Impediments for Automated Testing - An Empirical Analysis of a User Support Discussion Board

Kristian Wiklund*†, Daniel Sundmark†, Sigrid Eldh*‡, Kristina Lundqvist†
*Ericsson AB, SE-164 80 KISTA, Sweden
{kristian.wiklund,sigrid.eldh}@ericsson.com
†School of Innovation, Design and Engineering, Mälardalen University, Sweden
{daniel.sundmark,kristina.lundqvist}@mdh.se
‡Karlstad University, Karlstad, Sweden

***Abstract*—To better understand the challenges encountered by users and developers of automatic software testing, we have performed an empirical investigation of a discussion board used for support of a test automation framework having several hundred users. The messages on the discussion board were stratified into problem reports, help requests, development information, and feature requests. The messages in the problem report and help request strata were then sampled and analyzed using thematic analysis, searching for common patterns. Our analysis indicate that a large part of the impediments discussed on the board are related to issues related to the centralized IT environment, and to erroneous behavior connected to the use of the framework and related components. We also observed a large amount of impediments related to the use of software development tools. Turning to the help requests, we found that the majority of the help requests were about designing test scripts and not about the areas that appear to be most problematic. From our results and previous publications, we see a clear need to simplify the use, installation, and configuration of test systems of this type. The problems attributable to software development tools suggest that testers implementing test automation need more skills in handling those tools, than historically has been assumed. Finally, we propose that further research into the benefits of centralization of tools and IT environments, as well as structured deployment and efficient use of test automation, is performed.**

***Index Terms*—software testing;test automation;test tools;empirical software engineering**

I. Introduction

Software development organizations use automated test tools for many reasons, such as gaining speed [23], keeping the tests repeatable [23], conserving resources [21], and managing more testing in less time [29]. If done right, automation can be a very efficient way to reduce the effort involved in the development by eliminating or minimizing repetitive tasks and reducing the risk for human errors [12][18]. Automated testing can be considered to be a corner-stone in the modern software development process, where the use of test automation is increasing rapidly due to the introduction of methods such as test-driven development [11] and continuous integration [28]. The effect is that automated testing is not only used by specialist testers, but also by software developers in general.

Automated software testing is very tool-dependent, as the entire test flow from environment setup, via test execution, result analysis to teardown should be automated for the maximum benefits. Consequently, the complexity and total size of automated test systems and test scripts for a product are commonly in the order of, or even greater, than the complexity and size of the product being tested [3][25][14]. This complexity can, if unmanaged, lead to expensive setbacks for the automation effort, and in some cases, to disillusioned employees and abandoned automation [1]. The purpose of a test tool is to enable the tester apply his or her testing skills to the task at hand in a more efficient way than if done manually, and if this fails, the tool has no value [23].

The focus of this study was to investigate *impediments* experienced by users of the test automation framework, in order to further the understanding about what issues a tool user may encounter. Impediments are defined as "anything that prevents a team member from performing work as efficiently as possible" [27].

The study was performed in a organization that uses a shared test automation framework for its automated testing. When automating a test, the test framework user designs a test script using the functionality provided by the framework to perform a number of common tasks, such as configuring the system under test, controlling stimuli generators and monitoring equipment, generating a verdict, and sending the results to a database.

To identify user impediments related to use of the framework, we have analyzed the contents of an intraweb discussion board used for framework support questions, announcements, and development discussions. Users of the framework post messages to it, containing questions or information, making it possible for others, such as management and the framework developers, to see what is happening and to respond to and act on the messages. This discussion board is part of the official support channel, effectively making it the "ticket" system and entry point to first line support for the framework, and as such, a suitable source for information about what types of impediments, issues, and problems are encountered by a framework user.

The main contribution of this paper is an overview of the issues experienced by test developers in a large industrial software development organization, together with a narrative connecting these issues to published research. Very little information is previously available about real-world experiences with automated testing, in particular "negative" experi-

978-0-7695-5185-2/14 $31.00 © 2014 IEEE
DOI 10.1109/ICST.2014.24

ences [23].

Through our analysis, we found that mistakes by the users of the test framework, together with issues related to the centralized IT environment, formed the largest class of impediments related to using the test framework. The impediments related to the IT environment were primarily caused by the centralization and complexity of that system. The number of IT environment impediments would most likely have been lower in a less centralized or less secure environment. Turning to the user mistakes, the majority of the impediments were connected to configuration of the test system and test environments, such as using the wrong parameters or the wrong baseline of the tools. Many of the mistakes were recurring in nature, with the same basic problem occurring over and over again.

The paper is organized as following: In Section II we provide an overview of related work, followed by the study design including the codebook and threats to validity in Section III. Section IV provides an overview of the results from the analysis. Our interpretation of the results is discussed in Section V. Finally, Section VI contains our recommendations to practitioners, followed by conclusions and future work in Section VII

II. Related Work

The presence of impediments related to tool usage has been reported earlier in literature as well. Kasurinen *et al.* [15], exploring problems in testing practices, reported that "complicated tools cause errors" and proposed that the selection of tools should be focused on usability and configurability of the tools. Martin *et al.* [16] touches upon this as well, and reports of a situation where the configuration a complicated test environment consumes almost the entire planned test period. There are also parallels to our previous study [32], with different context, where we proposed that generalist engineers using a tool will experience problems if the tool is too configurable, and that tools must be designed in ways that minimize lead time to get them up and running. The observation in that case was that the subjects, as generalist software developers, preferred ease of use to flexibility. Karlström *et al.* [13], made a similar observation and reported that "test environments appear to be taken for granted by developers".

Berner *et al.*[1] in their report on findings from five industrial projects, proposes that automation of software installation and test configuration sometimes have more potential than the automation of test execution. Related to this, Pinheiro *et al.* [22] explored configuration testing of test environments through action research in an industrial context, and reported orders of magnitude in improvement to speed and trust in the environment when environment testing was in place. Technically, there are few similarities in their studied product and our studied product, but there are enough similarities to the impediments encountered in our study to make it relevant.

III. Study Design

A. Research Questions

RQ1: What type of tool-related impediments are encountered by the users of a test automation framework?

RQ2: What type of help do users of a test automation framework ask for?

B. Context

The unit of analysis is a test automation framework based on the TestNG open source Java unit test framework [2]. In addition to TestNG, it consists of a number of in-house developed components that provide services to the test script developers. Included in these services are mechanisms to manage test execution, store verdicts in test result databases, handle logs, communicate with the system under test, and control lab equipment. Typically, a test require a combination of IP traffic generators, power switches to turn equipment on and off, spectrum analyzers, and other equipment needed to simulate a real environment and monitor the execution of the system under test.

The test software consists of three tiers in addition to TestNG: one corporate-wide part containing generic shared components, one system under test (SUT)-specific part that adds modules for SUT interfacing and SUT-specific test equipment, and finally the test scripts themselves.

The system under test-specific part is developed by two teams of developers, and has an agile product owner, as well as funding and structured requirement and issue handling. In addition to developing the framework, the teams also provide user support, and on-demand classroom training to the developers and testers working with the system under test.

A schematic overview of the components in the test system is shown in Figure 1. The same version and configuration of the test framework is used both on the developer workstations and on the test execution servers. The equipment is remote-controlled through the corporate local area network.

The system under test is a networked real-time embedded system, and the current automated test scope using the studied framework is limited to the machine-machine interfaces only. There are automated GUI tests for monitoring software executing on workstations, but those are implemented through a different mechanism, and are not included in the results of this study.

The shared components of the framework has around 2000 users, while the specific instance and extension of the framework that we have studied has around 200 active users, who are performing automated tests on primarily embedded telecommunication software. The tests are started both manually and through an automated continuous integration mechanism [28] based on the Jenkins engine. The type of testing performed ranges from subsystem functional integration testing to system performance and stability testing.

The IT environment, such as work stations, build servers, test execution servers, and web servers, is managed by a

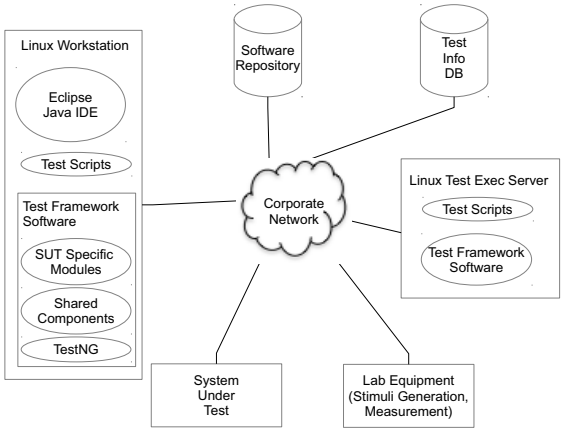

Fig. 1. Schematic overview of the test setup, showing the relations between different functional areas

central corporate function. From the point of view of the users of the test framework it is effectively outsourced to an IT service provider. The central IT function is responsible for all aspects of the IT environment, from data center real estate to day-to-day operations of the networks and test bed hardware. The standardized IT environment is accessed via remote terminal servers or secure shell login from the users' ordinary office laptops.

The studied instance of the framework and its extensions has been used for several years by a smaller group of engineers. During the last half of 2012, as part of an agile transformation and an associated reorganization, there was an influx of new users as well as a migration from two separate discussion boards, to the one analyzed.

The data we have used in our analysis is available through an intranet discussion board, which is part of the development collaboration area for the studied test framework. The collaboration area is a so-called "forge" for inner source [31] and membership in the collaboration area is required to be allowed to push changes to both the framework and the test scripts using the framework.

To initiate a discussion or a support request, a user of the discussion board submits a message in a web interface. This message is then visible to the other users of the discussion board, and it is possible to respond to it through comments in other messages. There is also an email gateway, making it possible to receive and send posts through normal email.

The primary support channel to the development team is through the discussion board, it serves as the "ticket system" for the first-line support, which means that it *should* contain a majority of the issues. However, according to information from the framework developers, this is not the case, as some framework users elect to contact a developer directly via instant messaging (IM) instead of using the board, as they consider the lead-time for support to be too long when using the discussion board. The consequences of this is that any issues solved in this way are absent from our analysis as well as invisible to the rest of the organization, limiting the use of the discussion board as a knowledge base.

C. Method Overview

The study was performed by qualitatively and quantitatively analyzing the messages posted to the support discussion board. The data was classified using thematic analysis [7]. The quantitative results of this analysis are presented in Section IV, and a discussion of the results in context of previously published literature is presented in Section V. The details of the study design, such as coding and sampling, are available in the following sections.

D. Classification of the Data

The classification of the data was performed in four steps:

1) After acquiring the data, all messages were grouped into their respective thread by an automated tool.
2) The first post of all threads was read to group them into their respective stratum, using the categories described in Section III-D1, "Discussion Thread Types", below. A simple PHP web interface was used to facilitate this work and keep track of the progress.
3) A random sampling was performed for each stratum, as described in Section III-E.
4) The randomly selected posts were analyzed in detail to find out what types of issues and questions were present. As part of this analysis, the detailed codes for each stratum were derived iteratively, identifying common themes in the data [24]. This was performed as a thematic analysis [7] using the nVivo qualitative analysis software suite [20], identifying the themes shown in Table I. The themes are further described in Section III-D2 and Section III-D3. The researcher performing the majority of the classification of the posts had significant domain knowledge about the test framework, which made it possible to form an opinion about the causes. When the cause was unclear, or if there was more than one reasonable cause, it was categorized as "unknown cause".

1) Discussion Thread Types: The discussion thread types formed the initial codes for the analysis and were defined prior to the analysis.

- Included in this study are
  - **Problem Reports** - The execution fails for some unexpected reason. Problem reports have a reactive nature, they are written after something stops working or behaves in an unexpected way. As such, they are impediments for the progress, since time and effort is consumed to resolve them.
  - **Help Requests** - Questions about something that the user wants to know how to do, or how to use. As opposed to problem reports, a request for help is a proactive search for knowledge, and usually do not cause the same level of stress for the user as a problem report. He or she knows that there is a gap of knowledge that needs filling. The underlying cause

for a help request may become an impediment if the help request is not answered in a timely fashion.

- Not included in this study are
  - **Feature Requests** - I want to do something that is not supported by the tool. The difference to a problem report is that a problem report targets behavior that is intended to be in place and work, while a feature request is about adding something completely new.
  - **Development Info** - Code review requests for the framework, deliveries of the framework, discussions on technical implementation details for framework features, notice of planned downtime, et cetera.

2) Problem Report Types: The problem report types were identified and refined during the thematic analysis of the data.

1) **Test Framework**
   - *Implementation Error* A problem related to issues with the test framework implementation. This can be programming or systemization mistakes, version mismatches between the components delivered as part of the framework package, or other problems that are caused by a deviation between the implementation and the intent of the developers of the framework.
   - *Documentation Error* This is a problem with the user documentation that caused the user of the framework to perform some kind of erroneous action.
   - *Global Configuration* This is a problem caused by global and site-level configuration and parameterization of the test framework or test tools, such as IP addresses needed to communicate with the SUT or lab equipment.
2) **User Behavior**
   - *Test Script Error* A programming mistake in a test script invoked by the user.
   - *User Configuration* A problem related to configuration and parameterization of the framework done by the user. Examples of this type of impediments are:
     - *Tool configuration* - using the wrong parameters for a complex lab tool
     - *Eclipse Issues* - Breaking one's workspace or using the wrong configuration for build or version control.
     - *Test Environment Baseline* - The user had selected the wrong version of a component or dependency for the framework. This includes using wrong Java version, or wrong Java class-path.
   - *Access Rights* - Supplying the wrong credentials, e.g. using wrong SSH key, and having wrong UNIX file permissions on files. Please note the difference to not having the proper access rights, as that is addressed in "IT Environment" below.
   - *Lab Equipment* This is a problem with some type of lab equipment apart from the test framework. Examples of lab equipment are various types of stimuli generators and measurement instrument.
   - *Tool Usage and Framework invocation* The user is trying to use the equipment in a creative way and fail, or lack sufficient skills for performing a task and perform it in the wrong way. This is classified per main tool group in the diagrams and table.
3) **SUT Issue** A problem related to the system under test, either an identified anomaly, or a misunderstanding of functionality.
4) **Unknown Cause** Either no cause could be identified, or there were multiple options for why the impediment occurred. There were also a few messages that were written in an extremely cryptic way that the researchers were unable to understand.
5) **Wrong Forum** The poster asked questions about things completely out of scope for the discussion board and was directed to ask in another forum or seek other ways to get support.
6) **IT Environment** This is a problem related to the IT environment, including networks, servers, workstations, and the configuration and installation of standard applications on that type of equipment. The compilers, source code repositories, and other components of the development environment are included here, as being standard components provided by the IT service organization. None of the issues in this category can be handled by staff in the studied organization, all IT environment issues have to be handled via the IT support.

   Examples of IT Environment-related impediments are:
   - *Access rights* - the user did not have the sufficient permissions to perform a task. The majority of these are related to the rules and mechanisms governing accesses and credential in the centralized IT environment. These issues would likely not occur in a less secure IT environment, or would have been solved locally by the users themselves.
   - *Network configuration* - it was not possible to access something due to firewall or routing configuration. The majority of this type of issues were related to "change weekends" where planned work was done to the environment.
   - *Resource shortage* - Overloaded servers, full disks, and other issues that primarily are solved by adding more hardware to the IT environment. It can be discussed if a full disk is a user behavior problem, such as excessive logging, or an IT environment problem. It is, however, something that the users experience as a limitation in the IT environment.
   - *Outage* - Some kind of outage, unknown what. Typical symptoms are "NFS server not responding", failed logins, and intermittent loss of connection when using SSH or terminal servers. The majority of this type of issues were transient or related to "change weekends" where planned work was done to the environment. A large part of the issues were connected to the use of terminal servers.

3) Help Request Types: The help request messages were classified into the same themes as the problem reports, changing the definition of the theme into a question about the area instead of a problem with the area. For example, for problem reports we define the "user configuration" theme as "a problem related to configuration and parameterization of the framework done by the user". Following this, for help requests we define the "user configuration" theme as "a question about how to configure and parameterize the framework to be able to do what the user wants to do". In addition to the themes derivable from the problem reports, we also identified the following themes related to user behavior:

- *API Question* - The user wants to know if something is supported by the test framework API, or how to use it in some specific way.
- *Want Code Example* - Looking for an example test script or similar to be able to learn how to do something
- *Test Script Design* - This addresses things such as "how do I structure my test scripts in the best way to achieve my objective".
- *Unix Usage* - Generic Unix Questions

E. Sampling

To keep the analysis effort at a reasonable level, we performed a sampling of the discussion board topics. To do this, the discussion forum posts were assigned codes according to the codebook in Section III-D, then the posts were stratified according to the codes. Each stratum was sampled using probability sampling [24] with random numbers generated by the *random.org* true random number service [9].

Basic survey statistics as described by Biemer and Lyberg [5] was used to calculate the necessary number of samples per stratum, as the investigation can be considered to be functionally equivalent to a survey. The purpose of this part of the study was to seek the proportions of the population that correspond to specific issues, where the population is all impediments related to the framework, and the sampling frame is the messages on the discussion board. Sampling makes the analysis manageable, but introduces a sampling error into the results. This sampling error can be visualized by calculating a confidence interval for the results [10], as shown in the bar charts illustrating the results.

Two simple equations [5] were used to estimate the number of samples required for a specified error margin d at a 95% confidence interval. The equations estimate the required number of samples for a 50% proportion, which is the worst-case situation. Applying Equation 1a and Equation 1b, we found that we should use at least 257 samples for an error margin of 5% for the problem report stratum, and at least 178 samples for an error margin of 5% for the help request stratum.

It should be noted that this was the initial sample size estimate, and that the final confidence intervals [10] shown in the figures were calculated using the actual data after sampling.

$$n_0 = \frac{1}{d^2} \tag{1a}$$

$$n = \frac{n_0}{1 + \frac{n_0}{N}} \tag{1b}$$

F. Validity

In this section we describe the various threats of validity that were considered during the study. The purpose of this is both to be transparent about the trustworthiness of the result, and to enable the researchers to the design the study to minimize the potential bias caused by these threats [26].

- **Construct validity** addresses if the study measures what we intend it to measure [24]. Our intent was to find out what the cause of the majority of the problems with the studied test framework is. To do this, we have analyzed the discussion board used for "first line support" for the framework, including only "publicly visible" problem reports. We have not considered other support paths, such as asking the neighbor, calling the developer, or solving the problem without assistance. Since the sample frame is limited to the posts on the board, this introduces the risk of a non-sampling error in the results [4].
  A stratified random sampling [5] was performed, which introduces a sampling error. The selection of the sample size was done to achieve an error margin of 5% or less, as shown in Section III-E. The actual sampling error margins are valid for a confidence interval [10] of 95%, and are illustrated in the diagrams with error bars.
  The coding and classification of the discussion board topics was performed by the researchers, and not as part of the process of solving the issues as described by for example Mays *et al.* [17]. Hence, there is a risk for misinterpretation of the issues, introducing an observer error into the results.
  In a few cases, there have been multiple issues discussed in the same thread, starting with a "me too" observation from someone else than the original poster, then it turns out that it was two different root causes. It is easy to imagine situations where different types of questions are mixed in the same thread too. If there is a large number of that type of posts in the source material, that have fallen outside our sampling, it may skew the statistics.
  Not all problem solving occurs via the discussion board, as reported in Section III-B, parts of the problem solving has been reported to occur via instant messaging to the tool development team. This means that information about those issues are not included in the discussion board contents.
- **Internal validity** concerns the analysis of the data. The causality and correlation implied and stated in the study is in the qualitative part of the paper. We link our qualitative conclusions as clearly as possible to our data as well as to the studies that support our discussion.
- **External validity** addresses generalization of the findings [26]. A limitation of this study is that it addresses

only one test framework in one organization. We have provided context information in Section III-B to enable comparison to other organizations and studies. In doing this, we have provided as much information as possible without compromising our agreements with the studied business. It must be noted that the context is complex, and that we provide an overview of one case that can be viewed in parallel with other research in the same area. In Section V, consisting of the narrative part of this paper, we discuss some of these parallels.

- **Reliability** addresses the repeatability of the study [26]. We have provided the definition of the codes used for the analysis, and how the work was performed. The main threat to the reliability is the interpretation of these codes by the researchers, combined with the domain knowledge needed to do the analysis, which makes it possible that other researchers not familiar with the studied framework would reach different conclusions. The coding influences the results, as the interpretation of different codes in relation to each other may depend on the persons performing the analysis.

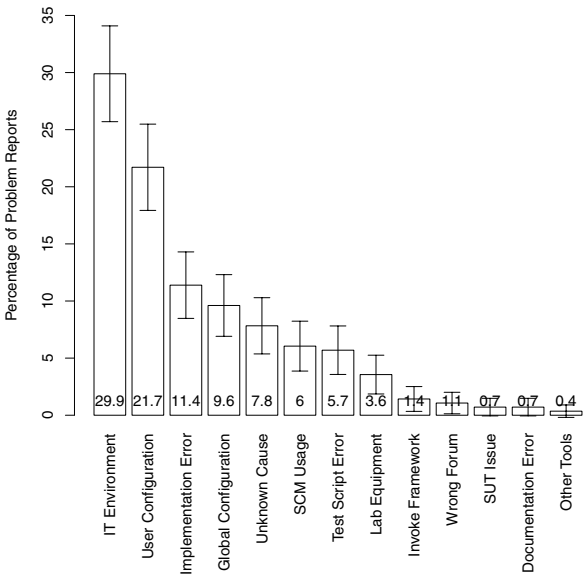

Fig. 2. Issues per Group of Causes

IV. Results

The analyzed data consists of 8796 posts, belonging to 1656 topics, from 2010-02-25 to 2013-01-04. 189 users were involved in the discussions, posting at least one post. The distribution of posts per user was highly uneven, the top three posters were responsible for 48.6% of the total number of posts on the board, and the fourth most active user had posted approximately one fourth as much as the top user. The majority of the posts by the top posters were in response to questions or comments by other users.

We found that 712 (43%) of the forum topics were related to problem reports, and that 315 (19%) of the topics were related to users asking for help in general. For the two classes of messages not analyzed in this study, 530 (32%) of the topics contained development info, and 99 (6%) contained feature requests.

Research question RQ1 was addressed through further analysis of the problem reports. In doing this, we found that using and setting up the framework and test system was the main source of impediments reported on the board, together with impediments related to the IT environment. The distribution of problem reports over themes is shown in Table I. The subthemes are illustrated in Figure 2 together with their 95% confidence intervals.

Research question RQ2 was addressed by analyzing and coding the "help request" posts from the message board. Through this analysis, we found that the main area about which the users ask for help about is API usage questions, as shown in Figure 3, followed by general design questions about how to design a test script. Table I compares the help request distribution to the problem report distribution.

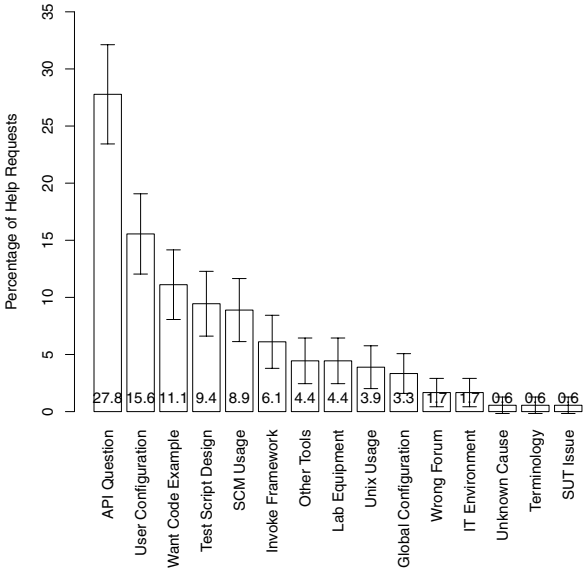

Fig. 3. Distribution of Help Requests Between Areas

TABLE I
THEMES, MAIN AREAS, AND SUB-AREAS

Theme	Area	Sub-Area	Problem Reports (%)	Help Requests (%)
User Behavior	User Configuration		21.7	15.6
	User Errors, Tool Usage	Invoke Framework	1.4	6.1
		Lab Equipment	3.6	4.4
		Other Tools	0.4	4.4
		SCM Usage	6.0	8.9
		UNIX/Linux		3.9
		Total	11.4	27.7
	Test Script Development	Want Code Example		11.1
		Test Script Design		9.4
		API Question		27.8
		Test Script Error	5.7	
		Total	5.7	48.3
	Total, Usage		38.8	91.5
Test Framework		Global Configuration	9.6	3.3
		Documentation	0.7	
		Implementation Error	11.4	
		Total	21.7	3.3
IT Environment			29.9	1.7
Unknown Cause / Strange Message			7.8	0.6
Wrong Forum			1.1	1.7
System Under Test			0.7	0.6
Terminology				0.6

V. DISCUSSION

In this section we discuss the results, based on the information in the messages set in relation to our experience and previously published literature. In doing this, we provide a view of our interpretation of the results and facilitate a connection to the larger context of the test automation discipline in general [19] .

Pinheiro *et al.* [22] reports on the introduction of test environment testing in a complex setup, which resulted in drastic improvements in the trust of the environment and speed of testing. In their report, they list an initial test scope for the environment testing that include testing folder permissions, database permissions, network resource availability, and configuration files. These are all issues that are present in some form in the material analyzed in this study. Drawing from this, we propose that once a test system grows to a certain size, it will become hard or impossible for the individual developer to manage the complexity. As a consequence, issues of this type, that are outside the domain of the system under test and outside the expertise of most developers, will occur.

There are three large groups of impediments in the studied material: impediments related to the IT environment, impediments related to the use of the test system, and impediments related to implementation mistakes in the test framework software itself.

The implementation mistakes can be categorized as test escapes from the testing of the test framework, and the intervention needed to prevent those is the same as with other software, increased or refocused quality control. One user wrote *"the main problem with the framework is that it is hard to use"*, which is an indication that the quality of the framework as such is perceived as sufficient.

While the number of issues connected to defects in the test framework as such were low in this study, we believe that a large number of user configuration errors, in particular those connected to change control, using the correct version Java and components in the framework, are traceable to a "pattern zero" behavior, that is, that one historically have not realized that that one is building and using a complex system when automating [1]. The studied organization is at the time of writing working hard on improving the change control for the test systems, but the effects of this is not visible in the data used for this study.

The largest group of problem reports discussed in the forum

were related to "user behaviors", that is, things that are caused by mistakes by the users themselves. This group of problems include things such as configuring up the equipment in the wrong way, not having sufficient skills, making a programming mistake in a test script, or being creative and using the equipment in a way not intended. The configuration issues are discussed in detail in Section V-B below.

A. Centrally Managed IT Environment

Turning to the IT environment, we found that the majority of the IT environment problems were connected to the use of remote login servers. There were problems with authentication and access rights. Secure shell is used to connect to remote resources, and some users have encountered problems with using the public key authentication mechanism needed to be able to do password-less login. This results in various symptoms, from not being able to pull or push code to the repositories, to somewhat odd error messages during testing. We also noticed issues connected to the viewers used to connect to the servers, and the stability of the network itself.

In a smaller scale operation, most of these issues would likely not occur due to a simpler or less secure design of the infrastructure, or be solved quickly. In the studied case, the IT environment is managed by another organization, making it effectively impossible for a user to solve the majority of those issues by himself.

Considering the amount of observed IT Environment impediments, it must be noted that the IT environment have a much larger impact than the majority of the other issues we have seen. Everyone have dependencies to the IT environment in their work. If there is a disturbance, even a minor one, a lot of people will notice it, which increases the likelihood of someone posting a question on the board. This should be compared to issues local to a user, where the user may ask his or her colleague, or continue seeking for a solution in other ways than posting on the discussion board.

The reason for the large number of IT problems posted to the board instead of being sent to the help desk can likely be explained by an observation by Damm *et al.* [8]. They state that "test tools easily become the excuse for every problem" and motivate that observation by explaining that tool usage is where problems first are discovered regardless of origin.

B. Configuration of the Framework

Looking at the user behaviors in detail, we found that the largest subgroup was related to configuring and setting up the framework. This is in contrast to what the users are asking for help about, the majority of the help requests are related to test script development.

We propose that the large number of configuration mistakes can be explained by the observation that test automation in the studied organization historically was the concern of experts and power users. In general, skilled users of a test system both need and accept extensive configurability of the test system, and are also more tolerant to odd behaviors and usage quirks than a generalist software developer [32]. When test automation is used by a larger audience this has the potential of introducing impediments for the work.

Two themes related to user configuration were clearly visible: using the wrong version of a component or dependency, and wrong selection of test system and system-under-test parameters.

As stated above and seen in Section IV and Table I, most of the problem reports can be attributed to areas that are not strictly test script development, but instead problems with configuration and support mechanisms that are needed to be able to do test script development. This is in contrast to what the users are asking for help about, we observe that the majority of the help requests are related to tool usage and test script development, and not about the things we noticed have the most problems, user configuration. User configuration errors correspond to 61% of the errors in the "user behaviors" theme, but only to 17% of the help requests in the same area.

The cause for this could be that the users are aware that the API is complex, and that their task is to develop test scripts. The rest of the test system seems to be expected to "just work", making the awareness of the risk of problems much lower. Karlström *et al.* [13] made a similar observation in their study from 2005, and proposed that "test environments appear to be taken for granted by developers".

Having the correct configuration of the test system is crucial for being able to perform tests reliably [22], and, sometimes, to be able to start the tests at all. In this category, we found three major types of issues:

(a) configuring the wrong parameters to the framework, such as what type the target test system is, the network addresses to the equipment used, or what ssh keys to use,
(b) selecting the wrong version of the framework in relation to the system-under-test, and
(c) having the wrong class-path for Java, leading to unstartable tests or unexpected behavior of the framework.

When the framework was migrated from Java 6 to Java 7, several users selected the wrong version of Java, indicating that this information had been over-looked somehow.

Easy-to-use tools are essential to make broad usage of automation feasible. In his recommendations on how to manage a test project, Black [6] proposes that configuration of a test environment can be too complex for a typical tester to handle, which is in line with our observations.

C. Development and Support Tools

Issues related to configuring and using the integrated development environment Eclipse forms a relatively large part of the problems that the users have when developing test code. When working with Eclipse, the projects worked on and the associated settings are stored in a workspace [30]. The majority of these problems seem to be related to misconfiguring the workspace or changing the contained code in an unintended way. The resolution for Eclipse problems was in most cases to remove the current workspace and create a new one. Exactly what the cause was in each case is usually hard

to determine, as removing the workspace will revert any test script changes, and reset the configurations and customizations to the workspace that were done by the user

Another large group of the user behavior-related issues, approximately 6%, were about using the software configuration management system. The purpose of software configuration management is to track changes in the software. Software configuration management can in itself be complex, and in this case, an additional complexity is that the test code is contained within the same collaboration "forge" as the discussion board, while the system under test is versioned in a completely different revision control system with a different paradigm of use. This indicates that there is a value in using the same tools for the same type of work. In the studied organization, a transformation project is ongoing to move all code, both test systems and delivered products, to the same software configuration management platform, which likely will reduce the number of usage problems.

Examples of problems encountered are duplicates of the code pushed to the software configuration management repository, and confusion that one have to "pull" the changes from the repo, as opposed to the system-under-test software configuration management system where updates are received immediately by all users after check in.

VI. Recommendations to Practitioners

Looking at the whole material, it is evident from the analysis of the discussion board that the users of the framework experience a lot of the same problems over and over again. We propose that using a proactive defect prevention process, such as the one described by Mays *et al.* [17] could help in systematically eliminating recurring issues. Such processes are in place for the system under test, but not for the studied test framework.

We emphasize the importance of using configuration management for the test system and test system configurations, in order to have a clear and obvious relation between the components in the test system, the versions of the test system, and the system under test, to avoid version mismatch of any kind. The studied organization has improved this mechanism since the data was collected.

Based on the tool-related impediments, we see a need for training in tasks that seem peripheral, but in fact are the foundation of the work in a software development team. Since development of test automation is software development [1], these skills are important for success.

Our final recommendation based on the findings in this study combined with the findings of Pinheiro *et al.* [22] , is to automate the environment set-up, and to make it as self-checking as possible. A large number of the problem reports encountered in this study could likely have been avoided by an automated "smoke test" for the environment that the user can validate the setup with once it has been configured, or additional functionality in the API that validate its inputs.

VII. Conclusions and Future Work

In this paper, we report on an investigation into the distribution of help requests and problem reports posted by the users of an automated test framework, and observed that the largest groups of problem reports are related to the centralized IT environment and to the setup and configuration of the test environments. In contrast to this, the largest group of help requests were about the design of test scripts. The large number of problems related to the centralization of the IT environment should be researched further. We propose that further research into the benefits and disadvantages of tool and infrastructure centralization is performed.

Turning to the majority of the problems, setup and configuration of the framework, we see an opportunity for organizations introducing environment testing, as recommended above, to replicate the study of Pinheiro *et al.* [22], which likely will result in important empirical data.

Few of the problems we encountered in our study are unknown [1][12][13][14][15][21][22][32], and neither the research community not practitioners should be surprised by the findings. Despite this, the same mistakes are repeated, and the same problems are encountered over and over again. Based on this, we conclude that investigations into how to systematically prevent commonly occurring impediments need to be performed, and that the results from those investigations need to be packaged in a way that is attractive, available, and useful to practitioners.

Acknowledgments

Erik Sternerson developed the script that extracted the raw data from the discussion board.

The research in this study was performed within the frame of ITS-EASY, an industrial research school in Embedded Software and Systems, affiliated with the School of Innovation, Design and Engineering at Mälardalen University, Sweden.

References

[1] S. Berner, R. Weber, and R. Keller. "Observations and lessons learned from automated testing". In: *Proceedings of the 27th international conference on Software engineering*. ACM, 2005, pp. 571–579.

[2] C. Beust and H. Suleiman. *Next Generation Java Testing: TestNG and Advanced Concepts*. Addison-Wesley Professional, 2007, p. 512.

[3] T. Bhat and N. Nagappan. "Evaluating the efficacy of test-driven development". In: *International symposium on empirical software engineering - ISESE '06*. New York, New York, USA: ACM Press, 2006, p. 356.

[4] P. P. Biemer. "Total Survey Error: Design, Implementation, and Evaluation". In: *Public Opinion Quarterly* 74.5 (Feb. 2011), pp. 817–848.

[5] P. Biemer and L. Lyberg. *Introduction to survey quality*. New York: Wiley Publishing, 2003.

[6] R. Black. *Managing the testing process: practical tools and techniques for managing hardware and software testing*. Wiley Publishing, 2009.

[7] D. S. Cruzes and T. Dybå. “Recommended Steps for Thematic Synthesis in Software Engineering”. In: *2011 International Symposium on Empirical Software Engineering and Measurement*. IEEE, Sept. 2011, pp. 275–284.

[8] L. Damm, L. Lundberg, and D. Olsson. “Introducing test automation and test-driven development: An experience report”. In: *Electronic Notes in Theoretical Computer Science* 116 (2005), pp. 3–15.

[9] M. Haar. *random.org - True Random Number Service*. URL: http://www.random.org/integers/.

[10] T. Hammel and S. Roths. *STAT 414 - Estimating a Proportion for a Small, Finite Population*. 2013. URL: https://onlinecourses.science.psu.edu/stat414/node/264.

[11] D. Janzen and H. Saiedian. “Test-driven development concepts, taxonomy, and future direction”. English. In: *Computer* 38.9 (Sept. 2005), pp. 43–50.

[12] K. Karhu, T. Repo, O. Taipale, and K. Smolander. “Empirical Observations on Software Testing Automation”. English. In: *2009 International Conference on Software Testing Verification and Validation*. IEEE, Apr. 2009, pp. 201–209.

[13] D. Karlström, P. Runeson, and S. Nordén. “A minimal test practice framework for emerging software organizations”. In: *Software Testing, Verification and Reliability* 15.3 (Sept. 2005), pp. 145–166.

[14] J. Kasurinen, O. Taipale, K. Smolander, K. Jussi, T. Ossi, and S. Kari. “Software test automation in practice: empirical observations”. In: *Advances in Software Engineering* 2010 (2010).

[15] J. Kasurinen, O. Taipale, and K. Smolander. “Analysis of Problems in Testing Practices”. In: *2009 16th Asia-Pacific Software Engineering Conference*. Los Alamitos, CA, USA: IEEE, Dec. 2009, pp. 309–315.

[16] D. Martin, J. Rooksby, M. Rouncefield, and I. Sommerville. “‘Good’ Organisational Reasons for ’Bad’ Software Testing: An Ethnographic Study of Testing in a Small Software Company”. In: *29th International Conference on Software Engineering (ICSE’07)* (May 2007), pp. 602–611.

[17] R. G. Mays, C. L. Jones, G. J. Holloway, and D. P. Studinski. “Experiences with Defect Prevention”. In: *IBM Systems Journal* 29.1 (1990), pp. 4–32.

[18] R. Munroe. *“Is it worth the time?”* 2013. URL: http://xkcd.com/1205/.

[19] G. W. Noblit and D. Hare. *Meta-Ethnography: Synthesizing Qualitative Studies*. SAGE Publications, Inc, 1988.

[20] *NVivo qualitative data analysis software, version 10*. QSR International Pty Ltd., 2012.

[21] C. Persson and N. Yilmazturk. “Establishment of Automated Regression Testing at ABB: Industrial Experience Report on ’Avoiding the Pitfalls’”. English. In: *Proceedings. 19th International Conference on Automated Software Engineering, 2004.* IEEE, Sept. 2004, pp. 112–121.

[22] C. Pinheiro, V. Garousi, F. Maurer, and J. Sillito. “Introducing Automated Environment Configuration Testing in an Industrial Setting.” In: *22nd International Conference on Software Engineering and Knowledge Engineering, SEKE*. Redwood City, CA, USA, 2010, pp. 186–191.

[23] D. M. Rafi, K. R. K. Moses, K. Petersen, and M. V. Mäntylä. “Benefits and limitations of automated software testing: Systematic literature review and practitioner survey”. In: *Automation of Software Test, 7th International Workshop on*. 2012.

[24] C. Robson. *Real world research*. 3rd ed. John Wiley & Sons, 2011.

[25] B. V. Rompaey and S. Demeyer. “Estimation of Test Code Changes Using Historical Release Data”. In: *2008 15th Working Conference on Reverse Engineering* (Oct. 2008), pp. 269–278.

[26] P. Runeson and M. Höst. “Guidelines for conducting and reporting case study research in software engineering”. In: *Empirical Software Engineering* 14.2 (Dec. 2008), pp. 131–164.

[27] *Scrum Alliance - Glossary of Scrum Terms*. 2012. URL: http://www.scrumalliance.org/articles/39-glossary-of-scrum-terms.

[28] S. Stolberg. “Enabling Agile Testing through Continuous Integration”. In: *Agile Conference, 2009*. Ed. by Y. Dubinsky, T. Dybå, S. Adolph, and A. Sidky. IEEE, Aug. 2009, pp. 369–374.

[29] O. Taipale, J. Kasurinen, K. Karhu, and K. Smolander. “Trade-off between automated and manual software testing”. In: *International Journal of System Assurance Engineering and Management* (Sept. 2011), pp. 1–12.

[30] L. Vogel. *Eclipse IDE Tutorial*. 2013. URL: http://www.vogella.com/articles/Eclipse/article.html%5C#workspace.

[31] J. Wesselius. “The Bazaar inside the Cathedral: Business Models for Internal Markets”. In: *IEEE Software* 25.3 (May 2008), pp. 60–66.

[32] K. Wiklund, S. Eldh, D. Sundmark, and K. Lundqvist. “Technical Debt in Test Automation”. In: *Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on*. Montréal, Canada: IEEE, 2012, pp. 887–892.