Maintaining Behaviour Driven Development Specifications: Challenges and Opportunities

Leonard Peter Binamungu, Suzanne M. Embury, and Nikolaos Konstantinou
School of Computer Science, University of Manchester, Manchester, UK
{leonardpeter.binamungu,suzanne.m.embury,nikolaos.konstantinou}@manchester.ac.uk

***Abstract*—In Behaviour-Driven Development (BDD) the behaviour of a software system is specified as a set of example interactions with the system using a "Given-When-Then" structure. These examples are expressed in high level domain-specific terms, and are executable. They thus act both as a specification of requirements and as tests that can verify whether the current system implementation provides the desired behaviour or not. This approach has many advantages but also presents some problems. When the number of examples grows, BDD specifications can become costly to maintain and extend. Some teams find that parts of the system are effectively frozen due to the challenges of finding and modifying the examples associated with them. We surveyed 75 BDD practitioners from 26 countries to understand the extent of BDD use, its benefits and challenges, and specifically the challenges of maintaining BDD specifications in practice. We found that BDD is in active use amongst respondents, and that the use of domain specific terms, improving communication among stakeholders, the executable nature of BDD specifications, and facilitating comprehension of code intentions are the main benefits of BDD. The results also showed that BDD specifications suffer the same maintenance challenges found in automated test suites more generally. We map the survey results to the literature, and propose 10 research opportunities in this area.**

***Index Terms*—behaviour-driven development, test suite maintenance, test suite evolution**

I. Introduction

In Behaviour-Driven Development (BDD), the behaviour of the required software is given as a collection of example interactions with the system, expressed using natural language sentences organised around a "Given-When-Then" structure [1], [2], [3]. This gives a specification that is expressed in non-technical, domain-specific terms, that should be readable and comprehensible by end-users. Importantly, the specification is also executable, thanks to "glue code" that links the natural language sentences to the code that is being built. Thus, the set of examples acts both as a high-level specification of the requirements for the software and as a suite of acceptance tests that can verify whether the current implementation meets the specification or not.

Like many other agile practices, the adoption and continued use of BDD is affected by organizational, people, process and technical factors discussed by Senapathi and Srinivasan [4], Vijayasarathy and Turk [5], Senapathi and Drury-Grogan [6], and Abdalhamid and Mishra [7]. As the technique enters its second decade of use, a considerable body of experience has been built up by practitioners, and lessons have been learnt about both the strengths and the challenges involved in its practical application. Anecdotal evidence from the software engineers we have worked with suggest that the maintenance challenges, in particular, can be severe, and are leading some teams to drop the technique and to return to technology-facing automated testing to play the role of their BDD specifications. However, to the best of our knowledge, no empirical studies have been undertaken by the academic community to capture these lessons and to understand how research might be able to address some of the problems encountered by users of large BDD specifications over the long term.

To make a start in filling this gap, we surveyed 75 BDD practitioners from 26 countries across the world on their experiences of using BDD. We collected both quantitative and qualitative data that gave us answers to the following research questions (RQs):

- **RQ1:** Is BDD in a considerable active use in industry at present?
- **RQ2:** What are the perceived benefits and challenges involved in using BDD?
- **RQ3:** To what extent are maintenance challenges prominent amongst the issues raised by users (and former users) of BDD, and what form do they take?

In addition to these more general questions, we wanted to test our hypothesis that duplication in BDD specifications is hard to detect and a cause of many of the maintenance challenges we have heard reported anecdotally by software teams. We therefore added a fourth research question to guide the design of the survey:

- **RQ4:** To what extent is the discovery and management of duplicates in BDD specifications seen as an unsolved problem by practitioners, and what techniques are being employed to deal with it?

From the survey results, we found that BDD is in active use in industry. Some respondent organizations use it on all projects, while the majority of respondent organizations use it on only some of the projects. Also, while a few previous practitioners are not currently using it due to various challenges, some of which are maintenance related, the majority of respondents among currently active and non-active practitioners plan to use BDD in the future as either a key tool on all projects or as an optional tool on some projects. Respondents gave the main benefits of BDD as the use of domain specific terms, improving communication among stakeholders, the executable nature of BDD specifications and facilitating comprehension

978-1-5386-4969-5/18/$31.00

of code intention. While respondents cited changing the way teams approach software development as the main downside of BDD, a significant number of responses indicated that BDD specifications suffer from the same maintenance challenges found in automated test suites.

Despite the reported maintenance challenges, we are aware of only the work of Suan [8] in this area. We conclude that there is a scarcity of research to inform the development of better tools to support the maintenance and evolution of BDD specifications, and we propose 10 open research opportunities in this area.

The paper is structured as follows: section II presents the current state-of-the-art in general adoption, and use of agile methods and BDD in particular; section III details the survey design and gives the details of participants; section IV presents the results of the study and the open research opportunities, while section V discusses the significance of the results. Section VI concludes.

II. Related Work

Senpathi *et al.* used the Agile Usage Model to identify the factors that affect the effective use of agile methods after they have been adopted by organizations, and the impact of the adoption [4]. In other work, Vijaysarathy *et al.* investigated reasons why people and organizations adopt and use agile practices, and the benefits and challenges that face development teams at early stages of adoption of particular agile practices [5]. Various summaries of the organizational, people, process and technical factors that affect the adoption of agile methods can be found in the literature [6], [7], [9], [10]. However, to the best of our knowledge, no study has attempted to examine these factors in the context of BDD.

The characteristics of BDD, the associated tools, and the strengths and limitations of the existing tool support have been summarized by Solis and Wang [11] and by Okolnychyi and Fögen [12]. Mishra discusses the general pros and cons of BDD [3] and Rahman and Gao highlight some of the maintenance challenges that teams might face when adapting BDD tests to changing environments [13]. The feasibility of BDD in practice and the ease with which people can learn and understand Gherkin, the most commonly used BDD language [12], was investigated by Rai [14]. Suan investigated techniques for detecting duplicates in BDD specifications through text similarity matching [8]. We found no published study that has attempted to establish the maintenance challenges involved in using BDD on projects in the long term.

III. Research Method

This section presents the survey design, the details of the participants, and the data analysis approach.

A. Survey Design

To attract more responses and yet ensure that aspects of interest are well captured, we designed a short web survey with 18 questions. Of the 18 questions, 6 were single choice, 7 were multiple choice, and 5 required open-ended free-form text responses. All single and multiple choice questions had an "other(s)" option to allow respondents to report other important information that might not have been well covered in the choices we gave. More specifically, 7 questions were on the the extent of BDD use; 5 questions sought to establish the sizes of typical BDD specifications in industry, the challenges presented by the presence of duplication in BDD specifications, and how industry teams detect and manage duplication in BDD specifications; 1 question sought to reveal any other noteworthy issues about BDD from practitioners; and 5 questions were on the demographics of the respondents. For brevity, we refer readers to section IV where the specific survey questions and the choice answers are discussed.

The survey was reviewed by a senior academic from our school who has experience in doing survey research. It was designed and deployed using SelectSurvey.NET[1], an instance of which is available for use on our university servers. Our respondents took 5-10 minutes to complete the survey, and we received responses over the period of more than 2 and a half months from July 2017.

B. Respondents

Developers of BDD projects and other members of industry agile teams who had ever used BDD, or were using BDD at the time of our survey, were our targets. The survey was distributed through a convenience sample of online agile discussion groups and personal emails. Though it reduces generalizability of findings, convenience sampling is appropriate when random sampling is practically impossible [15], [16]. The survey was accessed and completed through a web link. We also encouraged respondents to pass the survey on to others, and so some of our respondents might have been recruited through snowballing. A similar method of recruiting survey respondents was used by Kochhar *et al.* [17], and Witschey *et al.* [16].

We began by posting the survey on on-line communities where BDD topics are discussed. The following Google groups were targeted: Behaviour Driven Development Google group, Cucumber Google group, and BDD Security Google group. After learning about the survey through one of these groups, the Editor in Chief of BDD Addict[2], a monthly newsletter about BDD, included the survey in the July 2017 issue, in order to reach a wider BDD community. The survey was also shared with the following twitter communities: Agile Alliance, Agile Connection, Agile For All, Scrum Alliance, Master Scrum, RSpec—BDD for Ruby, and the twitter accounts for two of the authors.

We further identified email addresses of contributors to BDD projects on GitHub.com, and sent them personalized requests to complete our survey. Relevant projects were identified using the keywords "Behaviour Driven Development", "Behavior Driven Development", and "BDD"; we extracted the public email addresses of contributors from the resulting projects, up to and including the 5th page (10 results per page). (We selected

[1] http://selectsurvey.net/

[2] https://www.specsolutions.eu/news/bddaddict/

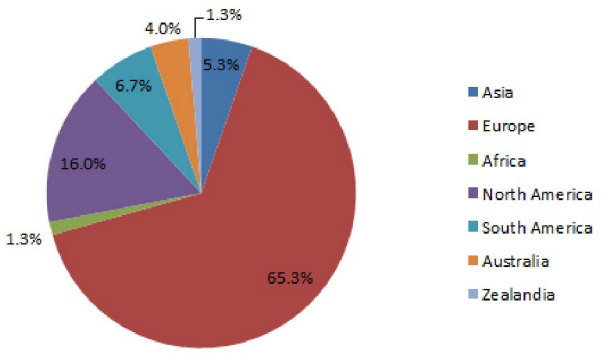

Fig. 1. Distribution of respondents by continent

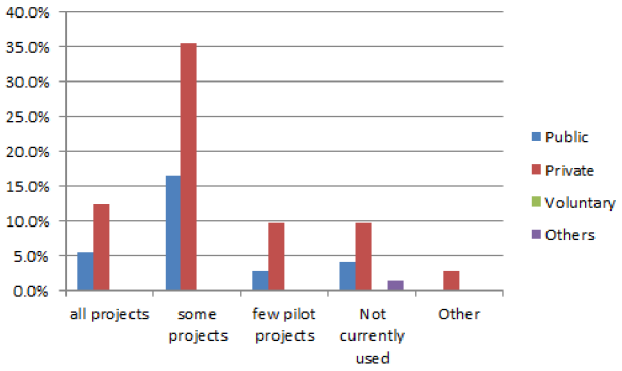

Fig. 2. Extent of BDD use by type of organization

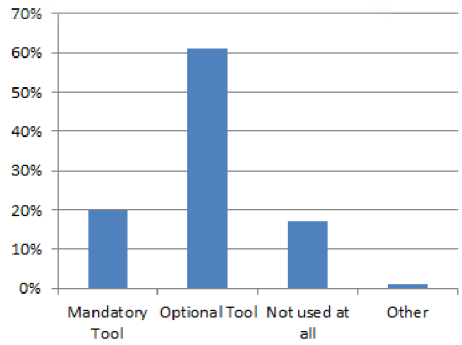

Fig. 3. Extent to which use of BDD is mandated by organisations

this limit after manual examination of the usefulness of email addresses on later pages for a sample of projects). We also searched for projects with keywords based on the names of the tools mentioned in the survey: namely, *Cucumber*, *FitNesse*, *JBehave*, *Concordion*, *Spock*, *easyb*, and *Specflow*. In total, 716 email addresses were identified and contacted about the survey.

Of the 566 people who viewed the survey, 82 began to complete it out of whom 75 submitted usable responses to the main questions. 11 out of the 13 (84.6%) main questions (questions not focusing on respondents' demographics) were completed by all the 75 respondents. We used *IP Address Geographical Location Finder*[3] to identify the geographical locations of respondents, and Fig. 1 shows the distribution of respondents by continent.

The organizations of respondents were distributed as follows: 35% public, 63% private, 1% sole trader, and 1% did not say. 44 participants gave the role they held in their organization, and most were in senior positions. However, the remaining 31 preferred not to state their roles, probably because we explicitly stated that identifying information was optional. Table I shows the distribution of job roles for the respondents. Additionally, though not everyone stated their organization, we noted that some respondents who specified their organizations came from large, well known multinational organizations.

TABLE I
DISTRIBUTION OF JOB ROLES OF SURVEY RESPONDENTS

Role	No.	%
Software Engineers/Architects	20	26.7
Quality Assurance Engineers/Business Analysts	4	5.3
Team Lead/DevOps Tech Lead	10	13.3
Consultant	3	4.0
Chief Executive Officer (CEO)	2	2.7
Chief Technology Officer (CTO)	4	5.3
Researcher	1	1.3
Did not say	31	41.3
Total	**75**	**100.0**

[3]http://www.ipfingerprints.com/geolocation.php

C. Data Analysis

We received a total of 82 responses. We removed 7 responses in which respondents had completed only demographics data, leaving 75 valid responses. To answer the research questions introduced in section I, we plotted the charts to represent the responses, and summarized the respondents' comments.

IV. RESULTS

This section presents the extent of BDD use amongst the BDD centric communities we surveyed, and the challenges they face.

A. Extent of Active Use of BDD

1) Extent of BDD Use in Various Types of Organizations: From the survey, we found that BDD is used more in private organizations than in other types of organization (see Fig. 2). Fig. 3 summarises responses as to whether BDD is a mandatory or optional tool for organizations.

2) Tools Used by Industry Teams to Support BDD and ATDD: Respondents use the tools in Fig. 4. Strictly speaking, some of these are more properly termed *Acceptance Test Driven Development (ATDD)* tools.

3) Plans to Use BDD in the Future: Almost half of the respondents said that their organizations will use BDD as an optional tool on some projects in the future, while more than a quarter of the respondents said that it will be used as a key tool on all projects. Fig. 5 summarizes the responses on planned future BDD use.

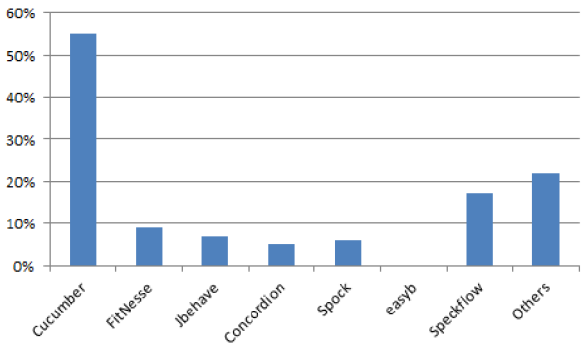

Fig. 4. BDD Tools used by respondents

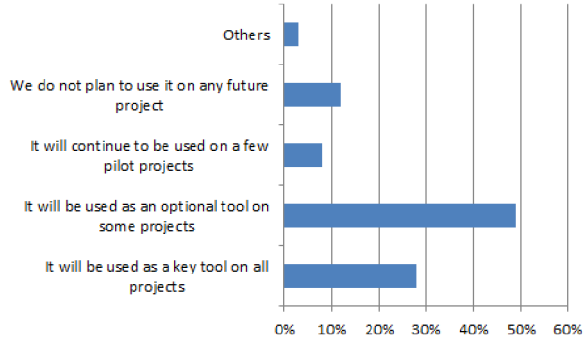

Fig. 5. Plans by organizations to use BDD in the future

B. Perceived Benefits and Challenges

1) Perceived Importance and Benefits of BDD: Fig. 6 presents the perceived importance of BDD use by the respondents. The views given under "other" were:

- *"Personally I find it very important, my clients though have different opinions. Usually it requires a certain collaboration within the organization which is hard to establish. It is not the tool that is hard to use, but more the people to get into this work flow."*
- *"BDD enables teams to write standard tests that are more expressive."*

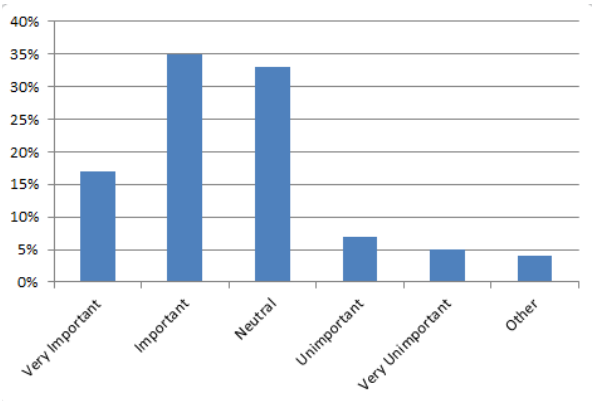

Fig. 6. Perceived importance of BDD use

Respondents' opinions on the benefits of BDD are presented in Table II. As the results show, respondents value the communication aspects of BDD, but also the benefits to developers in gaining early warning signals of problems.

TABLE II
BENEFITS AND CHALLENGES OF USING BDD

Benefits of BDD	**Rate (%)**
Software specifications are expressed in domain-specific terms, and thus can be easily understood by end users	67
Improves communication between various project stakeholders	61
Specifications can be executed to confirm correctness or reveal problematic software behaviour(s)	52
Code intention can be easily understood by maintenance developers	50
Attention is paid to validation and proper handling of data	24
Could produce better APIs since it emphasizes writing testable code	28
Other	7
Challenges of BDD	
Its use changes the team's traditional approach to software development, and that can be challenging	51
Its benefits are hard to quantify	35
It involves a steep learning curve	28
Other	21
It can lower team productivity	20

Under "other", respondents listed the following additional benefits:

- BDD offers an improved way of documenting the software and the associated code:
  - *"Living documentation that evolves with the system over time"*
  - *"Documentation is a working code"*
- Simplifies and enriches software testing activities:
  - *"Helps QA team to write tests without code implementation details"*
  - *"Make possible fullstack tests, differently from unit tests."*
  - *"Reusable, finite set of steps used by test developers"*
- Improves software designs by facilitating domain knowledge capture:
  - *"primarily a design tool → it enables us to gain clarity about the domains at hand, especially at the seams"*

2) Challenges Faced by BDD Practitioners: We wish to state at the outset that: because of the strong emphasis on collaboration among all project stakeholders, there is a thin line between organizational, people, process and technical challenges in the BDD workflow. However, in the present paper, the separation should be self-evident, should it be important.

The challenges faced by BDD practitioners, according to the respondents, are given in Table II. Respondents thought that the most challenging part of BDD is that it changes the usual approach to team software development. Under "other", the following challenges were mentioned:

- The emphasis on collaboration, an inherent part of a correct BDD workflow, can be difficult and even ignored, leading to later problems:
  - *"Needing to involve Business and final users"*
  - *"It's a simple concept but can be hard to get right. Many people make the assumption it's about test automation and try to use like a scripting tool and the project ends in failure"*
  - *"it does not succeed at being legible to colleagues outside of software engineering departments."*
  - *"Make other non-developers read tests. So far I have used BDD for couple of years and even though idea behind it [is] good, people who are not involved in testing are also not interested in test cases no matter how easy-to-read they are."*
- Lack of BDD coaching and improper application of the BDD workflow:
  - *"As with other kinds of testing, the best way to learn is from somebody who has experience. Thus just by downloading a framework, reading a bit and trying, one can produce tests which value is disputable."*
  - *"Danger of confusing the mechanics (automation, written specifications) with the intention (knowledge sharing, structured conversations, discovery of edge cases), focusing too much on the former."*
  - *"Once the Gherkin syntax is well known, stakeholders tend to skip ahead, reducing the benefits of the specification workshop."*
  - *"Its hard to find someone who really understand what should be tested by BDD therefore a bunch of developers have negative experience about it. Probably there is no a comprehensive material on the internet that can explain every aspect of BDD."*
  - *"requires design skills often absent or not valued"*
  - *"Main issue when applying BDD is to find time to do the Three Amigos workshop, it is not a tool issue but more a people one."*
- Ensuring that BDD specifications are easy to comprehend, execute and maintain:
  - *"All the usual challenges in getting automated testing running and maintained"*.
  - *"...Textual specs are too expensive to maintain long-term"*
  - *"BDD add unnecessary layer of maintaining specification and make them still readable with clean code."*
  - *"BDD is often associated with slow suites. The difficulty of managing duplication is proportional to that slowness. Therefore, as BDD scales, in my opinion it is crucial to find ways to run slow scenarios fast, either by reducing their scope, or by running them against multiple configurations of the system covered by the scenarios."*
  - *"...the complexity of the test software needed to support BDD is often as high as the software under test..."*

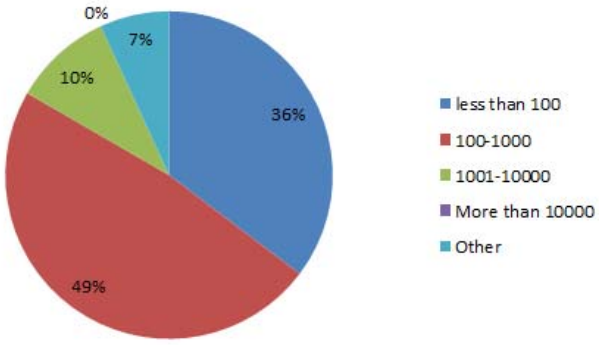

Fig. 7. Number of scenarios in industry BDD projects

  - *"Some developers don't like the duplication that can be created with having BDD separate to unit tests. BDD can also get out of hand and become far too technical and indecipherable by users"*
  - *"...tests were very brittle and manual QA types had limited ability to investigate."*

C. Challenges of Maintaining BDD Specifications

1) Size of BDD Suites: To provide context for the maintenance challenges reported, we asked for information about the typical sizes of the BDD suites used and managed by the respondents. Clearly, the maintenance challenges reported are likely to be of less significance if typical suites contain numbers of scenarios (i.e., examples) that can be managed by hand. Fig. 7 shows the typical size of the BDD suites the respondents work with. It can be noted that, while the majority are relatively small, a significant minority are considerably large to make manual individual inspection of all scenarios a costly task.

2) Maintenance Challenges: As it can be noted from the responses describing the general challenges of BDD, respondents admitted that BDD suffers from the same kinds of maintenance challenge associated with any current form of automated testing. Specifically, the maintenance challenges as presented earlier from the survey can be summarised as:

- Specifications can be hard to comprehend.
- It can be hard to locate sources of faults, especially in large BDD suites.
- It can be difficult to change specifications for the purpose or fault correction, accommodating new requirements, or adapting them to new environment.
- Making slow suites run faster.
- The need to maintain BDD tests in addition to unit tests.
- Coping up with the possible complexity of BDD tools.
- Duplication detection in BDD specifications.
- Duplication management in BDD specifications.

D. Duplication in BDD Suites

We now present the maintenance challenges presented by duplication in BDD suites, the extent at which duplication is present in industry projects, and the current state of practice in the detection and management of duplication in BDD specifications, as reported by the survey respondents.

1) Problems of Duplication: 61% of the respondents held the view that the presence of duplication in BDD specifications can cause them to become difficult to extend and change (leading potentially to frozen functionality). As well, while nearly half of the respondents (49%) said that the presence of duplication in BDD specifications can cause execution of BDD suites to take longer to complete than necessary, 43% thought that duplication can make it difficult to comprehend specifications. Under "other" (7%), the following problems of duplication in BDD suites were reported:

- The process of duplication detection and management can change the desired software behaviour:
  - *"Over refactoring features and scenarios to avoid duplication causes the requirements and their understanding to change from what the Product Owner wants."*
- Difficulty in comprehension and execution of specifications:
  - *"Contradicting specifications, if the duplication is not a result of the same team/individual working on it."*
  - *"Duplication in specs is usually a sign of incompletely or badly 'factored' behaviours, which can lead to overly complicated specs and difficult to set up system state."*
- Necessitates changes in several places in the suite during maintenance and evolution:
  - *"Changes required to be done in more than one place. I miss some 'include' keyword."*
- It is hard to use existing duplicate detection and management tools to detect and manage duplicates in specifications expressed in a natural language:
  - *"if the statements are in English prose basic refactoring tools / copy paste detection / renaming are difficult to catch and maintain."*
- BDD tests are end-to-end tests that are usually strongly connected to their unit tests, and that makes the process of detecting duplicate BDD tests difficult:
  - *"It's hard to detect duplication between BDD specs and unit-tests."*
- Difficulty in modelling how the scenarios are executed, and the scenarios can be very slow and brittle:
  - *"...criteria can hold at one level and cascade down - difficult to model *how* the scenarios are executed can be very slow and brittle (e.g. web tests) - hexagonal architecture please"*

2) Presence of duplication: Fig. 8 summarizes responses on the extent of the presence of duplication in the BDD specifications.

3) Detection and Management of Duplication in BDD Specifications: We now present the current state of practice in detecting and managing duplication in BDD specifications.

Fig. 9 relates the extent of duplication, suite size, and method of duplication detection reported by the survey. The pie chart in the same figure summarizes the different methods that are used to detect and manage duplication.

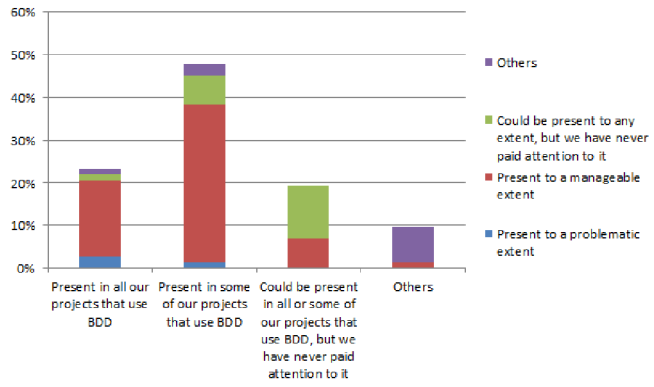

Fig. 8. Presence of duplication in the BDD specifications of industry teams

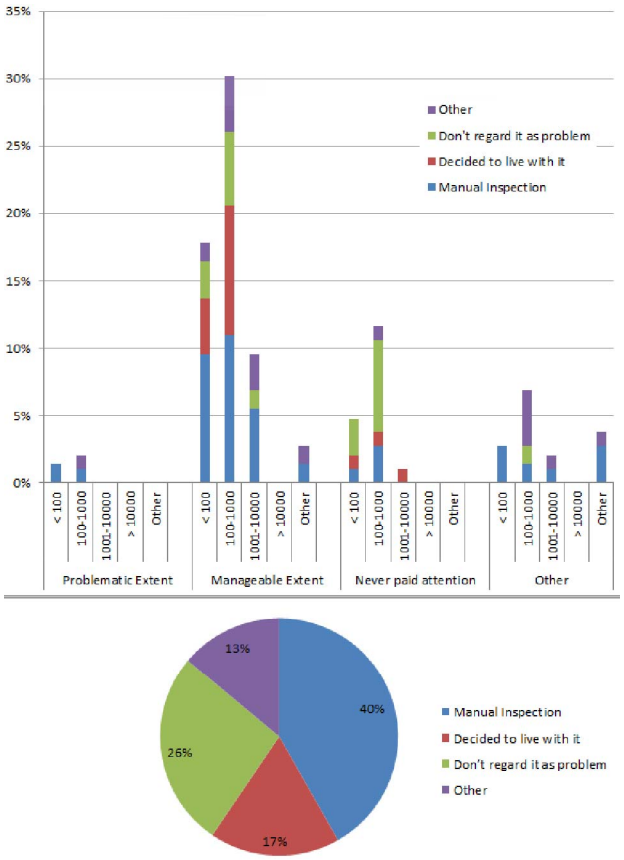

Fig. 9. Extent of duplication, size of BDD specs, and duplication detection method

Some respondents had the following additional thoughts on how they approach duplication detection and management:

- *"We are looking at ways to automate at least part of the process of finding duplicates"*
- *"Treat the test code much like the production code. Refactor frequently to control duplication and make test intentions clear"*
- *"Pay attention to SRP during or after collaborative specification."*

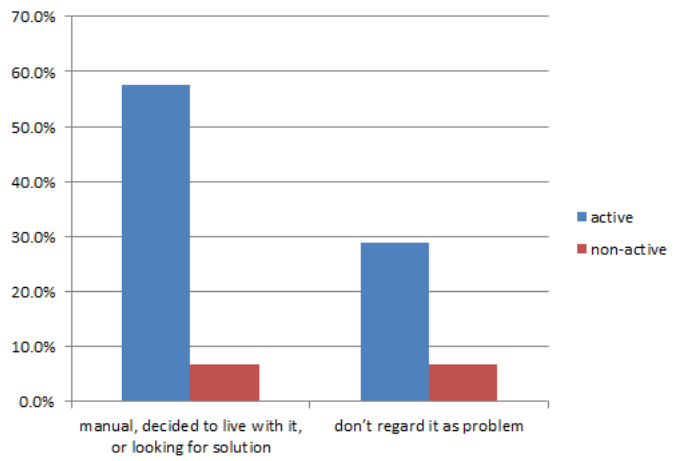

Fig. 10. Duplication detection methods among active and non-active BDD practitioners

- *"We organise the specifications specifically to prevent this. It would be one of the worst things to happen."*
- *"Using jbehave with 'givenScenario', we are able to reduce duplication by reusing steps."*

Fig. 10 shows the distribution of duplication detection methods among active and non-active BDD practitioners. An active practitioner in this regard is the one who uses BDD in either all projects, or some projects, or a few pilot projects. Almost 60% of the respondents were active BDD practitioners who either: perform manual inspection to detect duplication in their BDD specifications and thereafter decide on how to manage it, or have decided to live with it, given the complexity of the detection and management process, or are currently looking for an automated solution to detect and manage it.

E. Opportunities

Table III presents the research opportunities we derived from the challenges reported in the survey results, and maps the respective opportunities to the existing literature.

V. Discussion and Threats to Validity

We now discuss the significance of the results, providing answers to the research questions mentioned in section I. We also present the threats to validity of our results, and discuss the mitigation strategies.

RQ1: On whether BDD is in active use: To explain the activeness of BDD use in industry, we use the theory of *vertical* (or explicit) and *horizontal* (or implicit) use by Iivari *et al.* [49], [50]. Vertical use expresses the degree of rigour with which a particular method is followed, eg., strict adherence to the method's documentation or partial adherence. Horizontal use, on the other hand, refers to the use of a method across multiple teams and projects in an organization after initial adoption, learning, and internalization. With respect to *horizontal use*, for a range of organization types, we pay attention to: whether it is used on all projects, some projects, a few pilot projects, or not used at all; whether it is used as a mandatory or optional tool; and plans by organizations to use it in the future. As well, we use *vertical use* to discuss issues reported by practitioners that are related to conformity or non-conformity with the BDD workflow.

We note from the survey results that BDD is in active use in the industry. We learn from Fig. 2 and Fig. 3 that there is a substantial level of *horizontal use*, with some organizations using it on all projects, while others use it on some projects. Additionally, while there are organizations (20% of respondents) that have made BDD a mandatory tool, a significant proportion (61% of respondents) use it as an optional tool. This is to be expected as most organisations would use different software development techniques, for various reasons, including the dictates of a particular project. We can also expect that some organizations might use selected agile techniques, but not be committed users of every agile practice. We note, however, that there are *vertical use* concerns whereby some practitioners do not observe BDD best practice, notably by avoiding or downplaying the collaboration aspects, resulting in future costs. That said, we argue that the observed extent of use, and the plans to continue using BDD (Fig. 5) are sufficient to attract the attention of the research community in uncovering better ways to support BDD practitioners.

RQ2: Perceived benefits and challenges of using BDD: We use the following factors from the Agile Usage Model (AUM) [51], [6], [4] to explain the perceived importance, overall benefits and challenges of BDD. In the AUM, the following terms are used:

- **Relative advantage:** *"the degree to which the innovation is perceived to be better than its precursor"* ([4], p.2). This can be reflected in the ability of an agile method to offer benefits like improved productivity and quality, reduced cost and time, producing maintainable and evolvable code, improved morale, collaboration and customer satisfaction [5].
- **Compatibility:** *"the degree to which agile practices are perceived as being consistent with the existing practices, values, and past experiences"* ([51], p.4).
- **Agile Mindset:** A mindset that perceives challenges as learning opportunities, building on optimism to grow over time, with effort in place [4].
- **Agile Coach:** An individual with both technical and domain knowledge who can point an agile team in right directions without imposing matters [4].

Generally, BDD has appreciable rating, with more that 50% (Fig. 6) of the respondents affirming its importance. The use of domain specific terms, improving communication among stakeholders, the executable nature of BDD specifications, and facilitating comprehension of code intention are revealed as the main benefits of BDD (Table II). These all can be linked to the *relative advantage* factor in the AUM that BDD has over its precursor agile practices like Test-Driven Development [52]. Actually, it was the TDD's limitations in enabling teams to focus on implementing correct software behaviours that led

TABLE III
CHALLENGES AND RESEARCH OPPORTUNITIES FOR MAINTENANCE AND EVOLUTION OF BDD SPECIFICATIONS

ID	Challenge	Opportunity	Link to Related Literature
O_1	Hard to comprehend BDD specifications	Investigate BDD test smells, technical debt, and the adoption of test suite comprehension techniques to BDD specifications	[18], [19], [20], [21], [22], [23], [24], [25], [26], [27]
O_2	Difficulty of locating faults in large BDD suites	Investigate test fault localization techniques in the context of BDD specifications	[28], [29], [30]
O_3	Hard to change BDD suites	Investigate automated test repair for BDD specifications	[31], [32], [33], [34], [35]
O_4	Slow BDD suites	Investigate test minimization, selection and prioritization in the context of BDD	[36], [37], [38], [39]
O_5	The need to maintain BDD tests in addition to unit tests	Investigate integrated functional and unit test maintenance for BDD tests	[40], [35]
O_6	Complexity of BDD tools	Investigate BDD tools selection guides and possible tool improvements	[41], [42]
O_7	Duplication Detection in BDD specifications	Investigate duplication detection in the context of BDD	[43], [44]
O_8	Duplication management in BDD specifications	Investigate duplication management in the context of BDD	[45]
O_9	Non-adherence to the BDD workflow	Investigate the incorporation of maintenance concerns at the core of BDD workflow and tools	[46], [24], [21]
O_{10}	Scarcity of coaching and material guidelines on BDD	Investigate the impact of coaching and guidelines on producing maintainable specifications	[47], [48], [4], [21]

to the birth of BDD [1]. However, a comparative study would shed light on whether that foreseen BDD's potential has become practically evident.

The downside of BDD agreed to by most respondents has to do with changing the way teams used to approach software development. This is in line with the *compatibility* factor in the AUM: new innovations are likely to face resistance by some adopters, especially when the adopters are slow at embracing changes. We posit that an *agile mindset* is important in addressing this challenge. Teams' willingness to learn and adopt new techniques is important in the adoption of BDD. Imprecise understanding of the BDD workflow, non-adherence to it, the scarcity of coaching services and material guidelines also hinder the adoption and continued use of BDD. An *agile coach* with good understanding of the BDD workflow would help teams to navigate these challenges.

RQ3: Extent and form of maintenance challenges in BDD specifications: We learned that BDD tests suffer from the same maintenance challenges that test suites in automated testing face. Refer to section IV-C or table III for an extended list. Specifically on duplication, practitioners admitted that the presence of duplication in BDD specification causes, among others, test execution, maintenance and evolution problems. To be of desirable quality, like the code under test, test suites must be maintainable [21], [26], [53], [19], [54]. As such, we argue that it is important to investigate test maintainability aspects such as ease of change, bug detectability (for the bugs in both test and production code) as well as test comprehensibility [26], [21], and others, in the context of BDD, to support the work of practitioners.

RQ4: Problem of duplication in BDD specifications, and its detection and management: Most respondents think that, though present, duplication in their BDD specifications remains a manageable problem. However, duplication is still among the maintenance challenges of concern to some BDD practitioners. In fact, in some instances, it has scared away some practitioners from using BDD: *"We decided to not use BDD any more because it was hard to maintain it. In the beginning we were checking for duplication, but at one point it has become very hard to manage them. Even though our tests were very much readable, our code underneath became less and less readable."*

Referring to Fig. 9, for the most part, duplication detection and management is done manually (40% of respondents). Nevertheless, there is a significant proportion (17%) of respondents who have given up the duplication detection and management process, because of its complexity. Combining these, more than a half (57%) of the respondents are concerned with duplication detection and management, except for those in the "other" category, who either explicitly expressed their need for an automated solution or mentioned their specific current manual approach to duplication detection and management. We thus specifically identified O_7 and O_8 in Table III as opportunities for the research community to investigate innovative ways to help those who either use the manual process, or have given up, or are likely to experience serious duplication concerns in the future.

Opportunities: Based on the identified challenges, we summarise, in Table III, the available research opportunities and link them to the relevant existing literature that covers similar problems in other areas, apart from BDD. Specifically, O_1 to O_8 are directly related to maintenance, while O_9 and O_{10} focus on improving the process that is likely to result into specifications with significant maintenance problems. Inter alia, we argue that a body of scientific evidence is required to inform the following questions:

1) How could the BDD workflow be enhanced to produce maintainable specifications? Specifically, it might be worthy investigating on whether there are specific bits of the BDD workflow that are prone to producing hard-to-maintain specifications, and how that could be redressed.
2) Better ways to adapt existing unit test maintenance techniques to the context of BDD tests. Or how could better techniques and tools specifically for the maintenance of BDD tests be developed?

3) Better ways to apply the existing regression test suite reduction, selection and prioritization techniques [36] to address problems of slow suites due to the presence of duplication, and other concerns, in BDD specifications.
4) Characterization of duplication in the context of BDD specifications, and development of appropriate duplication detection techniques and tools.
5) How could the existing techniques and tools for duplication detection and management [43], [45] be applied to the problem of duplication detection and management in BDD specifications?

The threats to the validity of our results are as follows:

- We mainly depended on practitioners with online presence, either through github.com or other online forums where BDD and other agile topics are discussed. Thus, we might have missed some in-house practitioners that are not easily reachable through any of those means. To mitigate the effects of this, we requested those who completed or saw the survey to refer it to others. Also, we sent survey completion requests to some practitioners who were known in person to the authors, and requested them to share the survey to others.
- Some institutional laws and regulations might have determined whether or not participants responded in full. To mitigate the effects of this, we kept any identifying information optional, and clearly stated this at the beginning of the survey, and in all survey completion invitations.
- Most of the respondents might have been using a particular BDD tool. To mitigate the effects of this, the survey included seven tools from which respondents could choose several tools, as well as an "other(s)" option. Also, we followed the objective criteria mentioned in section III-B to identify email addresses to which survey completion requests were sent. Additionally, we posted the survey in general BDD and agile forums, in anticipation that respondents from those forums might be using different tools.

VI. Conclusions

BDD is now used by many software teams to allow them to capture the requirements for software systems in a form that is both readable by their customers and detailed enough to allow the requirements to be executed to check whether the production code implements the requirements successfully or not. The resulting feature descriptions, as sets of concrete scenarios describing units of required behaviour, provides a form of *living documentation* for the system under construction (as compared to the passive documentation and models familiar from other approaches to requirements engineering [2]). Unfortunately, management of BDD specifications over the long term can be challenging, particularly when they grow beyond a handful of features and when multiple development team members are involved with writing and updating them over time. Redundancy can creep into the specification, leading to bloated BDD specifications that are more costly to maintain and use.

Using quantitative and qualitative data collected through the web survey, we have reported the activeness of BDD use in industry, its benefits, general and specific maintenance challenges, particularly duplication. By reviewing the literature related to the identified challenges, we propose 10 research opportunities to support the maintenance and evolution of BDD specifications.

Acknowledgements

We wish to sincerely thank all of our respondents in the BDD and agile communities around the world, and Caroline Jay of the School of Computer Science, University of Manchester, UK, for reviewing our survey.

References

[1] D. North, "Introducing BDD," Better Software Magazine, 2006.
[2] M. Wynne and A. Hellesoy, *The Cucumber Book*. Pragmatic Programmers, LLC, 2012.
[3] A. Mishra, "Introduction to behavior-driven development," in *iOS Code Testing*. Springer, 2017, pp. 317–327.
[4] M. Senapathi and A. Srinivasan, "An empirical investigation of the factors affecting agile usage," in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. ACM, 2014, p. 10.
[5] L. Vijayasarathy and D. Turk, "Agile software development: A survey of early adopters," *Journal of Information Technology Management*, vol. 19, no. 2, pp. 1–8, 2008.
[6] M. Senapathi and M. L. Drury-Grogan, "Refining a model for sustained usage of agile methodologies," *Journal of Systems and Software*, vol. 132, pp. 298–316, 2017.
[7] S. Abdalhamid and A. Mishra, "Factors in agile methods adoption," 2017.
[8] S. W. Suan, "An Automated Assistant for Reducing Duplication in Living Documentation," Masters Thesis, School of Computer Science, University of Manchester, Manchester, United Kingdom, 2015.
[9] T. Chow and D.-B. Cao, "A survey study of critical success factors in agile software projects," *Journal of systems and software*, vol. 81, no. 6, pp. 961–971, 2008.
[10] D. Janzen and H. Saiedian, "Test-driven development concepts, taxonomy, and future direction," *Computer*, vol. 38, no. 9, pp. 43–50, 2005.
[11] C. Solis and X. Wang, "A study of the characteristics of behaviour driven development," in *Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on*. IEEE, 2011, pp. 383–387.
[12] A. Okolnychyi and K. Fögen, "A study of tools for behavior-driven development," *Full-scale Software Engineering/Current Trends in Release Engineering*, p. 7, 2016.
[13] M. Rahman and J. Gao, "A reusable automated acceptance testing architecture for microservices in behavior-driven development," in *Service-Oriented System Engineering (SOSE), 2015 IEEE Symposium on*. IEEE, 2015, pp. 321–325.
[14] P. Rai, "Extending automated testing to high-level software requirements: A study on the feasibility of automated acceptance-testing," Sweeden, 2016.
[15] R. D. Fricker Jr, "Sampling Methods for Online Surveys," *The SAGE Handbook of Online Research Methods*, p. 162, 2016.
[16] J. Witschey, O. Zielinska, A. Welk, E. Murphy-Hill, C. Mayhorn, and T. Zimmermann, "Quantifying developers' adoption of security tools," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ACM, 2015, pp. 260–271.
[17] P. S. Kochhar, X. Xia, D. Lo, and S. Li, "Practitioners' expectations on automated fault localization," in *Proceedings of the 25th International Symposium on Software Testing and Analysis*. ACM, 2016, pp. 165–176.
[18] F. Palomba, D. Di Nucci, A. Panichella, R. Oliveto, and A. De Lucia, "On the diffusion of test smells in automatically generated test code: An empirical study," in *Proceedings of the 9th International Workshop on Search-Based Software Testing*. ACM, 2016, pp. 5–14.
[19] A. Van Deursen, L. Moonen, A. van den Bergh, and G. Kok, "Refactoring test code," in *Proceedings of the 2nd international conference on extreme programming and flexible processes in software engineering (XP2001)*, 2001, pp. 92–95.

[20] G. Meszaros, *xUnit test patterns: Refactoring test code*. Pearson Education, 2007.

[21] D. Bowes, T. Hall, J. Petrić, T. Shippey, and B. Turhan, "How good are my tests?" in *Proceedings of the 8th Workshop on Emerging Trends in Software Metrics*. IEEE Press, 2017, pp. 9–14.

[22] G. Samarthyam, M. Muralidharan, and R. K. Anna, "Understanding test debt," in *Trends in Software Testing*. Springer, 2017, pp. 1–17.

[23] S. Panichella, A. Panichella, M. Beller, A. Zaidman, and H. C. Gall, "The impact of test case summaries on bug fixing performance: An empirical investigation," in *Proceedings of the 38th International Conference on Software Engineering*. ACM, 2016, pp. 547–558.

[24] E. Daka, J. M. Rojas, and G. Fraser, "Generating unit tests with descriptive names or: Would you name your children thing1 and thing2?" in *Proceedings of the 26th International Symposium on Software Testing and Analysis*, 2017, pp. 57–67.

[25] M. S. Greiler, "Test suite comprehension for modular and dynamic systems," 2013.

[26] D. Gonzalez, J. Santos, A. Popovich, M. Mirakhorli, and M. Nagappan, "A large-scale study on the usage of testing patterns that address maintainability attributes: patterns for ease of modification, diagnoses, and comprehension," in *Proceedings of the 14th International Conference on Mining Software Repositories*. IEEE Press, 2017, pp. 391–401.

[27] M. Greiler, A. Zaidman, A. v. Deursen, and M.-A. Storey, "Strategies for avoiding text fixture smells during software evolution," in *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 2013, pp. 387–396.

[28] A. Vahabzadeh, A. M. Fard, and A. Mesbah, "An empirical study of bugs in test code," in *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*. IEEE, 2015, pp. 101–110.

[29] R. Ramler, M. Moser, and J. Pichler, "Automated static analysis of unit test code," in *Software Analysis, Evolution, and Reengineering (SANER), 2016 IEEE 23rd International Conference on*, vol. 2. IEEE, 2016, pp. 25–28.

[30] M. Waterloo, S. Person, and S. Elbaum, "Test analysis: Searching for faults in tests (n)," in *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*. IEEE, 2015, pp. 149–154.

[31] B. Daniel, T. Gvero, and D. Marinov, "On test repair using symbolic execution," in *Proceedings of the 19th international symposium on Software testing and analysis*. ACM, 2010, pp. 207–218.

[32] B. Daniel, V. Jagannath, D. Dig, and D. Marinov, "Reassert: Suggesting repairs for broken unit tests," in *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*. IEEE Computer Society, 2009, pp. 433–444.

[33] S. R. Choudhary, D. Zhao, H. Versee, and A. Orso, "Water: Web application test repair," in *Proceedings of the First International Workshop on End-to-End Test Script Engineering*. ACM, 2011, pp. 24–29.

[34] M. Hammoudi, G. Rothermel, and A. Stocco, "Waterfall: An incremental approach for repairing record-replay tests of web applications," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2016, pp. 751–762.

[35] L. S. Pinto, S. Sinha, and A. Orso, "Understanding myths and realities of test-suite evolution," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*. ACM, 2012, p. 33.

[36] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," *Software Testing, Verification and Reliability*, vol. 22, no. 2, pp. 67–120, 2012.

[37] R. Kazmi, D. N. Jawawi, R. Mohamad, and I. Ghani, "Effective regression test case selection: A systematic literature review," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, p. 29, 2017.

[38] S. U. R. Khan, S. P. Lee, R. W. Ahmad, A. Akhunzada, and V. Chang, "A survey on test suite reduction frameworks and tools," *International Journal of Information Management*, vol. 36, no. 6, pp. 963–975, 2016.

[39] C. Catal and D. Mishra, "Test case prioritization: a systematic mapping study," *Software Quality Journal*, vol. 21, no. 3, pp. 445–478, 2013.

[40] A. Zaidman, B. Van Rompaey, S. Demeyer, and A. Van Deursen, "Mining software repositories to study co-evolution of production & test code," in *Software Testing, Verification, and Validation, 2008 1st International Conference on*. IEEE, 2008, pp. 220–229.

[41] A. Rodrigues and A. Dias-Neto, "Relevance and impact of critical factors of success in software test automation lifecycle: A survey," in *Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing*. ACM, 2016, p. 6.

[42] A. Causevic, D. Sundmark, and S. Punnekkat, "Factors limiting industrial adoption of test driven development: A systematic review," in *Software Testing, Verification and Validation (ICST), 2011 IEEE Fourth International Conference on*. IEEE, 2011, pp. 337–346.

[43] D. Rattan, R. Bhatia, and M. Singh, "Software clone detection: A systematic review," *Information and Software Technology*, vol. 55, no. 7, pp. 1165–1199, 2013.

[44] C. K. Roy and J. R. Cordy, "A survey on software clone detection research," *Queens School of Computing TR*, vol. 541, no. 115, pp. 64–68, 2007.

[45] C. K. Roy, M. F. Zibran, and R. Koschke, "The vision of software clone management: Past, present, and future (keynote paper)," in *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on*. IEEE, 2014, pp. 18–33.

[46] T. Xie, D. Marinov, and D. Notkin, "Rostra: A framework for detecting redundant object-oriented unit tests," in *Proceedings of the 19th IEEE international conference on Automated software engineering*. IEEE Computer Society, 2004, pp. 196–205.

[47] D. Kulak and H. Li, "Getting coaching that really helps," in *The Journey to Enterprise Agility*. Springer, 2017, pp. 197–209.

[48] A. Baah, *Agile Quality Assurance: Deliver Quality Software-Providing Great Business Value*. BookBaby, 2017.

[49] J. Iivari and M. Huisman, "The relationship between organizational culture and the deployment of systems development methodologies," *MIS Quarterly*, pp. 35–58, 2007.

[50] J. Iivari and J. Maansaari, "The usage of systems development methods: are we stuck to old practices?" *Information and software technology*, vol. 40, no. 9, pp. 501–510, 1998.

[51] M. Senapathi, M. Drury, and A. Srinivasan, "Agile usage: Refining a theoretical model." in *PACIS*, 2013, p. 43.

[52] K. Beck, *Test-Driven Development: by Example*. Addison-Wesley Professional, 2003.

[53] B. Zeiss, D. Vega, I. Schieferdecker, H. Neukirchen, and J. Grabowski, "Applying the iso 9126 quality model to test specifications," *Software Engineering*, vol. 15, no. 6, pp. 231–242, 2007.

[54] M. Greiler, A. Van Deursen, and A. Zaidman, "Measuring test case similarity to support test suite understanding," *Objects, Models, Components, Patterns*, pp. 91–107, 2012.