of engineers and squads be measured?	PROD	45,5%	54,5%	9,1%	30	155	36
Q11	How can a system for (semi) automated CRUD test data generation improve delivery of software solutions?	TP	44,4%	100,0%	0,0%	32	1	68
Q41	How do customers perceive software interfaces with regard to clarity and fitness of a solution?	CR	44,4%	88,9%	11,1%	32	44	20
Q73	To what extent affects a test coverage approach, including a specific threshold, the delivery and quality of software solutions?	TP	44,4%	77,8%	0,0%	32	95	68
Q118	What factors affect integration testing of software solutions?	TP	44,4%	77,8%	0,0%	32	95	68
Q161	What makes a great coder? What aspects affect the performance of DevOps teams and the quality of software with regard to characteristics of an individual software engineer?	PROD	44,4%	66,7%	22,2%	32	128	4
Q111	What factors affect debugging of complex data issues in production?	TP	44,4%	66,7%	0,0%	32	128	68
Q64	Is the infrastructure for development and in the OTAP flexible and modern enough to ease and speed up development?	PROC	43,8%	81,3%	12,5%	38	82	15
Q92	To what extent can an approach such as Chaos Monkey, where virtual machine instances and containers that run in the production environment are randomly terminated, help to build more resilient software services?	TP	42,9%	85,7%	0,0%	39	64	68
Q14	How can an inventory of tips and tricks for software development help software engineers to develop software?	BEST	42,1%	94,7%	0,0%	40	9	68
Q120	What factors affect limitation of access by engineers to production systems versus effectiveness of engineers?	BEST	42,1%	78,9%	10,5%	40	93	29
Q95	To what extent do dependencies on other teams affect team performance?	TC	41,2%	94,1%	0,0%	42	12	68
Q121	What factors affect loading high volume data with regard to availability of tools in industry?	BEST	41,2%	76,5%	0,0%	42	105	68
Q104	What aspects affects the performance of DevOps teams and the quality of software with regard to software architecture?	PROD	40,0%	100,0%	0,0%	44	1	68
Q164	What parts of a software system affect software quality most, with regard to aspects such as change frequency, bug severity, hit ratio of paths, and importance of code?	EQ	40,0%	93,3%	0,0%	44	19	68

			Percentages			Rank		
	Question	Category	Essential	Worthwhile+	Unwise	Essential	Worthwhile+	Unwise
Q47	How do the number and severity of security incidents relate to platforms or types of software such as Linux, Windows, Mainframe, CDaaS, non-CDaaS, TFS, BOTS, SAAS, and inhouse development?	BUG	40,0%	93,3%	0,0%	44	19	68
Q48	How does automated testing as part of the design and build process influence the time of delivery of software products?	BEST	40,0%	90,0%	0,0%	44	30	68
Q70	To what extent affect the requirements of a laptop the performance of DevOps teams and the quality of software?	PROD	40,0%	90,0%	0,0%	44	30	68
Q113	What factors affect estimation of lead time, cost, and quality of software deliveries?	SL	40,0%	80,0%	0,0%	44	84	68
Q85	To what extent affects security by design - meaning that the software has been designed from the foundation to be secure - the delivery of software solutions?	SL	40,0%	80,0%	0,0%	44	84	68
Q133	What factors affect testing and fixing bugs during a sprint versus testing and fixing bugs just before release (after several sprints)?	BEST	40,0%	80,0%	0,0%	44	84	68
Q168	Which input data is in the end result worth say ten times more than other?	TP	40,0%	80,0%	0,0%	44	84	68
Q30	How can the performance of individual software engineers be benchmarked internally ING and externally with other companies?	PROD	40,0%	50,0%	20,0%	44	157	6
Q131	What factors affect team on-boarding on ING frameworks with regard to speed of development?	DP	38,9%	88,9%	0,0%	54	44	68
Q12	How can an inventory of best practices for analytical solutions in software development help software engineers to develop software?	BEST	38,9%	88,9%	0,0%	54	44	68
Q165	What properties affect the quality of tooling for backlog management, development, such as tools for Configuration Items, issue tracking, logging, monitoring, and crash reporting?	BEST	38,9%	88,9%	0,0%	54	44	68
Q153	What factors affect trunk-based development - a source-control branching model, where developers collaborate on code in a single branch - with regard to quality of software code?	BEST	38,9%	83,3%	5,6%	54	73	61
Q134	What factors affect TFS (Team Foundation Services) - a Microsoft product that provides source code management - with regard to working with automated pipelines?	BEST	38,9%	72,2%	22,2%	54	118	4
Q63	Is it more expensive to develop front-end applications using 'modern' JavaScript frameworks, like Angular, Polymer, React than frameworks like Vaadin, GWT, Icefaces and maybe some others around?	BEST	38,5%	84,6%	7,7%	59	69	47
Q71	To what extent affects a checklist before pushing code into production improve performance and quality of a software system?	DP	38,1%	85,7%	0,0%	60	64	68
Q52	How does team maturity affect code quality and incidents?	TC	37,5%	93,8%	0,0%	61	15	68
Q142	What factors affect the performance and productivity of DevOps teams with regard to simultaneous slow and fast developments at the same time in same environments?	DP	37,5%	87,5%	0,0%	61	56	68
Q139	What factors affect the different software engineering approaches such as Scrum and Kanban in practical settings?	PROC	37,5%	68,8%	12,5%	61	124	15
Q127	What factors affect replication of production data into a test environment in order to be able to test migrations and releases on an exact copy of the production environment?	DP	36,4%	90,9%	9,1%	64	26	36
Q115	What factors affect individual software engineer productivity with regard to facilitation by others?	PROD	36,4%	90,9%	9,1%	64	26	36
Q116	What factors affect individual software engineer productivity with regard to knowledge and skills?	PROD	36,4%	81,8%	9,1%	64	77	36
Q45	How do software engineers deal with pressure from management or stakeholders?	PROD	36,4%	81,8%	0,0%	64	77	68
Q147	What factors affect the requirements for a version control system in order to test for fitness within the ING environment?	DP	36,4%	72,7%	0,0%	64	115	68
Q59	How valuable would it be to have architects that are more busy with helping other developer and not being fully busy with own development?	PROD	36,4%	72,7%	0,0%	64	115	68
Q135	What factors affect the composition of DevOps teams?	TC	35,7%	92,9%	0,0%	70	24	68

	Question	Category	Percentages Essential	Percentages Worthwhile+	Percentages Unwise	Rank Essential	Rank Worthwhile+	Rank Unwise
Q96	To what extent do standardization for solution development affect team performance?	TC	35,3%	94,1%	0,0%	71	12	68
Q94	To what extent do dependencies of people, people changing roles or people leaving a team affect team performance?	TC	35,3%	64,7%	5,9%	71	138	59
Q19	How can editors help software developers to document their public functions in a way that it is available for other developers?	CR	33,3%	100,0%	0,0%	73	1	68
Q122	What factors affect maintainability of software systems?	EQ	33,3%	100,0%	0,0%	73	1	68
Q10	How are deadlines be handled within the scope of an agile way of working?	PROC	33,3%	88,9%	0,0%	73	44	68
Q119	What factors affect lead time of software deliveries, with regard to idea creation, design, development, test, deploy, and release?	TP	33,3%	88,9%	11,1%	73	44	20
Q137	What factors affect the delivery of software solutions - including aspects such as architecture, modularization, and distributed components - with regard to collaboration of teams located in different countries?	PROC	33,3%	86,7%	6,7%	73	59	51
Q90	To what extent affects the upfront preparation of a design the delivery of software solutions?	PROC	33,3%	86,7%	0,0%	73	59	68
Q2	Are there practices of good software teams from the perspective of releasing software solutions into production?	PROC	33,3%	80,0%	0,0%	73	84	68
Q107	What factors affect adopting a micro-service architecture - an architectural style that structures an application as a collection of loosely coupled services, which implement business capabilities - for software engineering purposes?	DP	33,3%	77,8%	5,6%	73	95	61
Q18	How can descriptive statistics such as averages, minimum, and maximum of data-sets in use be measured in order to scale the processing of data?	DP	33,3%	77,8%	0,0%	73	95	68
Q146	What factors affect the quality of the CDaaS solution for continuous delivery of software, and how can improvements be realized?	PROC	33,3%	77,8%	0,0%	73	95	68
Q50	How does involvement of business stakeholders affect the delivery of software solutions?	TP	33,3%	77,8%	11,1%	73	95	20
Q100	To what extent does Test Driven Development - meaning that first test are designed and after that the code is prepared - affects the delivery and quality of software solutions?	TP	33,3%	77,8%	11,1%	73	95	20
Q132	What factors affect team performance and system quality with regard to the number of teams working simultaneously on one application?	DP	33,3%	75,0%	0,0%	73	106	68
Q29	How can the performance of DevOps teams be benchmarked over departments and with regard to the Dreyfus model?	PROD	33,3%	66,7%	0,0%	73	128	68
Q43	How do naming standards affect the development of software?	CR	33,3%	66,7%	11,1%	73	128	20
Q97	To what extent does documentation during software maintenance affect delivery of software solutions?	TP	33,3%	66,7%	11,1%	73	128	20
Q152	What factors affect the way how DevOps teams perform, with regard to product ownership, and business responsibilities?	DP	31,6%	94,7%	0,0%	89	9	68
Q69	To what extent affect pair programming - a software development technique in which two programmers work together at one workstation and switch roles from code writer to reviewer frequently - the delivery of software solutions?	PROC	31,3%	75,0%	6,3%	90	106	55
Q74	To what extent affects an agile way of working the platform or tools used for delivery of software solutions?	PROC	31,3%	68,8%	12,5%	90	124	15
Q56	How much time is spent on risk management, security coding, and resilient architecture as part of software development?	SL	30,0%	90,0%	0,0%	92	30	68
Q6	Do unit tests save more time in debugging than they take to write, run, or keep updated?	BEST	30,0%	90,0%	0,0%	92	30	68
Q129	What factors affect running systems on many Linux platforms versus running systems on a centralized mainframe?	BEST	30,0%	90,0%	0,0%	92	30	68
Q55	How much time does it take for a beginning software engineer to start having a real contribution to a DevOps team, and how to optimize that?	PROD	30,0%	90,0%	0,0%	92	30	68

	Question	Category	Percentages Essential	Percentages Worthwhile+	Percentages Unwise	Rank Essential	Rank Worthwhile+	Rank Unwise
Q103	What aspects affects the performance of DevOps teams and the quality of software with regard to an agile way of working and working in DevOps teams?	PROD	30,0%	90,0%	0,0%	92	30	68
Q20	How can effort needed - in terms of time, effort, and cost - to perform a software project reliable be estimated?	DP	30,0%	80,0%	10,0%	92	84	32
Q112	What factors affect different test frameworks with regard to effort spend and result of the test performed?	BEST	30,0%	60,0%	0,0%	92	144	68
Q148	What factors affect the use of Docker - combined with Docker management solution, such as kubernetes - and a delivery pipeline, such as the one supplied by open-stack with regard to team performance?	BEST	28,6%	85,7%	7,1%	99	64	49
Q144	What factors affect the performance of DevOps teams and the quality of software code with regard to growing software systems organically versus start building a software system from scratch?	EQ	28,6%	85,7%	0,0%	99	64	68
Q150	What factors affect the use of PowerShell versus Ansible for deployment of systems?	BEST	27,8%	55,6%	0,0%	101	152	68
Q126	What factors affect reliability and security of software libraries?	DP	27,3%	90,9%	0,0%	102	26	68
Q17	How can data be cashed in order to prevent from retrieving data multiple times?	DP	27,3%	81,8%	9,1%	102	77	36
Q87	To what extent affects the reservation of time and effort dedicated for re-factoring purposes the performance of a DevOps team and the quality of software?	PROD	27,3%	81,8%	0,0%	102	77	68
Q84	To what extent affects screen size the performance and productivity of software engineers?	PROD	27,3%	81,8%	0,0%	102	77	68
Q26	How can software engineers know what other engineers are using a software component when adjusting one?	RSC	27,3%	72,7%	0,0%	102	115	68
Q83	To what extent affects redesign of code the quality of code and the performance of DevOps teams?	EQ	26,7%	93,3%	0,0%	107	19	68
Q136	What factors affect the current setup of networks and WIFI versus an open segment of the network where Internet and a limited number of ING network resources are available?	BEST	26,7%	73,3%	0,0%	107	114	68
Q31	How can the process - in terms of the state of individual requests - be monitored in the production environment?	SVC	26,3%	89,5%	0,0%	109	42	68
Q108	What factors affect an architectural design of an application with regard to availability of such a design, and quality of such a design?	DP	26,3%	84,2%	0,0%	109	70	68
Q16	How can code be made understandable without extensive documentation?	DP	26,3%	78,9%	5,3%	109	93	63
Q160	What is the status of quantum-computing and what effect will it have on software development?	BEST	26,3%	57,9%	10,5%	109	148	29
Q106	What factors affect a focus on feature development instead of migrating towards new technologies or architectures versus a focus on migration towards new technologies or architectures as a part of feature development?	BEST	25,0%	87,5%	0,0%	113	56	68
Q24	How can software development be simplified in order to make it accessible for more people?	PROC	25,0%	81,3%	12,5%	113	82	15
Q157	What impact does code quality have on the ability to monetize a software service?	DP	25,0%	80,0%	5,0%	113	84	67
Q13	How can an inventory of code to be re-compiled or re-linked be prepared when a configuration item is changed?	BEST	25,0%	75,0%	0,0%	113	106	68
Q54	How does the use of a shell on Unix - both in industry and within ING- influence performance of software developing teams?	BEST	25,0%	62,5%	6,3%	113	142	55
Q138	What factors affect the development- and operations (DevOps) tools used by system engineers versus front-end engineers?	PROC	25,0%	50,0%	0,0%	113	157	68
Q51	How does knowledge sharing of backlog components affect team performance?	TC	23,5%	70,6%	5,9%	119	121	59
Q171	Why is security by many developers seen as 'not sexy'?	BUG	23,1%	46,2%	7,7%	120	162	47

			Percentages			Rank		
	Question	Category	Essential	Worthwhile+	Unwise	Essential	Worthwhile+	Unwise
Q158	What is the data flow structure of the application like upstream and downstream information of the application?	DP	22,2%	88,9%	0,0%	121	44	68
Q40	How do brightness and contrast of monitors relate to human eyes and brain with regard to fatigue?	BEST	22,2%	88,9%	0,0%	121	44	68
Q99	To what extent does test automation affect delivery of software solutions?	TP	22,2%	88,9%	0,0%	121	44	68
Q78	To what extent affects commitment of an individual software engineer or a DevOps team to desired time-lines the performance of a DevOps team and the quality of software?	PROD	22,2%	77,8%	0,0%	121	95	68
Q102	What are good ways for software engineers keep up to date with relevant technological developments?	BEST	22,2%	66,7%	0,0%	121	128	68
Q27	How can software solutions in one common language be developed in a way that it is applicable to every person, regardless of ones interest in software development?	CR	22,2%	55,6%	33,3%	121	152	1
Q101	To what extent should pen-testing be done within a team itself, or by a specialized pen-testing team?	TP	22,2%	44,4%	0,0%	121	165	68
Q23	How can software complexity best be measured with regard to agility of the code base?	DP	21,1%	84,2%	5,3%	128	70	63
Q170	Why do many developers focus on the newest of the newest? Why don't they leave this to a small group in order to use time and effort more efficient?	DP	21,1%	47,4%	26,3%	128	161	3
Q35	How can we create an overview from all NPA's (Microsoft Network Policy and Access Services) with their authorizations on Windows server?	DP	21,1%	42,1%	10,5%	128	167	29
Q105	What delays are most common inside development projects and what are the most common reasons for these delays, and how such delays be avoided?	PROC	20,0%	93,3%	6,7%	131	19	51
Q124	What factors affect platform selection with regard to data being processed on it?	DP	20,0%	90,0%	0,0%	131	30	68
Q68	To what extent affect different branching mechanisms the performance of DevOps teams and the quality of code?	DP	20,0%	90,0%	0,0%	131	30	68
Q4	Do distributed version control systems offer any advantages over centralized version control systems?	BEST	20,0%	90,0%	0,0%	131	30	68
Q79	To what extent affects data science the quality of code or the performance of DevOps teams?	EQ	20,0%	86,7%	0,0%	131	59	68
Q130	What factors affect software engineer productivity with regard to being a polyglot versus becoming an expert in one language?	PROD	20,0%	70,0%	10,0%	131	122	32
Q62	Is a static typed language better than a dynamic typed language?	BEST	20,0%	60,0%	0,0%	131	144	68
Q61	How well do time estimates approximate the actual time taken to complete a project?	DP	20,0%	55,0%	15,0%	131	154	12
Q66	Since there is a lot of variation of methods used within ING, what factors affect software delivery with regard to the different software development methods that are used in practice?	PROC	18,8%	75,0%	6,3%	139	106	55
Q21	How can PL1 software code be converted to Cobol code, while maintaining readability of the code in order to simplify an application environment?	BEST	18,2%	36,4%	18,2%	140	169	8
Q5	Do distributed version control systems offer any advantages over centralized version control systems?	PROD	18,2%	18,2%	9,1%	140	171	36
Q25	How can software development process simulations help to examine the impact of changes such as new policies and changes in the way of working?	PROC	17,6%	82,4%	0,0%	142	75	68
Q149	What factors affect the use of machine learning in software development over a period of ten years?	DP	16,7%	66,7%	16,7%	143	128	10
Q46	How do static code analysis tools such as Fortify and Sonar influence the quality of software engineering products?	BEST	14,3%	85,7%	0,0%	144	64	68
Q155	What factors affect using frameworks such as RIAF and BakerCatlogue as feature-rich monoliths versus smaller programs with long-time support (LTS) versions?	DP	14,3%	71,4%	0,0%	144	119	68

	Question	Category	Percentages Essential	Percentages Worthwhile+	Percentages Unwise	Rank Essential	Rank Worthwhile+	Rank Unwise
Q128	What factors affect running individual software programs with regard to a dedicated versus a shared environment?	EQ	14,3%	71,4%	7,1%	144	119	49
Q93	To what extent do data scientists affect the delivery of software solutions?	PROC	14,3%	57,1%	0,0%	144	150	68
Q156	What factors affect working in squads versus working in traditional project teams?	TC	14,3%	42,9%	14,3%	144	166	13
Q9	Does a trade-off between existing knowledge and usefulness of emerging technology affect the choice of a programming language for an upcoming project?	PROC	13,3%	66,7%	6,7%	149	128	51
Q44	How do risk management efforts lead to less security related incidents and higher availability of systems?	BUG	12,5%	75,0%	0,0%	150	106	68
Q3	Debugging old code often is complex; what factors affect the quality of legacy systems with regard to debugging complexity, retention of developers, requirements, and documentation?	PROC	12,5%	68,8%	6,3%	150	124	55
Q77	To what extent affects changing of requirements during development the delivery of software solutions?	PROC	12,5%	68,8%	18,8%	150	124	7
Q42	How do dependencies in Java code affect satisfaction of software engineers?	DP	12,5%	62,5%	0,0%	150	142	68
Q91	To what extent affects the use of a code generator the quality of software?	EQ	12,5%	56,3%	0,0%	150	151	68
Q117	What factors affect installing new infra-services - such as a server - with regard to configuration effort needed, lead setup time, and cost?	PROC	11,1%	88,9%	0,0%	155	44	68
Q81	To what extent affects licensing of tools the performance and productivity of DevOps teams?	DP	10,5%	63,2%	0,0%	156	141	68
Q86	To what extent affects the choice of a data modelling approach affects performance and productivity of a DevOps team?	DP	10,5%	57,9%	5,3%	156	148	63
Q57	How much time should be spent on average on sharing knowledge within a software development team?	BEST	10,0%	80,0%	10,0%	158	84	32
Q167	When does it make sense to reinvent the wheel versus use an existing library?	BEST	10,0%	60,0%	0,0%	158	144	68
Q60	How viable is it to use a formal method such as model checking next to testing?	BEST	10,0%	60,0%	0,0%	158	144	68
Q15	How can be described in simple, daily terms how software products run?	DP	10,0%	50,0%	10,0%	158	157	32
Q151	What factors affect the use of technical writers with regard to organization of writings and documentation in order to increase trace-ability and readability?	DP	9,1%	63,6%	9,1%	162	139	36
Q110	What factors affect data analytics with regard to the use of external sources - such as market research reports and follow market trends - and let individual teams handle their local evolution?	PROC	9,1%	63,6%	0,0%	162	139	68
Q38	How can we measure the time to market of software solutions delivered within a department at ING in order to benchmark the performance of that department against others.	DP	9,1%	54,5%	18,2%	162	155	8
Q39	How can Windows-server images be created in order to facilitate testing within a continuous delivery pipeline?	DP	9,1%	45,5%	27,3%	162	163	2
Q65	Is there a correlation between certification and developer effectiveness in the field?	PROD	9,1%	45,5%	9,1%	162	163	36
Q89	To what extent affects the time spent - in terms of full-time versus part-time - of a Scrum master the delivery of software solutions?	PROC	6,7%	20,0%	13,3%	167	170	14
Q28	How can the cost of data be identified, in order to sign a price tag to data?	DP	5,6%	50,0%	16,7%	168	157	10
Q169	Why are there so many open source projects for the same function?	BEST	5,6%	38,9%	11,1%	168	168	20
Q49	How does experience of software developers and ops-engineers influence performance and quality of software products?	BEST	0,0%	66,7%	0,0%	170	128	68
Q72	To what extent affects a clear separation between tests - such as unit test, integration test, end-to-end test - the delivery of software solutions?	TP	0,0%	60,0%	10,0%	170	112	18

Table 11: Statistically significant rating differences by demographics.

	Question	Category	Response	Discipline Dev	Test	PM
Q2	Are there practices of good software teams from the perspective of releasing software solutions into production?	PROC	Essential	**66.7%**	5.6 %	11.1%
Q110	What factors affect data analytics with regard to the use of external sources - such as market research reports and follow market trends - and let individual teams handle their local evolution?	PROC	Essential	**66.7%**	5.6 %	11.1%
Q89	To what extent affects the time spent - in terms of full-time versus part-time - of a Scrum master the delivery of software solutions?	PROC	Essential	**66.7%**	5.6 %	11.1%
Q21	How can PL1 software code be converted to Cobol code, while maintaining readability of the code in order to simplify an application environment?	BEST	Essential	**66.7%**	4.8 %	0.0%
Q88	*To what extent affects the test ability of software code the quality of code?*	EQ	Essential	**68.4%**	0.0 %	0.0%
Q95	To what extent do dependencies on other teams affect team performance?	TC	Essential	**68.4%**	0.0 %	0.0%
Q162	What methods are most effective in preventing security related vulnerabilities or bugs from being introduced in software code?	BUG	Essential	**68.4%**	0.0 %	0.0%
Q28	How can the cost of data be identified, in order to sign a price tag to data?	DP	Essential	**72.7%**	0.0 %	0.0%
Q97	To what extent does documentation during software maintenance affect delivery of software solutions?	TP	Essential	**50.0%**	0.0 %	0.0%
Q46	How do static code analysis tools such as Fortify and Sonar influence the quality of software engineering products?	BEST	Essential	**36.6%**	0.0 %	27.3%

The demographic with the highest rating is highlighted in **bold**. Questions that are also in Table 5 are shown in *italics*.

	Question	Category	Response	Management Role Manager	Individual	Architect
Q2	Are there practices of good software teams from the perspective of releasing software solutions into production?	PROC	Essential	41.4%	**44.8** %	6.9%
Q153	What factors affect trunk-based development - a source-control branching model, where developers collaborate on code in a single branch - with regard to quality of software code?	BEST	Essential	22.6%	**54.8** %	9.7%
Q97	To what extent does documentation during software maintenance affect delivery of software solutions?	TP	Essential	10.0%	**60.0** %	20.0%
Q46	How do static code analysis tools such as Fortify and Sonar influence the quality of software engineering products?	BEST	Essential	**69.2%**	15.4 %	0.0%

The demographic with the highest rating is highlighted in **bold**. Questions that are also in Table 5 are shown in *italics*. The role "Manager" includes the responses for "Manager" and "Lead".