Real Challenges in Mobile App Development

Mona Erfani Joorabchi Ali Mesbah Philippe Kruchten
University of British Columbia
Vancouver, BC, Canada
{*merfani, amesbah, pbk*}*@ece.ubc.ca*

***Abstract*—*Context:* Mobile app development is a relatively new phenomenon that is increasing rapidly due to the ubiquity and popularity of smartphones among end-users. *Objective:* The goal of our study is to gain an understanding of the main challenges developers face in practice when they build apps for different mobile devices. *Method:* We conducted a qualitative study, following a Grounded Theory approach, in which we interviewed 12 senior mobile developers from 9 different companies, followed by a semi-structured survey, with 188 respondents from the mobile development community. *Results:* The outcome is an overview of the current challenges faced by mobile developers in practice, such as developing apps across multiple platforms, lack of robust monitoring, analysis, and testing tools, and emulators that are slow or miss many features of mobile devices. *Conclusion:* Based on our findings of the current practices and challenges, we highlight areas that require more attention from the research and development community.**

***Index Terms*—mobile app development; mobile platforms; qualitative study**

I. Introduction

The ubiquity and popularity of smartphones among end-users has increasingly drawn software developers' attention over the last few years. There are currently around 800,000 mobile apps on Apple's AppStore [1] (38% of marketshare [2]), 650,000 on Android Market [3] (52%), 120,000 on Windows Marketplace [4] (3%) and 100,000 on Blackberry AppWorld [5] (6%). Recent estimations indicate that by 2015 over 70% of all handset devices will be smartphones, capable of running mobile apps [6].

As with any new domain, mobile application development has its own set of new challenges, which researchers have recently started discussing [7], [8]. However, most of these discussions are anecdotal in nature. While there are substantial qualitative studies on different areas of software engineering, to the best of our knowledge, no study has been conducted to investigate the challenges that mobile app developers face in practice.

Mobile apps fall broadly into three categories: *native*, *web-based*, and *hybrid* [9], [10]. Native applications run on a device's operating system and are required to be adapted for different devices. Web-based apps require a web browser on a mobile device. Hybrid apps are 'native-wrapped' web apps. A recent survey [11] revealed that developers are mainly interested in building native apps, because they can utilize the device's native features (e.g., camera, sensors, accelerometer, geolocation). Therefore, in this paper we mainly focus on native apps. Henceforth, we use the term 'mobile app' to denote 'native mobile application'.

The goal of our study is to gain an understanding of the current practices and challenges in native mobile app development; we conducted an explorative study by following a Grounded Theory approach, which is a research methodology stemming from the social sciences [12], gaining increasing popularity in software engineering research [13].

Thus, instead of starting with predetermined hypotheses, we set our objective to discover the process and challenges of mobile app development across multiple platforms. To that end, we started by conducting and analyzing interviews with 12 senior mobile app developers, from 9 different industrial companies, who are experts in platforms such as iOS, Android, Windows Mobile/Phone, and Blackberry. Based on the outcome of these interviews, we designed and distributed an online survey, which has been completed by 188 mobile app developers worldwide.

Our results reveal challenges of dealing with multiple mobile platforms during mobile development. While mobile devices and platforms are extensively moving toward fragmentation, the contemporary development process is missing the adaptation to leverage knowledge from platform to platform. Developers currently treat the mobile app for each platform separately and manually check that the functionality is preserved across multiple platforms. Furthermore, mobile developers need better analysis tools in order to track metrics for their apps during the development phase. Additionally, testing is a significant challenge. Current testing frameworks do not provide the same level of support for different platforms and current testing tools do not support important features for mobile testing such as mobility, location services, sensors, or different gestures and inputs.

II. Study Design

The objective of our study is to gain an understanding of the challenges mobile app developers face in practice.

A. Methodology

Considering the nature of our research goal, we decided to conduct a qualitative study by following a Grounded Theory approach [12], [14]. Grounded Theory is best suited when the intent is to learn how people manage problematic situations and how people understand and deal with what is happening to them [15]. It is also useful when the research area has not been covered in previous studies [16] and the emphasis is on new theory generation [17] i.e., understanding a phenomenon.

TABLE I: Interview Participants.

ID	Role	Platform Experience	Software Dev Exp (yr)	Mobile Dev Exp (yr)	Company (Mobile Dev Team Size)	Company's Platform Support
P1	iOS Lead	iOS, Android	6-10	6	A (20)	iOS, Android, Windows, Blackberry
P2	Android Lead	Android, iOS	6-10	6	A (20)	iOS, Android, Windows, Blackberry
P3	Blackberry Lead	Blackberry, iOS, Android	6-10	6	A (20)	iOS, Android, Windows, Blackberry
P4	iOS Lead	iOS	6-10	3-4	B (2-5)	iOS, Android
P5	Android Lead	Android	6-10	3	B (2-5)	iOS, Android
P6	iOS Dev	iOS	4-5	3-4	C (20+)	iOS, Android
P7	Windows Mobile Dev	Windows, Android	10+	2	D (1)	Windows
P8	Android Dev	Android	4-5	2-3	E (2-5)	iOS, Android
P9	Android Lead	Android, iOS, Windows	10+	5-6	F (6-10)	iOS, Android, Windows
P10	iOS Dev	iOS, Android	10+	3	G (1)	iOS, Android
P11	Android Lead	Android, Blackberry	10+	6+	H (1)	Android, Blackberry
P12	iOS Dev	iOS, Windows	10+	2-3	I (2-5)	iOS, Windows

Grounded Theory is gaining increasing popularity in software engineering research [13], [15], [16], [18]–[22].

B. Data Collection and Analysis

Our approach for conducting a Grounded Theory research includes a combination of interviews and a semi-structured survey. The interviews targeted *experts* in mobile app development and the survey was open to the general mobile development community.

Our interviews were conducted in an iterative style, and they are at the core of the data collection and analysis process. At the end of each interview, we asked the interviewees for feedback on our set of questions; what is missing and what is redundant. The analytical process involves collecting, coding and analyzing data after each interview, while developing theory simultaneously. From the interview transcripts, we analyze the data line-by-line, break down interviews into distinct units of meaning (sentences or paragraphs), allocate *codes* to the text and label them to generate *concepts* to these units. Our codes, where appropriate, are taken from the text itself. Otherwise, they are created by the authors to capture the emerging concepts. Furthermore, these concepts are then clustered into descriptive *categories*. They are re-evaluated and subsumed into higher-order categories in order to generate an emerging theory. Theoretical sampling evolves into an ever-changing process, as codes are analyzed and categories and concepts continue to develop [19]. We perform *constant comparison* [12] between the analyzed data and the emergent theory until additional data being collected from the interviews adds no new knowledge about the categories. Thus, once the interviewees' answers begin to resemble the previous answers, a state of *saturation* [23] is reached, and that is when we stop the interviewing process.

Based on the theory emerging from the interview phase, we designed a semi-structured survey, as another source of data, to challenge this theory. Before publishing the survey and making it publicly available, we asked four external people – one senior PhD student and three mobile app developers – to review the survey in order to make sure all the questions were appropriate and easily comprehensible. Most of our survey questions are closed-ended, but there are also a few optional open-ended questions for collecting participants' 'insights' and 'experiences'. The responses to these open-ended questions are fed into our coding and analysis step to refine the results, where applicable. This survey, as distributed to participants, is available online.[1]

C. Participant Demographics

Interviews. We interviewed 12 experts from 9 different companies. Each interview session took on average around 30 minutes. We recorded audio in the interview sessions and then transcribed them for later analysis. Table I presents each participant's role in their company, the mobile platforms they have expertise in, the number of years they have work experience in software development and in mobile app development, the size of the mobile development team, and finally all the mobile platforms that each company supports. Regarding the participants' experience in developing mobile app, five have around 6 years, four have 3-4 years and three have 2-3 years of experience. Five participants are mainly iOS experts, five are Android experts, one is a Windows expert, and finally one is a Blackberry expert.

Survey. Our survey was fully completed by 188 respondents. We released the survey on Dec 13, 2012 to a wide variety of mobile development groups. We targeted the popular Mobile Development Meetup groups, LinkedIn groups related to native mobile development and shared the survey through our Twitter accounts. We kept the survey live for two and a half months. In our attempt to distribute our online survey, it was interesting to see people's reactions; they *liked* our post on LinkedIn groups and gave encouraging comments such as "*I hope it will help to make mobile app developers' lives easier*". The demographics of the participants in the survey are as follows: they are 92% male, 5% female. They come from USA (48%), India (11%), Canada (10%), Israel (5%), The Netherlands (3%), UK (3%), New Zealand (2%), Mexico (2%), and 15 other countries. Regarding their work experience in software development, 52% have more than 10 years, 15% between 6-10 years, 20% between 2-5 years, and 13% less than 2 years. Their experience in native mobile development ranges from: 6% more than 6 years, 19% between 4-6 years, 59% have between 1-3 years, to 16% less than 1 year. The platforms they have expertise in include 72% iOS,

[1] http://www.ece.ubc.ca/~merfani/survey.pdf

65% Android, 26% Windows, 13% Blackberry, and 6% chose others (e.g., Symbian, J2ME).

III. Findings

The findings from our study consist of 4 main *categories*, and 25 subordinate *concepts*. For each concept, appropriate codes and quotes are presented in this section.

In addition to the general challenges faced by mobile developers (Section III-A), two major themes emerged from the study, namely (1) challenges of developing mobile apps across multiple platforms (Section III-B), and (2) current practices (Section III-C) and challenges (Section III-D) of mobile app analysis and testing.

A. General Challenges for Mobile Developers

In this subsection, we present the most prominent general challenges faced by mobile app developers, emerging from our study results.

Moving toward Fragmentation rather than Unification. 76% of our survey participants see the existence of multiple mobile platforms as a challenge for developing mobile apps, while 23% believe it is an opportunity for technology advances that drive innovation.

More than half of the participants mentioned that mobile platforms are moving toward fragmentation rather than unification:

- *Fragmentation across platforms:* Each mobile platform is different with regard to the user interface, user experience, Human Computer Interaction (HCI) standards, user expectations, user interaction metaphors, programming languages, API/SDK, and supported tools.
- *Fragmentation within the same platform:* On the same platform, various devices exist with different properties such as memory, CPU speed, and graphical resolutions. There is also a fragmentation possible on the operating system level. A famous example is the fragmentation on Android devices with different screen sizes and resolutions. Almost every Android developer in both our interviews and survey mentioned this as a huge challenge they have to deal with on a regular basis.

Furthermore, device fragmentation is not only a challenge for development but also for testing. All of our participants believe that platform versioning and upgrading is a major concern; For example, a respondent said: "*at the OS level, some methods are deprecated or even removed*". So developers need to test their apps against different OS versions and screen sizes to ensure that their app works. Subject P5 said they mostly maintain "*a candidate list of different devices and sizes*". P11 explained, "*because we monitor our application from the feedback of the users, we tend to focus on testing on the devices that are most popular.*" Thus, the current state of mobile platforms adds another dimension to the cost, with a wide variety of devices and OS versions to test against. P11 continued, "*right now we support 5 or 6 different (app) versions only because there are different OS versions, and on each of those OS versions we also have 3-4 different screen sizes to make sure the application works across each of the Android versions.*" A respondent stated, "*we did a code split around version 2.3 (Android). So we have two different versions of the applications: pre 2.3 version and post 2.3 version. And in terms of our policy, we made that decision since it is too difficult to port some features*".

Monitoring, Analysis and Testing Support. "*Historically, there has almost been no one doing very much in mobile app testing*", stated P10 and explained that until fairly recently, there has been very little testing, and very few dedicated testing teams. However, that is changing now and they have started to reach out for quality and testing. Automated testing support is currently very limited for native mobile apps. This is seen as one of the main challenges by many of the participants. Current tools and emulators do not support important features for mobile testing such as mobility, location services, sensors, or different gestures and inputs. Our results indicate a strong need of mobile app developers for better analysis and testing support. Many mentioned the need to monitor, measure, and visualize various metrics of their apps through better analysis tools.

Open/Closed Development Platforms. Android is open source whereas iOS and Windows are closed source. Some participants argued that Apple and Microsoft need to open up their platforms. P5 explained: "*We have real challenges with iOS, not with Android. Because you don't have API to control, so you have to jump into loops and find a back door because the front door is locked. Whatever Apple allows is not enough sometimes.*" An example of such lack of control is given: "*to find out whether we are connected to the Bluetooth.*" On the other hand, P9 explained that because Android is open source and each manufacturer modifies the source code to their own desires and releases it, sometimes they do not stick to the standards. A simple example is provided: "*the standard Android uses commas to separate items in a list, but Samsung phones use a semicolon.*" A respondent stated, "*Many Android devices have been badly customized by carriers and original equipment manufacturers.*"

Data Intensive Apps. Dealing with data is tricky for apps that are data intensive. As a respondent explained: "*So much data cannot be stored on the device, and using a network connection to sync up with another data source in the backend is challenging.*" Regarding offline caching in hybrid solutions, P1 said: "*Our apps have a lot of data and offline caching doesn't seem to really work well.*"

Keeping Up with Frequent Changes. One type of challenge mentioned by many developers is learning more languages and APIs for the various platforms and remaining up to date with highly frequent changes within each software development kit (SDK). "*Most mobile developers will need to support more than one platform at some point*", a respondent stated. "*Each platform is totally different (marketplaces, languages, tools, design guidelines), so you need experts for every one of them. Basically, it is like trying to write simultaneously a book in Japanese and Russian; you need a native Japanese and a*

native Russian, or quality will be ugly", explained another respondent. As a result, learning another platform's language, tools, techniques, best practices, and HCI rules is challenging.

Many developers complained about the lack of an integrated development environment that supports different mobile platforms. An exception was P1 who explained: "*Right now we develop in two main platforms: iPhone and Android. That is not really that hard, the native SDKs are pretty mature and they are easy to learn.*"

B. Developing for Multiple Platforms

67% of our interview participants and 63% of our survey respondents have experienced developing the *same* app for more than one mobile platform.

Native vs. Hybrid Mobile Apps. Subjects P1 and P8 support developing hybrid apps. The remaining 10 interviewees are in favour of building pure native apps and believe that the current hybrid model tends to look and behave much more like webpages than mobile applications. P11 argued that "*the native approach offers the greatest features*" and P4 stated "*user experience on native apps is far superior [compared] to a web app.*" In a number of cases the participants had completely moved away from the hybrid to the native approach. A recurring example given is Facebook's recent switch from an `HTML5`-based mobile app to a native one.

On the other hand, P1 argued that "*it really depends on the complexity and type of the application*", for example, "*information sharing apps can easily adopt the hybrid model to push news content and updates across multiple platforms.*"

In the survey, 82% responded having native development experience, 11% have tried hybrid solutions, and 7% have developed mobile web apps. Most respondents are in favour of the native approach: "*Mobile web doesn't feel or look like any of the platforms.*" Others said that: "*HTML5 has much potential and will likely address many of the current problems in the future as it saves development time and cost*"; or: "*Since many big players are investing a lot on HTML5, it may take a big chunk of the front-end side when it becomes stable.*"

Most of the participants argued that when development cost is not an issue, companies tend to develop native apps. Of course it also depends on the application type; where better user experience or device specific features are needed, native seems to be the clear choice.

Lastly, when we asked our participants that whether native app development will be replaced by hybrid solutions or mobile web development due to its challenges, all the interviewees and 70% of survey participants disagreed, and 10% indicated that there will always be a combination of native and hybrid approaches.

Limiting Capabilities of a Platform's Devices. Not all devices and operating systems of a platform have the same capabilities. For instance, Android has different versions and browsers in some of those versions have poor support for `HTML5`. Most of the participants in favour of the hybrid approach believe that once the adaptation is complete (e.g., with mature web browsers in the platforms), there would be more interest from the community for hybrid development.

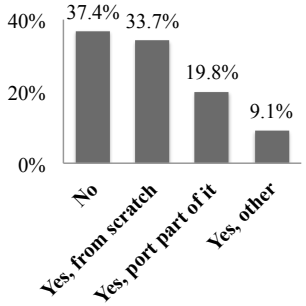

Fig. 1: Have you developed the *same* native mobile app across different platforms?

Reusing Code vs. Writing from Scratch. 67% of our interview participants have tried both methods of writing a native mobile app from scratch for a different platform and reusing some portions of the same code across platforms. The majority stated that it is impossible or challenging to port functionality across platforms and that when code is reused in another platform, the quality of the results is not satisfactory.

Figure 1 shows that out of the 63% survey respondents, who have experienced developing mobile apps across different platforms, 34% have written the *same* app for each platform from scratch, and 20% have experienced porting some of the existing code. A respondent said, "*every platform has different requirements for development and porting doesn't always produce quality*"; or: "*At this moment, I believe that it is best to create the apps from scratch targeting the individual OS.*" P11 argued that "*we ported a very little amount of the code back and forth between Android and Blackberry, but we typically write the code from scratch. While they both use Java, they don't work the same way. Even when basic low levels of Java are the same, you have to rewrite the code.*"

In addition to the differences at the programming language level (e.g., `Objective-C` versus `Java`), P9 elaborated why migrating code does not work: "*A simple example is the way they [platforms] process push messages. In Android, a push message wakes up parts of the app and it requests for CPU time. In iOS the server would pass the data to Apple push server. The server then sends it to the device and no CPU time to process the data is required.*" These differences across platforms force developers to rewrite the same app for different platforms, with no or little code reuse. This is seen as one of the main disadvantages of native app development.

Behavioural Consistency versus Specific HCI Guidelines. Ideally, a given mobile app should provide the same functionality and behaviour regardless of the target platform it is running on. However, due to the internal differences in various mobile devices and operating systems, "*a generic design for all platforms does not exist*"; For instance, P12 stated that "*an Android design cannot work all the way for the iPhone.*" This is mainly due to the fact that HCI guidelines are quite different

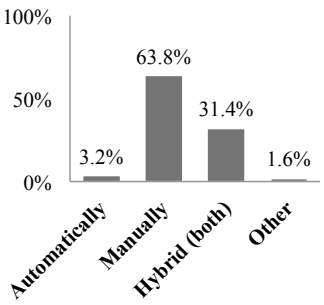

Fig. 2: How are your native mobile apps tested?

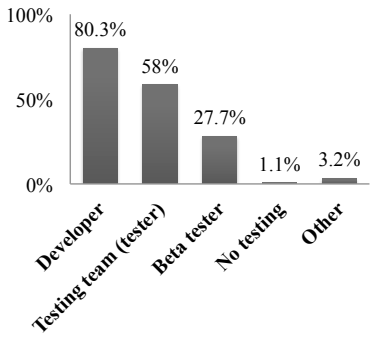

Fig. 3: Who is responsible for testing your native mobile apps?

across platforms, since no standards exist for the mobile world, as they do for the Web for instance. Thus, developers are constantly faced with two competing requirements:

- *Familiarity for platform users:* Each platform follows a set of specific HCI guidelines to provide a consistent look-and-feel across applications on the same device. This makes it easier for end users to navigate and interact with various applications.
- *Behavioural consistency across platforms:* On the other hand, developers would like their application to behave similarly across platforms, e.g., user interaction with a certain feature on Blackberry should be the same as on iPhone and Android.

Thus, creating a reusable basic design that will translate easily to all platforms while preserving the behavioural consistency is challenging. As P9 stated: "*The app should be re-designed per platform/OS to make sure it flows well*"; A respondent put it: "*We do screen by screen design review for each new platform*"; or: "*Different platforms have different strengths and possibilities. It is foolish to try to make the apps exactly the same between platforms*"; and: "*It requires multi-platform considerations at the designing stage and clever decisions should be made where platform-specific design is necessary.*"

Time, Effort, and Budget are Multiplied. Due to the lack of support for automated migration across platforms, developers have to redesign and reimplement most of the application. Therefore, creating quality products across platforms is not only challenging, but also time consuming and costly, i.e."*developing mobile apps across platforms natively is like having a set of different developers per each platform*", stated P11. As a result, "*re-coding against wildly different API sets*" increases the cost and *time-to-market* within phases of design, development, testing, and maintenance, which is definitely a large issue for start-up and smaller companies.

C. Current Testing Practices

As outlined in Subsection III-A, many developers see analysis and testing of mobile apps as an important activity to provide dependable solutions for end-users. Our study results shed light on the current practices of mobile application analysis and testing.

Manual Testing is Prevalent. As shown in Figure 2, 64% of our survey participants test their mobile apps manually, 31% apply a hybrid approach, i.e., a combination of manual and automated testing, and only 3% engage in fully automated testing. P3 explained: "*Right now, manually is the best option. It's kind of like testing a new game, testing on consoles and devices. It is that kind of testing I believe just maybe smaller, but you have to worry about more platforms and versions.*" A respondent stated: "*Organizations, large and small, believe only in manual testing on a small subset of devices*"; and another one said: "*It's a mess. Even large organizations are hard to convince to do automated testing.*"

Developers are Testers. There are different combinations of testing processes and approaches currently taken by the industry. They can be categorized based on a company's size, clients, development culture, testing policy, application type, and the mobile platforms supported. These testing approaches are performed by various people such as developers, testing teams, beta testers, clients, as well as third party testing services. As indicated in Table I, our interviewees' companies vary from small size with 1–2 developers to larger mobile development companies or teams with over 20 developers. As expected, larger companies can afford dedicated testing teams, while in smaller companies testing is mainly done by developers or clients (end-users). Figure 3 depicts the results of our survey with regard to roles responsible for testing. 80% of the respondents indicated that the developers are the testers, 53% have dedicated testing teams or testers, and 28% rely on beta testers.

The majority of the participants, with or without testing teams, stated that after developing a new feature, the developers do their own testing first and make sure it is functional and correct. This is mostly manual testing on simulators and if available on physical devices.

Test the App for Each Platform Separately. Our interviews reveal that our participants treat each platform completely separately when it comes to testing. Currently, there is no coherent method for testing a given mobile app across different platforms; being able to handle the differences at the UI level is seen as a major challenge. Testers write "*scripts that are specific for each platform*", and they "*are familiar with*

the functionality of the app, but are testing each platform separately and individually". We also notice that there are usually separate teams in the same company, each dedicated to a specific platform with their own set of tools and techniques; P6, an iOS developer, said: "*I am not sure about Android, as the teams in our company are so separate and I don't even know what is going on with the other side.*" Responses provided by 63% of our survey participants, who develop the *same* native mobile app for more than one platform, confirmed the interview results, stating: "*The test cases apply to each platform, but they must be implemented uniquely on each platform*", or: "*Same as for one platform, but multiple times*", and: "*I have to do it twice or more depending on how many platforms I have to build it on*", or: "*Treat them as separate projects, as they essentially are, if native. Do testing independently.*"

Levels of Testing. Figure 4 illustrates different levels of testing applied on mobile apps. There is very little automation for different levels of testing, e.g., around 3% for each of GUI, acceptance, and usability testing. P2 noted: "*It is not really well structured or formal what we do. We do some pieces of all of them but the whole testing is a manual process.*"

GUI Testing. More than half of the participants admitted that GUI testing is challenging to automate. P2 said: "*Automated UI testing is labor intensive, and can cause inertia when you want to modify the UI. We have a manual tester, core unit testing, then employ beta field testing with good monitoring.*"

P7 stated: "*Our company has Microsoft products. With Microsoft studio interface you can emulate a lot of sensors for testing GUI where as in Eclipse for Android, you need to click a lot of buttons. You can emulate the position in your phone, but Android doesn't do this.*"

P3 elaborated: "*Blackberry is actually really hard to create test scripts for GUI testing. Because it is not like other platforms, which are touch-based and layout-based. With Blackberry, you have to know what field manager is and it is hard to actually get this information by clicking on buttons. You have to go through the whole array of elements.*"

Some tools were highlighted such as ROBOTIUM [24] and MONKEYRUNNER [25] for Android. A few iOS developers said they have tried MONKEYTALK (formerly called FONEMONKEY) [26] and KIF [27] for GUI testing; P1 stated: "*I find KIF to be a lot more mature than the testing tools provided by Apple but it is still hard to be used for our custom and dynamic applications.*"

Unit Testing. Our study shows that the use of unit testing in the mobile development community is relatively low. Both interview and survey results (See Figure 4) reveal that unit testing for native mobile app is not commonplace yet.

On the one hand, some respondents argued that "*the relatively small size of mobile apps makes unit testing overkill*"; or: "*Deciding whether it's worth writing unit tests or save the time and test manually is always difficult*"; and: "*Complete unit testing to get full coverage is overkill. We only unit test critical code*"; or: "*Small projects with small budgets - the overhead of creating rigorous test plans and test cases would have a serious impact on the budget.*"

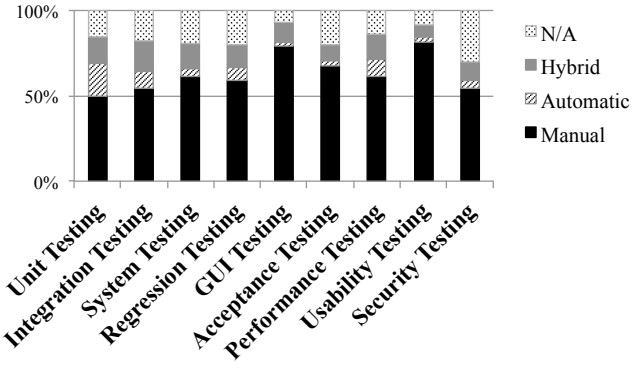

Fig. 4: What levels of testing do you apply and how?

On the other hand, others said that "*the rapidly changing user expectations and technology means unit testing is crucial.*" Our interviewees believe that having a core script for generic features is the best approach in the long term. P12 said: "*Unit tests are still the best. They are easy to run, and provide immediate feedback when you break something.*"

Unit testing seems to be more popular among Android and Windows developers, using `JUnit` and `NUnit`, respectively.

Two iOS participants have tried writing unit tests for iPhone using SENTESTINGKITFRAMEWORK [28], a built-in Xcode tool, as well as XCODE INSTRUMENTS [29]. P1 stated: "*iOS apps are not really built to be unit tested*", P12 argued: "*iOS doesn't make it easy to have test automation*" and a respondent said: "*Apple's Developer testing tools don't play well.*"

Beta Testers and Third Party Testing Services. Beta testing, mostly with TESTFLIGHT [30], seems to be quite popular in mobile app development; although P5 emphasized that "*the beta testers are in the order of dozens not thousands.*" TestFlight automates parts of the process, from deploying the app to collecting feedback. Further, there are many cases in which the clients are responsible for testing, i.e., recruiting beta testers or acceptance testing. P6 explained that they have internal and external client tracking systems: "*Basically we have two bug tracking systems, internal and client tracking system (external). The client create bugs in that tracking system and our testing team try to reproduce bugs to see if it is a valid and reproducible bug. If so they duplicate it in our internal tracking system. Then developers will look at it again.*"

Additionally, some developers rely on third party testing services such as PERFECTOMOBILE [31] and DEVICEANYWHERE [32]. However, "*it is usually too volatile and the tools in many cases support very simple apps. Honestly not really worth the effort*", said one of our interviewees. Other participants' attitudes toward testing services are varied; P12 argued: "*Services should be affordable, and not just report bugs but also provide some documents that indicate how people test the application, and give a high level overview*

of all the paths and possibilities that are tested." Another respondent said: "*Most online testing services charge a very hefty premium even for apps that are distributed for free*"; and: "*It is nice to test an app by a third party, someone who is not the developer. At the same time, just random testing doesn't do the trick. You need to have a more methodical approach but the problem with methodical approaches is that they turn the price up.*" P11 said: "*We don't want to lock in on one specific vendor and tend to use open-source tools, such as JUnit.*" Another problem mentioned is that "*if we want to change something the way we want to, we don't have access to the source code. So we can't change the services of the framework.*"

D. Analysis and Testing Challenges

In this subsection, we present the challenges experienced, by our interview participants and survey respondents, for analyzing and testing native mobile apps.

Limited Unit Testing Support for Mobile Specific Features. Although `JUnit` is used by more than half of the Android participants, many also point out that "*JUnit is designed for stationary applications and it has no interface with mobile specifics such as sensors (GPS, accelerometer, gyroscope), rotation, navigation*". As a result, "*there is no simple way to inject GPS positions, to rotate the device and verify it that way*". P11 explained: "*we are creating a 'map application', which requires users typically being out doors, moving around and navigating, which is not supported by current testing tools.*" Writing mobile specific test scenarios requires a lot of code and is time consuming and challenging. A number of participants indicated that having "*a JUnit type of framework with mobile specific APIs and assertions*" would be very helpful.

Monitoring and Analysis. Both our interview and survey data indicate a strong need of mobile app developers for better analysis and monitoring support. Many mentioned the need to monitor, measure, and visualize various metrics of their apps such as memory management (to spot memory leaks), battery usage (to optimize battery life), CPU usage, pulling/pushing data, and network performance (over various networks, e.g., 2G, 3G, 4G and wireless connections) through better analysis tools. "*A visualization tool such as those hospital monitoring devices with heart rate, blood pressure, etc., would help to gain a better understanding of an app's health and performance*", explained P8.

Handling Crashes. One major problem mentioned in mobile app testing is about crashes, which are often intermittent, non-deterministic, and irrecoverable. It is challenging for developers to capture enough information about these crashes to analyze and reproduce them [33], so that they can be fixed. Many developers in our study found it helpful to have a set of tools that would enable capturing state data as a crash occurs and creating a bug report automatically. P5 stated: "*Dealing with the crashes that are very hard to catch and harder to reproduce is an issue. It would be good that when the crashes happen, system logs and crash logs can be immediately captured and sent to developer over the phone.*"

Emulators/Simulators. Emulators are known to mimic the software and hardware environments found on actual devices whereas simulators only mimic the software environment. Many mobile developers believe that better support is needed to mimic real environments (e.g., network latency, sensors) for testing. Another issue mentioned is that *rooted* simulators and emulators are needed in order to access features outside of the application, such as settings, play store, bluetooth and GPS, which could be part of a test case. Also, performance of emulators is a key factor mentioned by many of our participants. Compared to iOS Simulator, "*Android emulator is very slow. I use my device for testing instead*", said P8.

Missing Platform-Supported Tools. Almost all of the participants mentioned that current tools are weak and unreliable with no or limited support for important features for mobile testing such as mobility, location services, sensors and different inputs. They have experienced many automation failures, or many cases where testing tools actually slowed the development process down substantially.

Some of our participants stated that platform-supported tools are needed, e.g., "*unit testing should be built-in*". A respondent said: "*the platforms have to support it (testing). 3rd party solutions will never be good enough.*", and another one said they need "*strong integrated development environment support*". Some noted that the process will be similar to that for web applications, "*it took years to create powerful tools for analyzing and testing web apps, and we are still not there completely*".

Rapid Changes Over Time. Our interview reveals that requirements for mobile app projects change rapidly and very often over time. This is the reason our participants argued that they have difficulties to keep the testing code up to date. A respondent said: "*changing requirements means changing UI/logic, so GUI and integration tests must be constantly rewritten.*" P1 stated: "*there are some testing tools out there, but we don't use any of them because we can't keep the tests updated for our dynamic apps*". P10 stated that due to rapid changes they have "*time constraints for creating test scripts and performing proper testing.*".

Many Possibilities to Check. An issue mentioned by more than half of the participants is the fact that there are so many different possibilities to test, and places that could go potentially wrong on mobile apps. Thus, "*it is difficult to identify all the usage scenarios and possible use cases while there is a lot of hidden states; for example, enabling/disabling for the location services, and weak and strong network for network connectivity*". P12 finds: "*The relation between apps should be well managed, you might be interrupting other apps, or they might be interrupting yours*". P12 provides an example: "*manage the states when an audio recoding app goes into background.*" Furthermore, a participant argued that based on missing or misleading usage specifications, they should avoid under-coverage (missing test cases) and over-coverage (waste

of time and human resources for testing situations that won't happen in the real world).

App Stores and Usability Testing. Developers have to follow mobile app stores' (e.g., AppStore, Google Play) requirements to distribute their apps to end users. These requirements change often, and developers need a way to test their apps' conformance. "*I would like to have something more robust for me to mimic what the publisher (store) will be doing so that I can catch the error earlier in the development process.*" said a respondent. Additionally, "*pushing the app to individual devices is more complex than necessary*", for instance in iPhone.

At the same time, end-users nowadays have the ability to collectively rank apps on the stores. If users like an app, they download and start using it. If not, they delete it and move on immediately. If they really like it, they rank it high; Or if they really dislike it, they go on social media and complain. Thus, a low-quality release can have devastating consequences for mobile developers. As a result, there is a huge emphasis on usability testing. As P8 explained, "*definitely one of our big challenges is usability testing, which is manual. I do heuristic evaluations personally and then evaluate with real users*". P11 elaborated: "*Our usability testing encompasses a large portion of not only features but also UI.*"

IV. Threats to Validity

Similar to quantitative research, qualitative studies could suffer from threats to validity, which is challenging to assess as outlined by Onwuegbuzie *et al.* [34].

For instance, in codification, the researcher bias can be troublesome, skewing results on data analysis [21]. We tried to mitigate this threat through triangulation; The codification process was conducted by two researchers, one of whom had not participated in the interviews, to ensure minimal interference of personal opinions or individual preferences. Additionally, we conducted a survey to challenge the results emerging from the interviews.

Both the interview and survey questionnaire were designed by a group of three researchers, with feedback from four external people – one senior PhD student and three industrial mobile app developers – in order to ensure that all the questions were appropriate and easily comprehensible.

Another concern was a degree of generalizability. We tried to draw representative mobile developer samples from nine different companies. Thus, the distribution of participants includes different companies, development team sizes, platforms, application domains, and programming languages – representing a wide range of potential participants. Of course the participants in the survey also have a wide range of background and expertise. All this gives us some confidence that the results have a degree of generalizability.

One risk within Grounded Theory is that the resulting findings might not fit with the data or the participants [12]. To mitigate this risk, we challenged the findings from the interviews with an online survey, filled out by 188 practitioners worldwide. The results of the survey confirmed that the main concepts and codes, generated by the Grounded Theory approach, are in line with what the majority of the mobile development community believes.

Lastly, in order to make sure that the right participants would take part in the survey, we shared the survey link with some of the popular Mobile Development Meetup and LinkedIn groups related to native mobile app development. Furthermore, we did not offer any financial incentives nor any special bonuses or prizes to increase response rate.

V. Discussion

We discuss some of the challenges that are worth further investigation by the research and development community.

Same App across Multiple Platforms. A prominent challenge emerging from our study is the fact that developers have to build the same native app for multiple mobile platforms. Although developing for multiple platforms is a recurring problem that is not unique to the mobile world, the lack of proper development and analysis support in the mobile environment exacerbates the challenges.

Opting for standardized cross-platform solutions, such as HTML5, seems to be the way to move forward. However, HTML5 needs to be pushed towards maturation and adoption by major mobile manufactures, which in turn can mitigate many of the cross-platform development problems. Another possible direction to pursue is exploring ways to declaratively construct [35] native mobile applications, by abstracting the implementation details into a model, which could be used to generate platform-specific instances.

Checking Consistency across Platforms. Another related challenge is checking the correctness and consistency of the app across different platforms. One way to tackle this problem is by constructing tools and techniques that can automatically infer interaction models from the app on different platforms. Our recent work reverse engineers a model of iOS applications [36]. Similarly, others [37], [38] are looking into Android apps. The models of the app, generated from different platforms, can be formally compared for equivalence on a pairwise-basis [39] to expose any detected discrepancies. Such automated techniques would drastically minimize the difficulty and effort in consistency checking, since many mobile developers manually "*do screen by screen design review for each new platform*".

Testing Apps for Multiple Platforms. Regarding the testing challenges, follow-up studies could focus on generating test cases for mobile apps. A centralized automatic testing system that generates a (different) test case for each target platform could be a huge benefit. While platform-specific features can be customized, core features could share the same tests. Thus, further research should focus on streamlining application development and testing efforts regardless of the mobile platform.

Testing APIs from App Stores. Mobile developers need better and easier ways of checking their apps' conformance to app stores' guidelines. Currently, after a submission, sometimes they have to wait for considerable amounts of time to receive

feedback from the stores. In order to catch the inconsistencies of their code with a store's guidelines and internal APIs earlier, it would be beneficial if the stores provided a set of testing APIs (e.g., as services), which developers could use to check their code against, before submitting to the stores.

Testing Mobile Specific Features. The existing testing frameworks have serious limitations for testing mobile specific features and scenarios such as sensors (GPS, Accelerometer, gyroscope), rotation, navigation, and mobility (changing network connectivity). As a consequence developers either need to write much test fixture code to assert mobile specific scenarios or opt for manual testing. Thus, creating "*a `JUnit` type of framework with mobile specific APIs and assertions*" would be really beneficial.

Other Challenging Areas. There are also serious needs for (1) rooted emulators that can mimic the hardware and software environments realisticly; (2) better analysis tools, in order to measure and monitor different metrics of the app under development; (3) techniques that would help debugging apps by capturing better state data when unexpected crashes occur.

VI. Related Work

We categorize related work into two classes: mobile application development and ground theory studies in software engineering.

Mobile application development. There have been a number of studies [40]–[44] analyzing different web-based or hybrid mobile app development frameworks. For instance, Palmieri *et al.* [40] report a comparison between four different cross-platform tools (RHODES, PHONEGAP, DRAGONRAD and MOSYNC) to develop applications on different mobile OSs. Huy *et al.* [45] studied and analyzed four types of mobile applications, namely, native, mobile widgets, mobile web, and `HTML5`. Masi *et al.* [9] propose a framework to support developers with their technology selection process for the development of a mobile application, which fits the given context and requirements.

Researchers have recently started discussing [7], [8] some of the challenges involved in mobile app development. However, most of these discussions are anecdotal in nature. Our study, on the other hand, aims at understanding the challenges by interviewing and surveying mobile developers in the field.

Grounded theory studies in software engineering. Many researchers have used a grounded theory approach in qualitative software engineering studies [15]–[22], [46]–[48] in order to understand software development practices and challenges of industrial practitioners [13].

Adolph *et al.* [15] use grounded theory in a field study to understand how people manage the process of software development to "get the job done".

Greiler *et al.* [18] conduct a grounded theory study to understand the challenges involved in Eclipse plug-in testing. The outcome of their interviews with 25 senior practitioners and a structured survey of 150 professionals provides an overview of the current testing practices, a set of barriers to adopting test practices, and the compensation strategies adopted because of limited testing by the Eclipse community.

Coleman *et al.* [17], [19] adopt the grounded theory methodology to report on the results of their study of how software processes are applied in the Irish software industry. The outcome is a theory that explains when and why software process improvement is undertaken by software developers.

Through a grounded theory approach, Sulayman *et al.* [20] perform interviews with 21 participants representing 11 different companies, and analyze the data qualitatively. They propose an initial framework of key software process improvement success factors for small and medium Web companies.

Wiklund *et al.* [49] report a case study on factors that contribute to inefficiencies in use, maintenance, and development of automated testing.

Kasurinen *et al.* [48] discuss the limitations, difficulties, and improvement needs in software test automation for different types of organizations. They surveyed employees from 31 software development organizations and qualitatively analyzed 12 companies as individual cases. They found that 74% of surveyed organizations do not use test automation consistently.

To the best of our knowledge, our work is the first to report a qualitative field study targeting mobile app development practices and challenges.

VII. Conclusions

Our study has given us a better, more objective understanding of the real challenges faced by the mobile app developers today, beyond anecdotal stories.

Our results reveal that having to deal with multiple mobile platforms is one of the most challenging aspects of mobile development. Since mobile platforms are moving toward fragmentation rather than unification, the development process cannot leverage information and knowledge from a platform to another platform. When the 'same' app is developed for multiple platforms, developers currently treat the mobile app for each platform separately and manually check that the functionality is preserved across multiple platforms. Also creating a reusable user-interface design for the app is a trade-off between consistency and adhering to each platform's standards. Our study also shows that mobile developers need better analysis tools to measure and monitor their apps. Also, testing is a huge challenge currently. Most developers test their mobile apps manually. Unit testing is not common within the mobile community and current testing frameworks do not provide the same level of support for different platforms. Additionally, most developers feel that current testing tools are weak and unreliable and do not support important features for mobile testing such as mobility (e.g., changing network connectivity), location services, sensors, or different gestures and inputs. Finally, emulators seem to lack several real features of mobile devices, which makes analysis and testing even more challenging.

Acknowledgements

We are grateful to all the participants of our study (interviews and the survey). This work was supported in part by the

Institute for Computing, Information and Cognitive Systems (ICICS) at the University of British Columbia (UBC).

REFERENCES

[1] "App Store Metrics," http://148apps.biz/app-store-metrics/.

[2] "Smartphone Platform Market Share," http://www.comscore.com/Insights/Press_Releases/2013/3/comScore_Reports_January_2013_U.S._Smartphone_Subscriber_Market_Share.

[3] "Android Market Stats," http://www.appbrain.com/stats/.

[4] "Windows Marketplace," http://windows-phone.co/tag/windows-phone-apps/.

[5] "Blackberry AppWorld," http://drippler.com/rim/blackberry_playbook_lte#!295631.

[6] Berg Insight, "The mobile application market," http://www.berginsight.com/ReportPDF/ProductSheet/bi-app1-ps.pdf.

[7] J. Dehlinger and J. Dixon, "Mobile application software engineering: Challenges and research directions," in *Proceedings of the Workshop on Mobile Software Engineering*. Springer, 2011, pp. 29–32.

[8] A. I. Wasserman, "Software engineering issues for mobile application development," in *FSE/SDP workshop on Future of software engineering research*, ser. FoSER'10. ACM, 2010, pp. 397–400.

[9] E. Masi, G. Cantone, M. Mastrofini, G. Calavaro, and P. Subiaco, "Mobile apps development: A framework for technology decision making," in *Proceedings of International Conference on Mobile Computing, Applications, and Services.*, ser. MobiCASE'4, 2012, pp. 64–79.

[10] "Native, web or hybrid mobile-app development," IBM Software, Thought Leadership White Paper. [Online]. Available: http://www.computerworld.com.au/whitepaper/371126/native-web-or-hybrid-mobile-app-development/download/

[11] "Voice of the Next-Generation Mobile Developer, Appcelerator / IDC Q3 2012 Mobile Developer Report," http://www.appcelerator.com.s3.amazonaws.com/pdf/Appcelerator-Report-Q3-2012-final.pdf.

[12] B. Glaser and A. Strauss, *The discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine Transaction, 1967.

[13] S. Adolph, W. Hall, and P. Kruchten, "Using grounded theory to study the experience of software development," *Empirical Softw. Engg.*, vol. 16, no. 4, pp. 487–513, 2011.

[14] J. W. Creswell, *Qualitative inquiry and research design : choosing among five approaches (2nd edition)*. Thousand Oaks, CA: SAGE, 2007.

[15] S. Adolph, P. Kruchten, and W. Hall, "Reconciling perspectives: A grounded theory of how people manage the process of software development," *J. Syst. Softw.*, vol. 85, no. 6, pp. 1269–1286, 2012.

[16] K. Karhu, T. Repo, O. Taipale, and K. Smolander, "Empirical observations on software testing automation," in *Proceedings of the International Conference on Software Testing Verification and Validation (ICST)*. IEEE Computer Society, 2009, pp. 201–209.

[17] G. Coleman and R. O'Connor, "Investigating software process in practice: A grounded theory perspective," *J. Syst. Softw.*, vol. 81, no. 5, pp. 772–784, 2008.

[18] M. Greiler, A. van Deursen, and M.-A. Storey, "Test confessions: a study of testing practices for plug-in systems," in *Proceedings of the International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 2012, pp. 244–254.

[19] G. Coleman and R. O'Connor, "Using grounded theory to understand software process improvement: A study of Irish software product companies," *Inf. Softw. Technol.*, vol. 49, no. 6, pp. 654–667, 2007.

[20] M. Sulayman, C. Urquhart, E. Mendes, and S. Seidel, "Software process improvement success factors for small and medium web companies: A qualitative study," *Inf. Softw. Technol.*, vol. 54, no. 5, pp. 479–500, 2012.

[21] V. Kettunen, J. Kasurinen, O. Taipale, and K. Smolander, "A study on agility and testing processes in software organizations," in *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA)*. ACM, 2010, pp. 231–240.

[22] A. C. C. Franca, D. E. S. Carneiro, and F. Q. B. da Silva, "Towards an explanatory theory of motivation in software engineering: A qualitative case study of a small software company," *2012 26th Brazilian Symposium on Software Engineering*, vol. 0, pp. 61–70, 2012.

[23] B. Glaser, *Doing Grounded Theory: Issues and Discussions*. Sociology Press, Mill Valley, California, 1998.

[24] "Robotium," http://code.google.com/p/robotium/.

[25] "MonkeyRunner," http://developer.android.com/tools/help/monkeyrunner_concepts.html.

[26] "MonkeyTalk for iOS & Android," http://www.gorillalogic.com/testing-tools/monkeytalk.

[27] "KIF iOS Integration Testing Framework," https://github.com/square/KIF.

[28] "SenTestingKit Framework," http://cocoadev.com/wiki/SenTestingKit.

[29] "About Instruments," https://developer.apple.com/library/mac/#documentation/developertools/Conceptual/InstrumentsUserGuide/Introduction/Introduction.html.

[30] "TestFlight," https://testflightapp.com/.

[31] "Perfecto Mobile," http://www.perfectomobile.com/.

[32] "DeviceAnywhere," http://www.keynotedeviceanywhere.com/.

[33] P. Zhang and S. Elbaum, "Amplifying tests to validate exception handling code," in *Proceedings of the International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 2012, pp. 595–605.

[34] A. Onwuegbuzie and N. Leech, "Validity and qualitative research: An oxymoron?" *Quality and Quantity*, vol. 41, pp. 233–249, 2007.

[35] Z. Hemel and E. Visser, "Declaratively programming the mobile web with Mobl," in *Proceedings of Intl. Conf. on Object oriented programming systems languages and applications (OOPSLA)*. ACM, 2011, pp. 695–712.

[36] M. Erfani Joorabchi and A. Mesbah, "Reverse engineering iOS mobile applications," in *Proceedings of the Working Conference on Reverse Engineering (WCRE)*. IEEE Computer Society, 2012, pp. 177–186.

[37] C. Hu and I. Neamtiu, "Automating GUI testing for Android applications," in *Proceedings of the 6th International Workshop on Automation of Software Test*, ser. AST '11. ACM, 2011, pp. 77–83.

[38] W. Yang, M. R. Prasad, and T. Xie, "A grey-box approach for automated GUI-model generation of mobile applications," in *Proceedings of the International Conference on Fundamental Approaches to Software Engineering (FASE)*. Springer-Verlag, 2013, pp. 250–265.

[39] A. Mesbah and M. R. Prasad, "Automated cross-browser compatibility testing," in *Proceedings of the International Conference on Software Engineering (ICSE)*. ACM, 2011, pp. 561–570.

[40] M. Palmieri, I. Singh, and A. Cicchetti, "Comparison of cross-platform mobile development tools," in *Intelligence in Next Generation Networks (ICIN), 2012 16th International Conference on*, 2012, pp. 179 –186.

[41] M.-C. Forgue and D. Hazaël-Massieux, "Mobile web applications: bringing mobile apps and web together," in *Proceedings of the 21st international conference companion on World Wide Web*, ser. WWW '12 Companion. ACM, 2012, pp. 255–258.

[42] T. Paananen, "Smartphone Cross-Platform Frameworks," Bachelor's Thesis., 2011.

[43] S. Diewald, L. Roalter, A. Möller, and M. Kranz, "Towards a holistic approach for mobile application development in intelligent environments," in *Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia*, ser. MUM '11. ACM, 2011, pp. 73–80.

[44] Y.-W. Kao, C.-F. Lin, K.-A. Yang, and S.-M. Yuan, "A cross-platform runtime environment for mobile widget-based application," in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Conference on*, 2011, pp. 68 –71.

[45] N. P. Huy and D. vanThanh, "Evaluation of mobile app paradigms," in *Proceedings of the International Conference on Advances in Mobile Computing and Multimedia*, ser. MoMM. ACM, 2012, pp. 25–30.

[46] S. P. Ng, T. Murnane, K. Reed, D. Grant, and T. Y. Chen, "A preliminary survey on software testing practices in Australia," in *Proceedings of the Australian Software Engineering Conference*. IEEE Computer Society, 2004, pp. 116–125.

[47] D. Falessi, M. A. Babar, G. Cantone, and P. Kruchten, "Applying empirical software engineering to software architecture: challenges and lessons learned," *Empirical Softw. Engg.*, vol. 15, no. 3, pp. 250–276, 2010.

[48] J. Kasurinen, O. Taipale, and K. Smolander, "Software test automation in practice: empirical observations," *Adv. Soft. Eng.*, vol. 2010, pp. 4:1–4:13, 2010.

[49] K. Wiklund, S. Eldh, D. Sundmark, and K. Lundqvist, "Technical debt in test automation," in *Proc. International Conference on Software Testing, Verification and Validation (ICST)*. IEEE Computer Society, 2012, pp. 887–892.