UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Maria Raquel Lopes de Couto

On the Challenges and Solutions of Training Localization and Internationalization Testers in the Context of Manual Testing

Recife

2023

Maria Raquel Lopes de Couto

On the Challenges and Solutions of Training Localization and Internationalization Testers in the Context of Manual Testing

Trabalho apresentado ao Programa de Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Engenharia de Software e Linguagens de Programação

Orientador (a): Breno Alexandro Ferreira de Miranda

Recife

2023

Catalogação na fonte
Bibliotecária Nataly Soares Leite Moro, CRB4-1722

C871o Couto, Maria Raquel Lopes de

On the challenges and solutions of training localization and internationalization testers in the context of manual testing / Maria Raquel Lopes de Couto – 2023.

83 f.: il., fig., tab.

Orientador: Breno Alexandro Ferreira de Miranda.

Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2023.

Inclui referências.

1. Engenharia de software e Linguagens de programação. 2. Globalização. 3. Localização. 4. Internacionalização. 5. Teste de software. 6. Mapeamento sistemático I. Miranda, Breno Alexandro Ferreira de (orientador). II. Título

005.1 CDD (23. ed.) UFPE - CCEN 2024 – 013

Maria Raquel Lopes de Couto

"On the challenges and Solutions of Training Localization and Internationalization Testers in the Context of Manual Testing"

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Engenharia de Software e Linguagens de Programação.

Aprovado em: 17/11/2023

In memorian to my beloved grandmother, Francisca Lopes.

ACKNOWLEDGEMENTS

I would like to start my acknowledgements expressing my gratitude to all the women who came before me and fought, sometimes with their own lives, for women's educational rights. Without them, I would not be where I am and could not be writing this work. I join them in the continuous fight for women's rights to assure that the next generation of girls have even more opportunities.

My beloved mother, Maria de Lourdes for all the love and for always encouraging me to study. My estimated brother, Robson Couto, who is an inspiration to me since I was a child, he always know to use the words to cheer me up. The love and supporting of both was fundamental in the last two years, their words during the difficult times made me stronger.

I am immensely grateful to my advisor, Breno Miranda. There was a moment when I tried to gave up from the MSc degree and he encouraged me to continue because this work was important and worth to insist in. With his orientation and support I was able to publish my first works. Also, his jokes and friendship made this process more weightless. I am also grateful to Professor Bruno Cartaxo that guided us in the Systematic Mapping Study (SMS) performed in this work. I would not forget to thank my graduation advisor, Ialis Cavalcante, he was the first person that believed that I could be a researcher, now I also do believe.

My friends are my source of daily happiness, great memories and moments. Mauro, Celinha, Abner, João Tavares, Elisabete, Lais, Mairla, Helen, Wagner, Lívia, Evan, Luan, I can conquer the world if you guys are holding my hand. Thank you for the great moments that distracted me when everything was tough and I was sad, some of you are my chosen family, love you!

I would like to thank my colleagues from work, Arthur, Gabi, Natalia, Luana, Túlio, João, Malu and Julia for supporting me in this work. Special thanks to Gabi, for all the kindness in helping me to configure the applications used in this work, and for the volunteers from Class 25 that agreed to participate of the empirical evaluation reported in this work. Thank you all!

Finnaly, I would like to thank the CIn/Motorola project from UFPE for supporting this research.

"Quando o homem decidir reformar sua consciência, o mundo tomará outro roteiro". Carolina Maria de Jesus (Jesus 1963)

ABSTRACT

Internationalization (i18n) and localization (L10n) tests are necessary to guarantee the quality of software that supports more than one language and is targeted to be launched in the global market. In this work, we aimed to understand the state of the art and practice of i18n/L10n testing. To achieve this goal, we conducted a Systematic Mapping Study (SMS) to answer the following research questions: "What are the main challenges faced by L10n and i18n testers?", "What are the most commonly used testing strategies for L10n and i18n testing?", "What is the current tooling support for L10n and i18n testing?" and finally, "Are there differences in L10n and i18n testing techniques depending on the context in which they are applied?". The primary studies retrieved by our search string reported that the main challenges faced by L10n and i18n testers are the lack of scientific research on the topic, high manual labor, very limited automation support, the cost of fixing internationalization faults, and testing right-to-left (RTL) languages. The most commonly applied testing strategy is system level testing since this type of testing is typically performed in the final stages of development. Regarding the current tooling support for this practice, only four tools were reported in the primary studies as specialized for the context of L10n and i18n testing. Lastly, the context (mobile, web, or system) is directly linked to the testing technique applied due to the nuances and unique challenges of each context. From these results, the lack of materials and tooling support caught our attention, as it directly impacts the training of novice L10n and i18n testers. Motivated by this problem, we developed an approach that involved the creation of L10N-TRAINER, this tool simulates i18n/L10n faults in mobile applications to replicate the issues found during this type of testing. To evaluate the effectiveness of L10N-TRAINER, we conducted an empirical study in a real industrial setting. This study with L10N-TRAINER demonstrated that using our tool improved the performance of the group of participants who were first introduced to L10n and i18n testing with an application seeded by the tool. The group that was initially introduced to the practice using the traditional methodology employed by the company showed stability or a decline in their results, while the group trained first with the training application improved their discovery of true bugs, increasing from 2 to 5 true bugs.

Keywords: localization; internationalization; software testing; tool; systematic mapping study.

RESUMO

Testes de internacionalização (i18n) e localização (L10n) são necessários para garantir a qualidade de um software que suporta mais de um idioma e tem como objetivo ser lançado no mercado global. Neste trabalho, objetivou-se compreender o estado da arte e prática dos testes de i18n/L10n. Para atingir tal objetivo, conduziu-se um Mapeamento Sistemático (MS) para responder às seguintes questões de pesquisa: "Quais os principais desafios enfrentados pelos testadores de L10n and i18n testing?", "Quais as estratégias de teste mais comumente usadas?", "Qual é o suporte atual de ferramentas para L10n and i18n testing?" e finalmente, "Existem diferenças nas técnicas de L10n and i18n testing dependendo do contexto em que elas são aplicadas?". Os estudos primários relataram que os principais desafios enfrentados pelos testadores de i18n/L10n são: a falta de pesquisas científicas sobre o tema, o alto trabalho manual, o suporte de automação muito limitado, o custo de corrigir falhas de internacionalização e testar idiomas Righ-to-Left (RTL). A estratégia de teste comumente aplicada é o teste em nível de sistema, visto que esse tipo de teste é normalmente realizado nos estágios finais de desenvolvimento. Com relação ao suporte atual de ferramentas para esta prática, apenas quatro ferramentas foram relatadas nos estudos primários como específicas para o contexto de testes de i18n/L10n. Por fim, o contexto (mobile, web ou sistema) está diretamente ligado à técnica de teste aplicada devido às nuances e desafios únicos de cada contexto. O desafio da falta de materiais e ferramentas foi considerado crítico, pois impacta diretamente no treinamento de novos testadores. Motivados por este problema, desenvolveu-se uma abordagem que envolveu a criação da L10N-TRAINER, uma ferramenta que simula falhas de i18n/L10n em aplicações móveis para replicar os problemas de L10n and i18n testing. Para avaliar a eficácia da L10N-TRAINER, conduziu-se um estudo empírico em um ambiente industrial real. Esse estudo demonstrou que o uso da ferramenta melhorou o desempenho do grupo de participantes apresentados ao teste pela primeira vez com um aplicativo alterado pela ferramenta. O grupo que inicialmente iniciou a prática utilizando a metodologia tradicional empregada pela empresa apresentou estabilidade ou queda em seus resultados, enquanto o grupo que foi treinado inicialmente com o aplicativo de treinamento melhorou sua performance, aumentando de 2 para 5 bugs encontrados.

Palavras-chave: globalização; localização; internacionalização; teste de software; mapeamento sistemático.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ASCII	American Standard Code for Information Interchange
g11n	Globalization
i18n	Internationalization
IT	Information Technology
JSON	JavaScript Object Notation
L10n	Localization
LLMs	Large Language Models
LTR	Left-to-right
MT	Machine Translation
RTL	Right-to-left
SMS	Systematic Mapping Study
SUT	Software Under Test
UI	User Interface
XML	eXtensible Markup Language

CONTENTS

1 INTRODUCTION

1.1 CONTEXT

From a business perspective, developing software that is accessible worldwide is essential for enhancing a company's reputation and for expanding their reach. Additionally, globally-distributed software is vital for promoting inclusiveness as demonstrated by a survey conducted in 29 countries by Common Sense Advisory (Survey: Consumers prefer their Own Language). The survey found that 76% of the users prefer products with information in their own language and 40% will not buy from websites that are not in their language. To succeed in a global market, companies must consider adapting their products to the language, culture and requirements of each country they want to be present in. By doing so, businesses can better cater to the needs of their target audiences, and improve customer satisfaction.

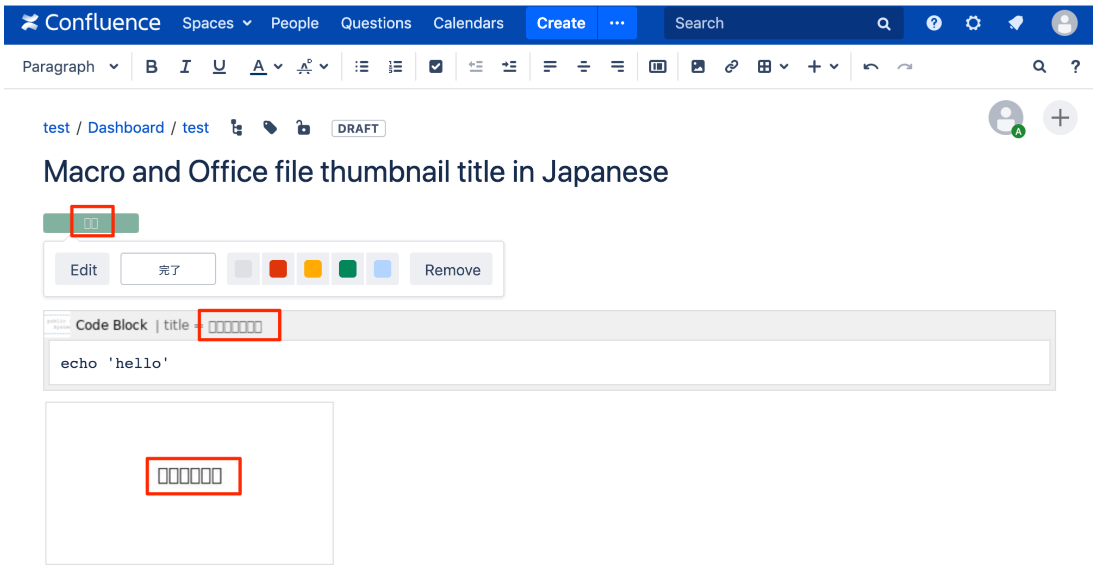
Products designed for a global market must consider three key processes: Globalization (g11n), which refers to the software development life-cycle of globally-distributed products; Internationalization (i18n), which involves designing the software such that it can handle multiple languages and cultural conventions; and Localization (L10n), which entails adapting a product to meet the linguistic and cultural needs of the specific country or region in which it will be used or sold.

Typically, software is first developed in the English language before undergoing the localization process. During localization, language-specific and culturally relevant elements are adapted to meet the needs of specific *locales*. This involves the translation of all relevant text strings, as well as adjustments to currency formats, date/calendar and time conventions, and design elements. A *locale* is as combination of a language and a region: even if Portuguese is the spoken language in both Brazil (BR) and Portugal (PT), different *locales* (pt-BR and pt-PT, respectively) are defined to accommodate the particularities of the use of the Portuguese language in each country/culture.

The importance of `i18n`/`L10n` testing is emphasized by the fact that `i18n`/`L10n` failures are easily noticeable by end-users.

During the `L10n` process the User Interface (UI) strings can increase in size and cause differences in the layout when compared with the English version. Another issue that can occur in this process is the absence of support for languages that use non-ASCII characters leading to issues such as the one displayed in Figure 1, when the software does not support the characters used for those languages and displays empty boxes (know as tofu symbol) instead.

Figure 1 – Missing content due non support of non-ascii characters.



```
echo 'hello'
```

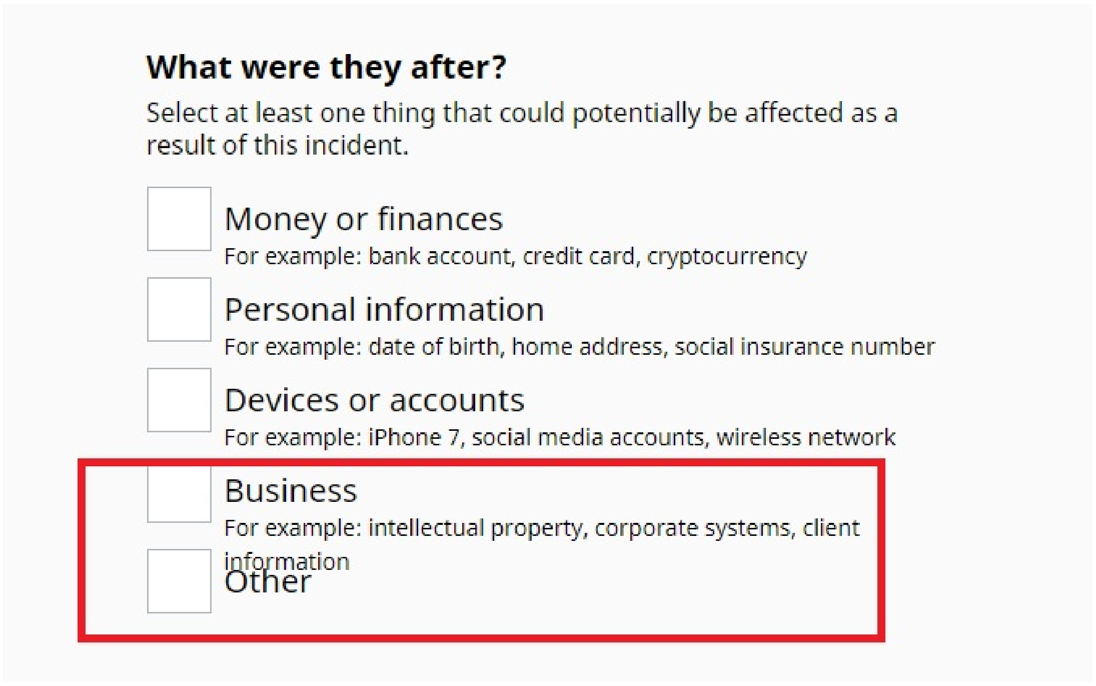
Source: (Confluence).

To identify i18n/L10n issues during the development stage avoiding them reaching the end user, some L10n and i18n testing techniques are applied (Linguae), such as **Linguist testing**, that checks the accuracy of translation within the context, usually this is made by professional linguists. Besides that, this also involves looking for terminology inconsistencies, missing content and issues related to date/calendar and time format. **Functional testing** verifies the behaviour of the application under the supported locales by the software, a common validation from this process is to check if the text fields support the input of non-ascii characters without crashes. **Cosmetic testing** in turn, aims to find issues related to character display on the UI. Images, icons and the layout are checked to verify if they are according to the locale conventions. Issues such as overlapping (Figure 2) are found with this type of testing, later in Chapter 2 this issue is further described.

Cosmetic issues, despite not impacting in the software's functionalities in general, need to be detected during the testing stage since the UI is the way that users interact with the software to achieve their needs and goals. Ignoring the validation of such issues can indicate negligence in the software development and be decisive for the success or failure of the software. For this reason it is important to address these issues during the L10n and i18n testing, and to achieve this, it is important that the testers are properly trained to detect such failures during the testing phase.

In this work we performed a study, further described in Chapter 3, in order to identify the

Figure 2 – Cosmetic issue - Overlapping.



What were they after?

Select at least one thing that could potentially be affected as a result of this incident.

Money or finances
For example: bank account, credit card, cryptocurrency

Personal information
For example: date of birth, home address, social insurance number

Devices or accounts
For example: iPhone 7, social media accounts, wireless network

Business
For example: intellectual property, corporate systems, client information

Other

Source: (Github).

main challenges faced by L10n and i18n testers. We find out that training L10n and i18n testers poses its own set of challenges. Ideally, testers should possess a deep understanding of different cultures, languages, and technical aspects related to software adaptation. Unfortunately, testers generally lack fluency in all of the languages supported by the Software Under Test (SUT) (Awwad and Slany 2016). This limitation can make it challenging for them to differentiate between a genuine failure and an expected behavior when conducting tests on the SUT in a foreign language. Another requirement for L10n and i18n testers is attention to detail, in order to look for issues using the shoes of the end user. In this sense, the training of novice L10n and i18n testers should address the most common failures found when performing this type of testing, so that the testers are prepared to identify those issues even when not having knowledge in the tested language.

In this study we aimed to contribute with the L10n and i18n testing community by developing a tool, the L10N-TRAINER, to aid L10n trainers in the challenge of training novice testers. Our tool seeds L10n and i18n faults in applications that can be used for training novice testers. By using our approach the trainer has the opportunity to train a novice tester using an app that simulates real L10n and i18n failures whilst also stimulating manual and exploratory test.

1.2 CONTRIBUTIONS

The main contributions of this work are:

1. A SMS in the field of L10n and i18n testing. By running the elaborated search string we could find 1073 papers, from this total only 18 were selected as primary studies. These papers answered the following research questions:

   - *Q1: What are the main challenges faced by* L10n *and* i18n *testing?* The works retrieved in our search pointed four main challenges: the lack of scientific research in the topic; high manual labor and very limited automation support; the cost of fixing internationalization faults; and testing right-to-left (RTL) languages. The lack of research and tooling support directs affects the training of novice testers due the limited material to rely on.
   - *Q2: What are the testing strategies most used for* L10n *and* i18n *testing?* The vast majority of the papers focused on system testing.
   - *Q3: What is the current tooling support for* L10n *and* i18n *testing?* We identified four tools as specialized tools for the context of L10n and i18n testing.
   - *Q4: Are there differences between* L10n *and* i18n *testing techniques depending on the context in which they are applied? Given the nuances and unique challenges of each context there is a need for specialized tools and techniques that are tailored to each context.*

2. The development of L10N-TRAINER, a tool to assist in the training of novice L10n and i18n testers. The tool seeds L10n/i18n faults in Android or React-Native applications in order to simulate the failures found during L10n and i18n testing.

3. An empirical evaluation with L10N-TRAINER in a real industrial setting. We performed an empirical study with L10N-TRAINER to evaluate the its effectiveness in the training of novice testers in a real industrial setting. The evaluation counted with 9 participants that were divided in two groups: group A started to learn about L10n and i18n testing with an application seeded by L10N-TRAINER and then was trained following the traditional training approach adopted by the industrial setting. Group B did the opposite. By following our proposed approach, group A improved their performance in finding bugs, from 2 bugs in the training application to 5 in the real industrial

software. On the other hand, group B showed stability in the results, 3 bugs in the real industrial software and 3 in the training application. The testers that were exposed to `L10n` and `i18n` testing failures in their training using a more simpler application could recognize the failures in the production software more easily.

1.3 DISSERTATION ORGANIZATION

This section describes how the subsequent chapters of this work are structured:

- **Chapter 2:** Presents the main concepts related to this dissertation. The process of g11n, i18n and L10n are further described and the most common failures found during `L10n` and `i18n` testing are presented, we also give more details about how a globalized software is tested.
- **Chapter 3:** In this chapter we relate the steps and main findings of the SMS performed in this work. We share with the testing community the main challenges faced by the practitioners of `L10n` and `i18n` testing, as well as the tools and techniques used for this practice according to the literature.
- **Chapter 4:** The L10N-TRAINER concept, modules and functionalities are presented. We also describe a walkthrough to familiarize the reader with the tool user interface and to describe how it is used. The contents of this chapter were partially published in the Brazilian Symposium on Software Engineering (SBES'23) (Couto and Miranda 2023).
- **Chapter 5:** Presents the empirical evaluation performed with L10N-TRAINER in a real industrial setting. We share the empirical study design, pilot results, description of participants, execution steps, and a discussion of the observed results. The contents of this chapter were partially published in the Brazilian Symposium on Systematic and Automated Software Testing (SAST'23) (Couto and Miranda 2023).
- **Chapter 6:** Concludes our work and discusses our approach limitations, contributions and future work

2 BACKGROUND

In this chapter we describe the main concepts and definitions related to this work. We introduce the concepts of globalization (*g11n*), internationalization (*i18n*) and localization (*L10n*). After introducing these concepts we present a brief description about how a localized software is tested. Finally, the most common types of failures revealed by internationalization and localization testing are presented.

2.1 GLOBALIZATION, INTERNATIONALIZATION, AND LOCALIZATION

Globalization is a term used in Geography to describe how trade and technology have made the world into a more connected and interdependent place. Furthermore, it captures in its scope the economic and social changes that have come about as a result (Geographic). With similar purpose, this term is also used in software development to describe the process of launching a product into the global market.

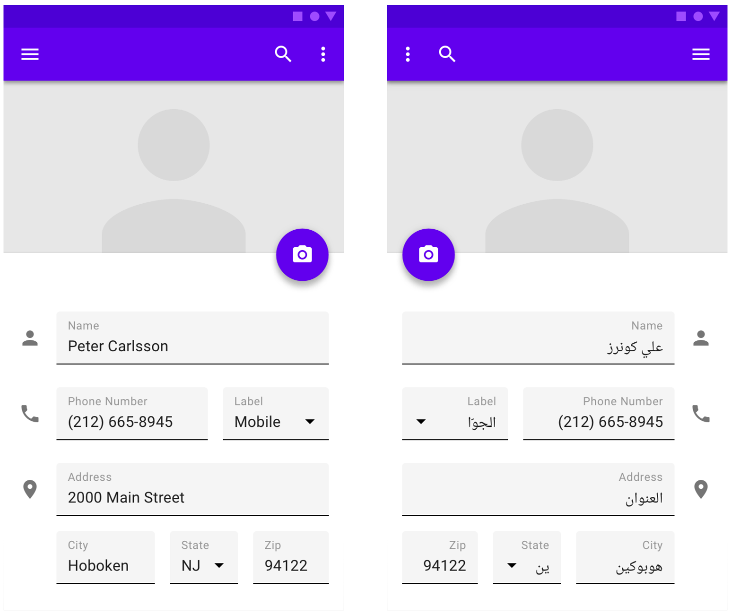
Globalization entails a comprehensive and well-structured product development life cycle that allows the designing of software that can be distributed and used across multiple countries and cultures. This includes identifying and implementing features that allow the customization of language and other cultural conventions (LISA 2003). In order to deliver a software to a global audience, g11n involves two more concepts: Internationalization and localization.

In this sense, ***internationalization*** is related to the code design and it is a crucial step for globalization and localization processes. It is responsible for preparing the software to be adapted to different languages, regions and cultures without the need of a code version of the software for each locale supported. By doing the internationalization of the software, it should be possible to display appropriate date format, characters, numbers, and measure units for a specific region (Ynion 2020).

To prepare a software to be later localized there are a few good practices recommended, such as: externalizing the resources, making the layout more flexible for adjusting the UI for longer translations, changing the code to adhere to different locale formats (names, dates, units and so on) and having bidirectional support for RTL locales (locales where the reading is done from right to left). Figure 3 shows an example of UI adapted to RTL.

The code in Listing 2.1 displays an example of the `i18n` process in practice. Instead of

Figure 3 – LTR and RTL versions of contacts app UI.



(a) LTR UI version. (b) RTL UI version.

Source: (Design).

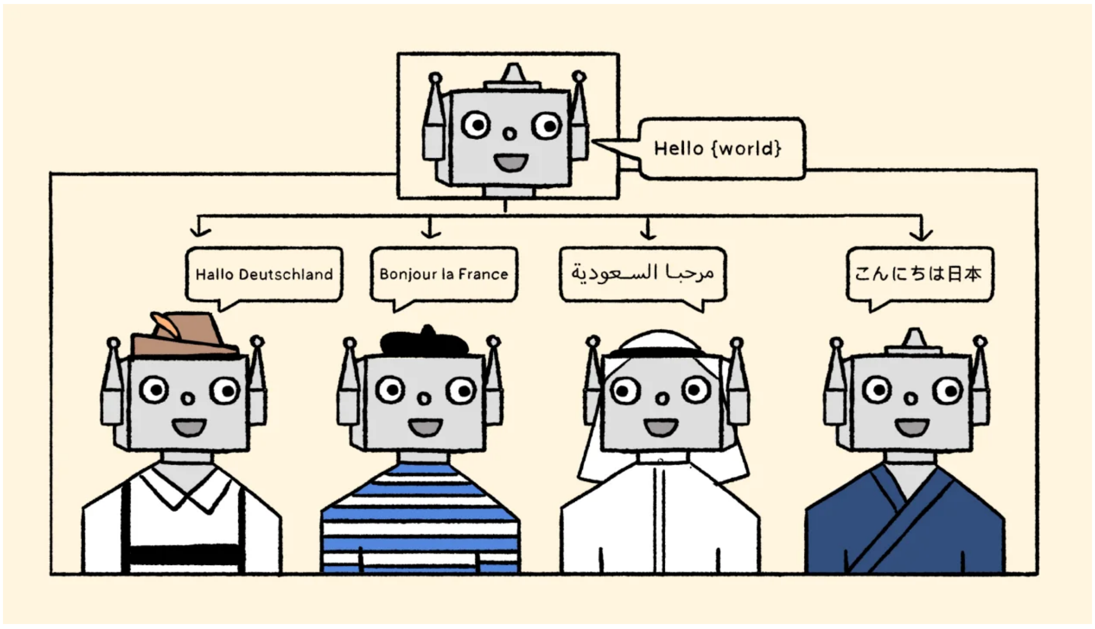
using hard-coded strings for “Good Morning”, the key `hello_world` is used to automatically retrieve the string value for each target language without requiring fundamental engineering changes. When executing the code, the application loads the data related to the chosen application/system idiom. This process is illustrated in Figure 4.

```
/*
Localizable.strings (Portuguese)
*/
"hello_world" = "Bom dia!";

/*
Localizable.strings (English)
*/
"hello_world" = "Good Morning!";

let greetingsId = "hello_world"
let goodMorning = NSLocalizedString(greetingsId, comment: "")

print(goodMorning)
```

Listing 2.1 – Internationalized code example

Although Internationalization seems to be a labour-intensive task, typically is a one-time cost. So the more languages and cultures a product is localized to, the greater the return on investment for internationalization (Krukowski).

Figure 4 – Example of i18n process.

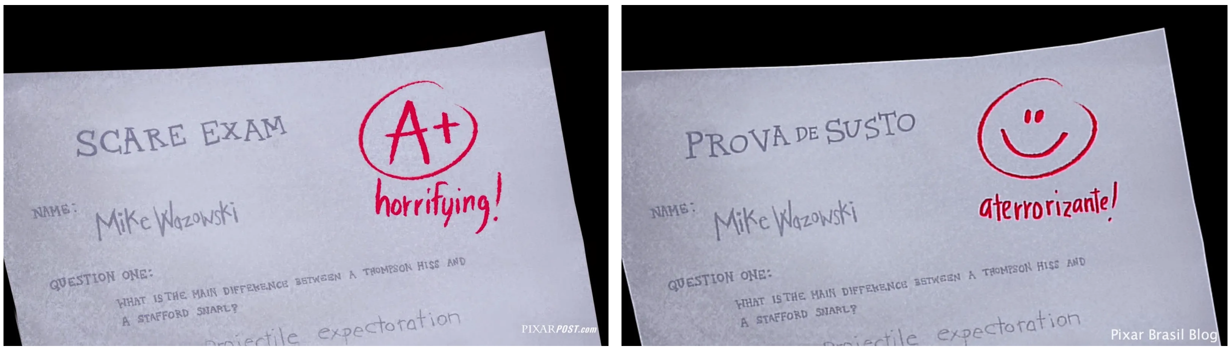


Source: (Krukowski).

Localization is the process of modifying the software to account for differences in distinct markets. While translation is a crucial aspect of localization, it is just one part of a broader process that encompasses various modifications to the software. This can include making changes to the software to accommodate local cultural conventions, such as adapting images or colors to be more appropriate for a specific culture. For example, sometimes the name of a software or feature needs to be changed to adapt it to some regions where the terminologies used might be considered offensive in that culture.

L10n also takes in consideration the context whilst translating the strings. Slang or popular saying needs to be adapted to the culture in order keep their meaning. A good example of this, that is beyond software development, is the localization of movies and video games. Frequently, movies scenes need to be modified into the country's local language or the scene won't hold the same emotional resonance or laugh factor. For example, Figure 5 displays a scene of the "Monsters University" movie, in the original setting Figure 5a and the localized Brazilian Portuguese version Figure 5b. The grade "A+" used in the original scene was replaced by a smile face in the variant in order to make more sense for that country's audience. This example is also a good model of a simple way to apply localization. Instead of using the grades patterns adopted by each country, Pixar used the simple strategy of replacing the grade by the smiling face drawing. This example shows in practice why localization is important not only for software, but for globalized products in general, because it is directly linked to the user experience.

Figure 5 – Monsters university movie localization example.



(a) English version scene. (b) Brazilian Portuguese version scene.

Source: (Pixar).

2.2 TESTING LOCALIZED SOFTWARE

To ensure software quality and reinforce that software meets the conventions and requirements of a specific culture, it is essential to conduct i18n and L10n testing. This is particularly important because i18n and L10n failures are easily noticeable by end-users. These failures may lead to a poor user experience and ultimately damage a company's reputation.

According to Ramler and Hoschek (Ramler and Hoschek 2017) L10n testing is rarely investigated in scientific work and there are only a few reports on automated approaches for L10n testing providing very little empirical results or practical advice. Awwad and Slany (Awwad and Slany 2016) relate that some companies and developers do not test their applications from the perspective of i18n/L10n at all, either automatically or manually. The lack of material and information about this testing strategy turns into a limiting factor for the spread of the practice.

Performing i18n and L10n testing can be challenging and time-consuming due to the number of tests required, which is directly proportional to the number of supported locales. Following the localization process, multiple variants of the software are typically created and tested, with one variant for each supported locale. For example, the scope of the `L10n` and `i18n` testing is directly related to the number of languages supported by the SUT. For instance, if the SUT supports 20 languages and there are 50 test cases in the `L10n`/`i18n` test suite, the tester has to perform 1000 test cases (20 languages times 50 test cases). For this reason, i18n and L10n testing are strong candidates for test automation. Another major challenge is the fact that general testers do not have fluency in all of the target languages, so it is difficult for them to discern between true positive and false positive bugs.

Usually, `L10n` and `i18n` testing is performed at the early stage of development since most of the failures of this type of testing are only visible on the UI. Another fact that delays the `L10n` and `i18n` testing is the final definition of the strings used in the software. When an application is under development, some strings from the source might be altered, the ideal is leaving the `L10n` to end steps in order to avoid costs re-translating the modified strings (Microsoft). The `L10n` process is considered expensive, because to assure the quality of the localized text linguistics professionals are needed for each supported locale of the application. Besides, in some cases the fixing of `L10n` issues is performed by those professionals and this labour is expensive, with hourly rates for these professionals reaching $77 per hour (Tarasiewicz). This cost impacts in the final price of performing `L10n` and `i18n` testing, which might affect the dissemination of this practice.

As previously stated, `L10n` and `i18n` testing industrial practice are rarely addressed in the literature which makes the testing strategies for this practice unclear. In gray literature the listing strategies for this type of testing involves (Microsoft):

- Comparing the localized version of the UI with the source (English) in order to find discrepancies on the UI elements.
- Verifying inputting, storing, processing, and outputting multilingual text with no data loss.
- Checking the user interface elements to verify if they are respected, including mirroring and text rendering.
- Verifying the proper handling of language settings such as date, time, number and currency formatting, calendar and sorting settings.

2.3 LOCALIZATION AND INTERNATIONALIZATION FAILURES

In this section the most common failures found during `L10n` and `i18n` testing are described. The issues are very noticeable for the software's end user since they are highly visible on the UI. Sometimes, these failures are classified as *cosmetic*, mostly because they do not influence the functionalities of the software. However, they indicate negligence during the software development. For better comparison and understanding, in some of our examples representing the failures, we presented the source version of the UI (English language) and the localized version with the failure side by side.

Frequently, it is difficult, even for professionals, to classify the failures into L10n failures or i18n failures, mostly because the classifier factor will only be discovered when the root cause of the issue is further investigated. For example, a missing translation can be an outcome of a hard-coded string in the software, and this is a i18n fault. However, if the string is not hard-coded and the issue happens because the string was missing or delayed during the translation process, it is a failure. The tester that performs black box testing, without having access to the code of the application is not able to classify the issue into the categories when reporting it. For this reason we did not classify the following issues into i18n or L10n issues. This classification is also not addressed frequently in the literature; Š Packevičius and colleagues (40) describe a series of i18n and L10n failures, however, they depict them solely based on a list of UI defects, without adhering to any documentation related to i18n and L10n testing or classifying them.

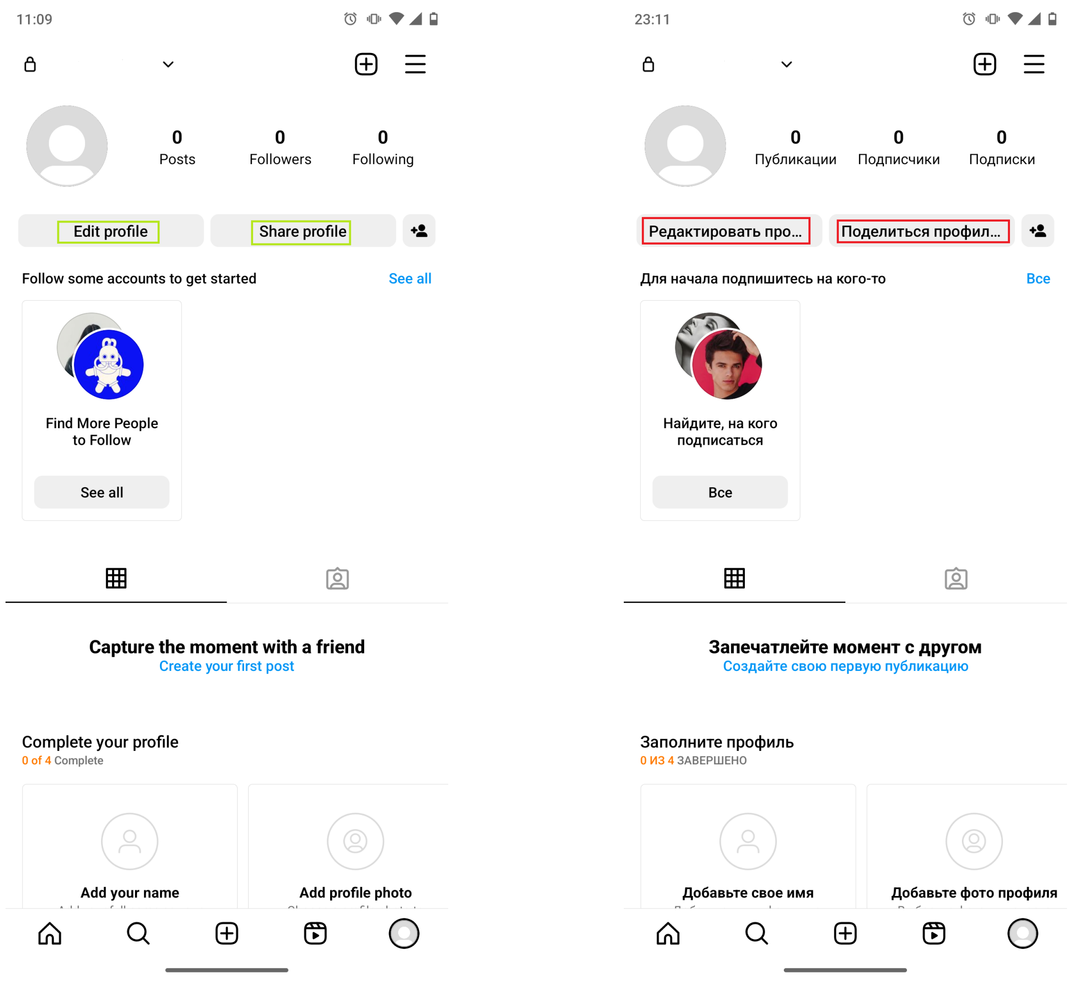
2.3.1 Ellipsis

Ellipsis are very common in a software that has been through the process of i18n and L10n. In fact, during the translation process, the size of a string can expand from 100% to 200% in another language (IBM). This increase in the string's size is often the root cause of ellipsis, that happens when a string does not fit in the layout and the three dots sign is used as a way to indicate that the string continues, see the visual representation in Figure 6.

Although ellipsis are used as a system resource for larger translations, there are scenarios that need to be handled carefully. The first one is when the string is truncated by the ellipsis in the middle of a word resulting in a swearword, this can be negative for the software and publicity damaging for the developer.

The other scenario is when the string is cropped by the ellipsis leaving the end user without enough instructions to perform an action. This affects directly the usability of the software and can be a decisive fact for a software success.

Figure 6 – Ellipsis issue.



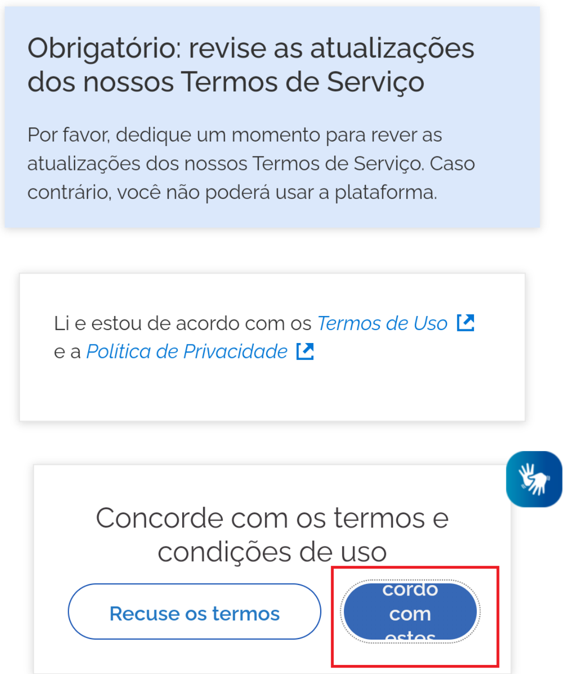
(a) English version of Instagram profile screen. (b) Russian version of Instagram profile screen.

Source: Created by the author.

2.3.2 Truncation

Truncation failure is a bit similar to ellipsis, however, in this issue the string is partially cropped by the layout without using ellipsis sign. Figure 7 displays an example of truncation issue, the full Portuguese text for the button should be "Concordo com estes termos", however, the string is not fully displayed in the button. The root cause of this issue often happens when the layout is not prepared to support longer translations. The identification of this failure is tricky because the text might seem to be completely shown, however a piece of the string is hidden by the layout. There are some cases that the identification of this issue is only noticed by L10n and i18n testing strategies, such as visually checking how longer translations are displayed on the UI.

Figure 7 – Example of truncation issue.



Obrigatório: revise as atualizações dos nossos Termos de Serviço

Por favor, dedique um momento para rever as atualizações dos nossos Termos de Serviço. Caso contrário, você não poderá usar a plataforma.

Li e estou de acordo com os *Termos de Uso* e a *Política de Privacidade*

Concorde com os termos e condições de uso

Source: Created by the author.

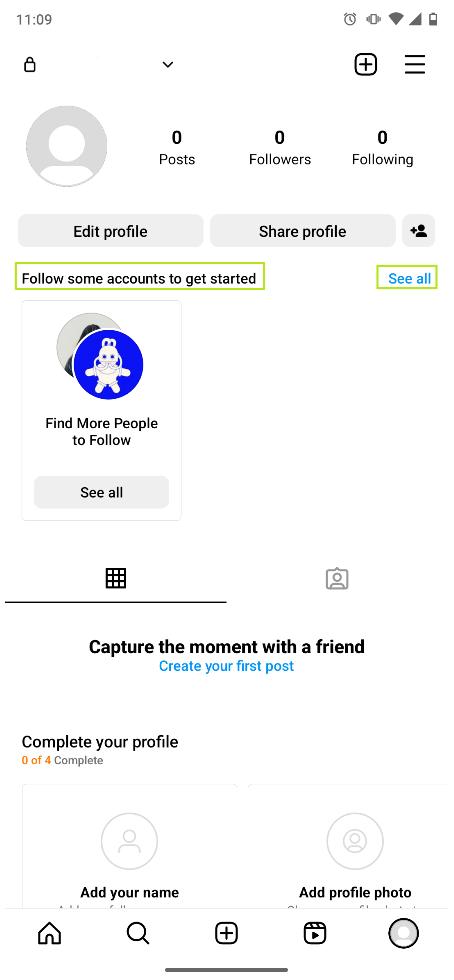
2.3.3 Overlapping

In the same line of root cause of truncation and ellipsis, the overlapping issue is also derived from the increase of the strings during translations. This issue happens when the UI elements try to support longer translations and overlap other UI elements, an example of this issue is present in Figure 8. Usually, to solve this issue the string needs to be shortened without modifying its meaning, when this is not possible, it is needed to re-adapt the application layout.

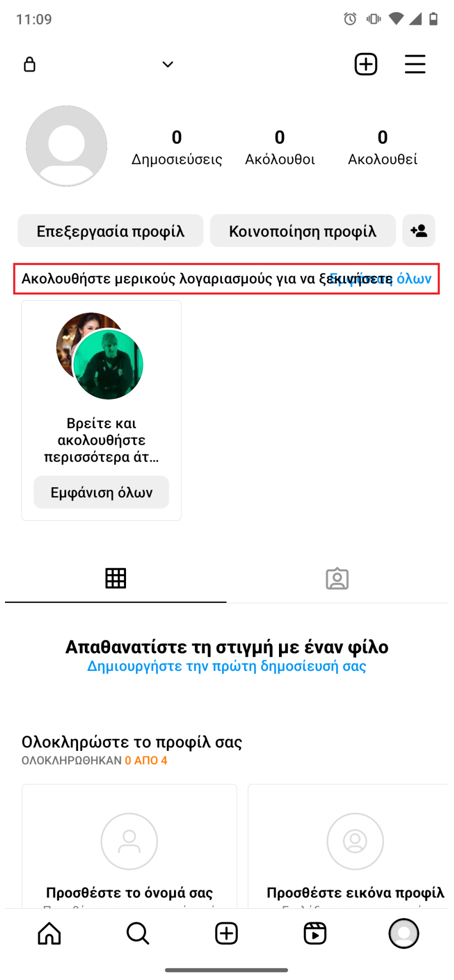
2.3.4 Missing translation

Missing translation occurs when some strings are not translated from the source language to the target language and the UI presents content in both source and target language. See an example of this failure in Figure 9, note that some strings from the screens are in English language and other are localized in Brazilian Portuguese. It is important to highlight that this is not applicable to strings that are propositional left in the source language, such as nouns, brand, feature's name, and technologies (e.g Bluetooth) names. An example of the last one is presented in the third image from Figure 9, the string "Proxy" not being translated is not

Figure 8 – Example of overlapping issue.



(a) English version of Instagram profile screen.


11:09
0 Δημοσιεύσεις
0 Ακόλουθοι
0 Ακολουθεί
Επεξεργασία προφίλ
Κοινοποίηση προφίλ
Ακολουθήστε μερικούς λογαριασμούς για να ξεκινήσετε
Εμφάνιση όλων
Βρείτε και ακολουθήστε περισσότερα άτ...
Εμφάνιση όλων
Απαθανατίστε τη στιγμή με έναν φίλο
Δημιουργήστε την πρώτη δημοσίευσή σας
Ολοκληρώστε το προφίλ σας
ΟΛΟΚΛΗΡΩΘΗΚΑΝ 0 ΑΠΟ 4
Προσθέστε το όνομά σας
Προσθέστε εικόνα προφίλ

(b) Greek version of Instagram profile screen.

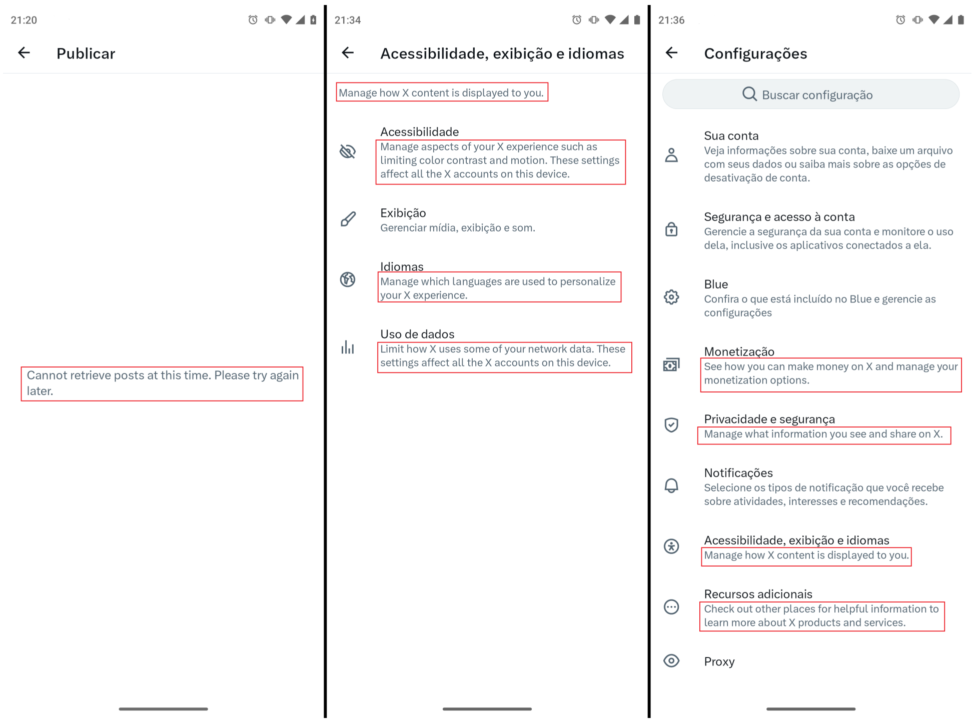
Source: Created by the author.

considered an issue since it is a technology term without translation.

2.3.5 Data format issues

Data format issues are related to units (such as temperature, memory size, frequency, time) and date/calendar format that does not follow the locale expected pattern. For example, in Figure 10, the application is set to pt-BR and the expected pattern for date in Brazil is (day-month-year), however, the date is following the North American pattern that is (month-day-year).

Figure 9 – Missing translation issue for an application configured for Brazilian Portuguese.


21:20

Publicar

Cannot retrieve posts at this time. Please try again later.

21:34

Acessibilidade, exibição e idiomas

Manage how X content is displayed to you.

Acessibilidade
Manage aspects of your X experience such as limiting color contrast and motion. These settings affect all the X accounts on this device.

Exibição
Gerenciar mídia, exibição e som.

Idiomas
Manage which languages are used to personalize your X experience.

Uso de dados
Limit how X uses some of your network data. These settings affect all the X accounts on this device.

21:36

Configurações

Buscar configuração

Sua conta
Veja informações sobre sua conta, baixe um arquivo com seus dados ou saiba mais sobre as opções de desativação de conta.

Segurança e acesso à conta
Gerencie a segurança da sua conta e monitore o uso dela, inclusive os aplicativos conectados a ela.

Blue
Confira o que está incluído no Blue e gerencie as configurações

Monetização
See how you can make money on X and manage your monetization options.

Privacidade e segurança
Manage what information you see and share on X.

Notificações
Selecione os tipos de notificação que você recebe sobre atividades, interesses e recomendações.

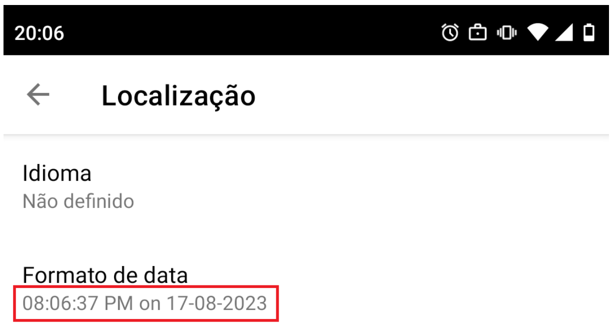
Acessibilidade, exibição e idiomas
Manage how X content is displayed to you.

Recursos adicionais
Check out other places for helpful information to learn more about X products and services.

Proxy

Source: Created by the author.

Figure 10 – String following LTR pattern instead of RTL.


20:06

Localização

Idioma
Não definido

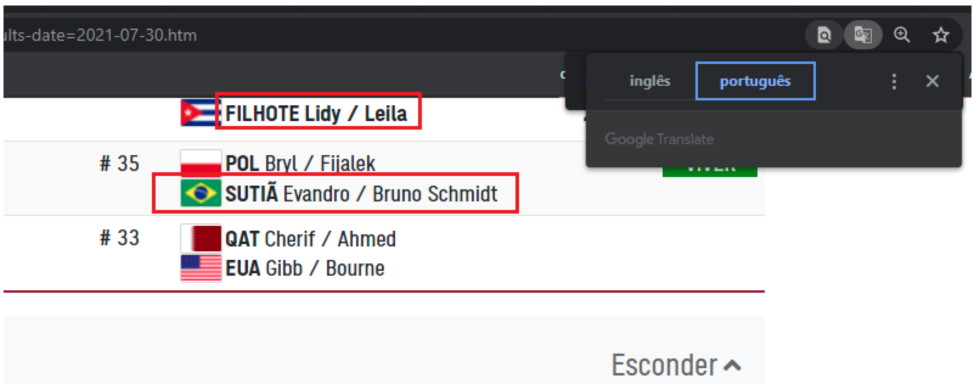
Formato de data
08:06:37 PM on 17-08-2023

Source: Created by the author.

2.3.6 Out of context translation

The process of localizing a software can be very expensive since translating the strings while taking into consideration the context and culture of the target regions requires highly specialized professionals, such as linguists. Due this cost associated, some developers choose to do the translations of the strings using Machine Translation (MT) techniques or translation engines, as result from this operation the translations might get out of the context. A word in English can have two or more translations in other languages depending of the context. For instance, the word "fits" can be applied in the context of something that can be embedded inside another thing, and it can also be used to say that something does not match with another thing. If we merely translate a sentence using this word without taking into account its meaning in the context, the result could be something completely different from the expected. To illustrate, Figure 11 displays two cases were the end user used google translate in a web page to translate its content into Brazilian Portuguese. The abbreviation for Brazil "(BRA)" was translated as "sutiã", that is the an undergarment worn to support the breasts. Also, the abbreviation for Cuba "(CUB)" was translated as "filhote", that is the same as pup.

Figure 11 – Out of context translation for the abbreviation "BRA".



FILHOTE Lidy / Leila
# 35 POL Bryl / Fijalek
SUTIÃ Evandro / Bruno Schmidt
# 33 QAT Cherif / Ahmed
EUA Gibb / Bourne

Source: Created by the author.

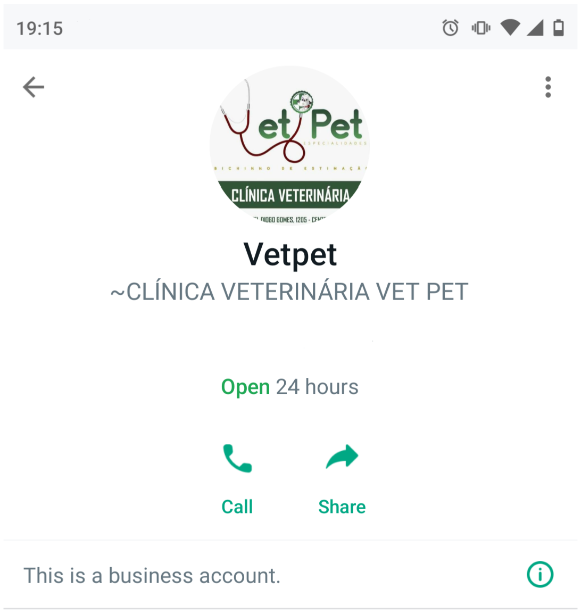
2.3.7 Grammar related issues

Probably this one of the most difficult issue for the `L10n` and `i18n` testers to identify, unless the practitioner is native or specialized in the tested target language. Usually, the identification of such problems are only caught by linguists or end-users since it is almost impossible for the testers to have knowledge in all languages supported by the software.

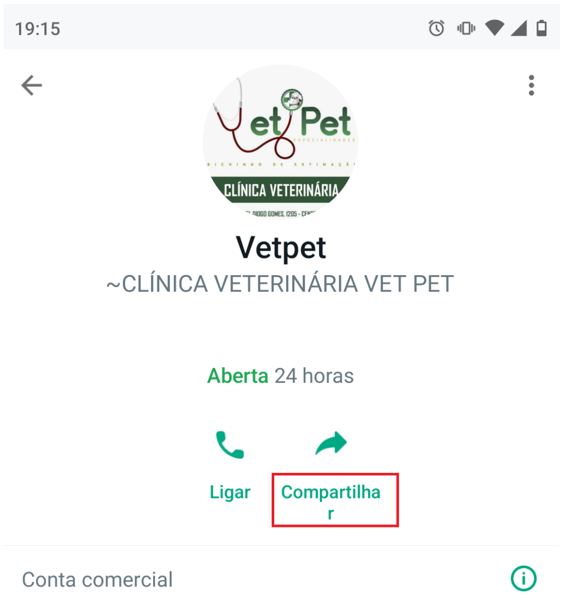
2.3.7.1 Wrong syllabification

The wrong syllabification happens when wrapping words do not fit on the UI elements. It is trickier to the tester to determine if the word is correctly syllabicated when having no or basic knowledge in the tested idiom. In Figure 12 the word "compartilhar" is broken into multiple lines and the syllabification is wrong, it should be com-par-ti-lhar.

Figure 12 – Example of wrong syllabification issue.



(a) English version of Whatsapp contact screen.



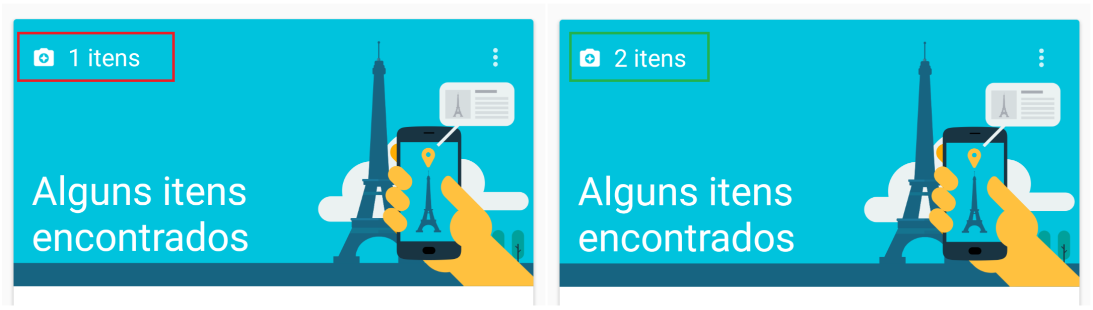
(b) Brazilian Portuguese version of Whatsapp contact screen.

Source: Created by the author.

2.3.7.2 *Grammatical gender and plural/singular*

Most of the nouns in the English language do not have a gender, when localizing a software to another idiom using this language as source, grammatical gender issues can result from this process. Beyond this, another issue that might occur is the use of plural for singular objects, or the inverse. For example, in Figure 13 counting of the itens is wrong, "1 **itens**", when it should be "1 **iten**" for singular.

Figure 13 – Wrong pluralization for item in pt-br.



Source: Created by the author.

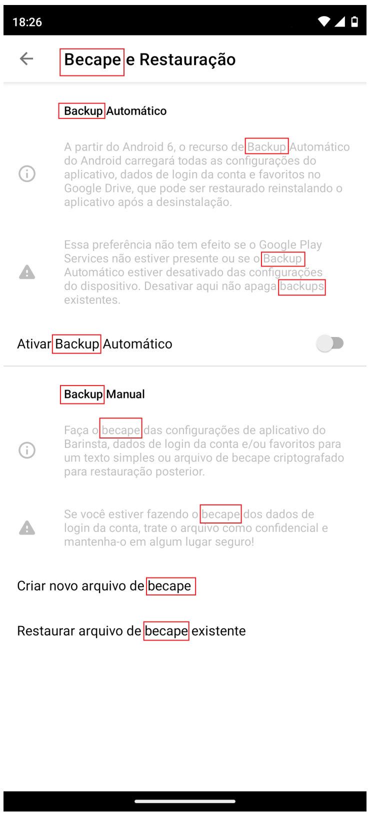
2.3.7.3 Punctuation rules

There are specific punctuation rules associated with languages that might be missing after the localization. For example, for Spanish the use of question and exclamation marks is used in the beginning and end of a sentence.

2.3.8 Inconsistency of terms

The inconsistency of terms happens when the strings are translated and a discrepancy of terms used for a same word occurs. Although the meaning of the words is the same, it can be confusing for the user or provide inconsistency in the instructions that use these terminologies. For example, in Figure 14 the terms "becape" and "backup" are used for the English word 'backup'.

Figure 14 – Term inconsistency for the word "Backup".



18:26

Becape e Restauração

Backup Automático

A partir do Android 6, o recurso de Backup Automático do Android carregará todas as configurações do aplicativo, dados de login da conta e favoritos no Google Drive, que pode ser restaurado reinstalando o aplicativo após a desinstalação.

Essa preferência não tem efeito se o Google Play Services não estiver presente ou se o Backup Automático estiver desativado das configurações do dispositivo. Desativar aqui não apaga backups existentes.

Ativar Backup Automático

Backup Manual

Faça o becape das configurações de aplicativo do Barinsta, dados de login da conta e/ou favoritos para um texto simples ou arquivo de becape criptografado para restauração posterior.

Se você estiver fazendo o becape dos dados de login da conta, trate o arquivo como confidencial e mantenha-o em algum lugar seguro!

Criar novo arquivo de becape

Restaurar arquivo de becape existente

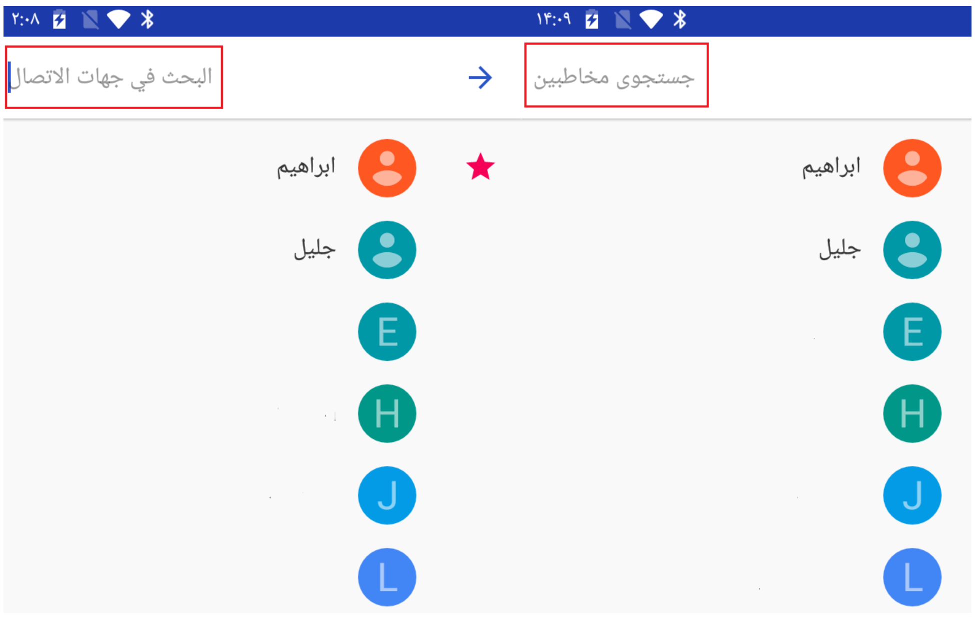
Source: Created by the author.

Besides this example, the inconsistency also can occur in terms that are sometimes written with a capitalized first letter and in other occasions written with minuscule letters.

2.3.9 RTL specific faults

There are some regions in the world where reading is made from the right to the left, this cultural aspect should also be taken in account into the i18n and L10n of the software developed for those regions. The RTL related issues can be related to UI not being mirrored, strings following the LTR pattern and scrolling actions being done in the wrong direction. In the example of Figure 15 the strings highlighted in red are following the LTR pattern, they should be right aligned such as the contacts icons.

Figure 15 – String following LTR pattern instead of RTL.



Source: Created by the author.

2.4 CONCLUDING REMARKS

In this chapter we presented the concepts of globalization, localization and internationalization. The process and some challenges of L10n and i18n testing were briefly discussed and some of the most common failures found during this type of testing were introduced and illustrated.

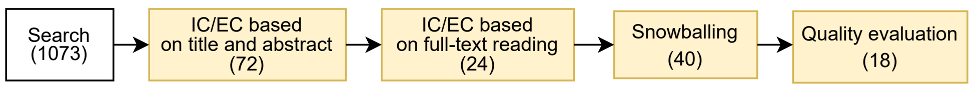
3 RELATED WORK

To comprehend the state-of-the-art and practice of `L10n` and `i18n` testing, we conducted a SMS. This chapter delineates the protocol, results, and findings derived from this study. We searched primary studies on i18n and L10n testing by querying the Scopus database, and by performing a backward snowballing cycle. We started with a set of 1073 papers and, after applying our inclusion/exclusion and quality evaluation criteria, we selected a final set of 18 primary studies. In the following we provide both a quantitative analysis, and a qualitative evaluation of i18n and L10n testing, from the perspective of the challenges, testing strategies, tooling support, and contextual differences reported by the selected studies.

3.1 SYSTEMATIC MAPPING PROTOCOL

In this section we outline the methodology employed for conducting our study, which involved a Systematic Mapping Study. Our approach largely adhered to the guidelines proposed by Kitchenham and Charter (Kitchenham and Charters 2007) and Petersen et al. (Petersen et al. 2008). Figure 16 shows the number of papers that were considered throughout each step of this mapping study. For comprehensive details on the study protocol and all the materials employed in this study (including studies selection, quality assessment, data extraction, and data synthesis), we have compiled a replication package that is accessible for reference [1].

Figure 16 – The selection procedure.


Search (1073)
IC/EC based on title and abstract (72)
IC/EC based on full-text reading (24)
Snowballing (40)
Quality evaluation (18)

Source: Created by the author.

3.1.1 Research questions

The primary aim of this Systematic Mapping Study is to examine the current state-of-the-art in internationalization and localization testing. With this objective in mind, we have formulated the following research questions to guide our investigation:

[1] <https://github.com/RaquelCouto/Msc-L10N-Trainer>

- RQ1: What are the main challenges faced by i18n and L10n testing?
- RQ2: What are the testing strategies most used for i18n and L10n testing?
- RQ3: What is the current tooling support for i18n and L10n testing?
- RQ4: Are there differences between i18n and L10n testing techniques depending on the context in which they are applied (e.g., mobile, web, desktop applications)?

3.1.2 Search strategy

By following the recommendations of Mourão et al. (Mourão et al. 2020), we adopt a hybrid search strategy that combines the use of Scopus with Parallel Snowballing. Their study found that this approach strikes the appropriate balance between precision and recall in the search process.

In this section, our focus is specifically on the methodological aspects of the search strategy related to the Scopus search engine. The specific methodology employed for the snowballing process will be discussed in Section 3.1.4.

After careful discussion, we decided that the search string to search for studies in Scopus should have the following main terms: localization/internationalization and software test. Then, we added synonyms for each term (e.g. , multilingual, globalization) and wildcards like *?* that replaces a single character anywhere in a word (e.g., locali?ation matches both localization and localisation), as well as * that replaces multiple characters anywhere in a word (e.g. test* matches test, testing, tester, among others). Also, we limited the search to the *Computing* subject area only, and explicitly excluded results related to *fault localization*, *bug localization*, and *indoor localization* since we observed that there were an intractable amount of false positives on these topics, that are clearly out of scope of this research.

Following is the search string we defined to search in Scopus, which returned 1073 papers in March, 24th of 2022. An spreadsheet with all those papers is available online [1].

```
TITLE-ABS-KEY ( ( locali?ation OR internationali?ation OR globali?ation OR multilingual OR
L10n OR i18n OR g11n ) AND ( software AND test* ) AND NOT ( “fault locali?ation” OR
“bug locali?ation” OR “indoor locali?ation” ) ) AND ( LIMIT-TO ( SUBJAREA , “COMP” )
```

3.1.3 Selection procedure

The search in Scopus yielded a total of 1073 papers, which were then utilized as input for the selection procedure. The selection procedure, following the recommendation of Kitchenham and Charter (Kitchenham and Charters 2007), consisted of two steps conducted in pairs to minimize subjectivity. To assess the inter-rater reliability of each step, Cohen's Kappa Test (Viera and Garrett 2005) was employed.

In the first step, we analyzed the papers' titles and abstracts only, excluding those that were clearly out of the scope of this research. When in doubt, we kept the papers for a more detailed analysis in the second step. The Kappa statistics for this step was 0.97, which shows a *almost perfect* level of agreement according to the Kappa reference table (Viera and Garrett 2005). After completing the first step, we excluded 1001 papers, resulting in 72 remaining papers for the second step. Although we attempted to exclude papers not relevant to `L10n/i18n` testing by incorporating specific keywords into our search string, the most of papers excluded in this step were unrelated to the topic; instead, they focused on studies related to GPS applied to robotics. The spreadsheet used for this step is available online [1].

In the second step, a comprehensive analysis of the paper's complete content enabled us to exclude 48 papers, leaving us with a total of 24 papers for further consideration. The Kappa statistics for this step was 0.73, which shows a *substantial* level of agreement according to the Kappa reference table (Viera and Garrett 2005). The spreadsheet used for this step is available online[1].

The process of selecting studies for inclusion in this research followed a set of well-defined inclusion and exclusion criteria. The inclusion criteria were designed to identify scientific papers specifically related to localization and internationalization testing, written in the English language, and providing insights relevant to at least one of our research questions. Conversely, the exclusion criteria were applied to exclude proceedings, retracted papers, non-scientific papers, and any papers that did not address at least one of our research questions. By applying these criteria, we ensured that the selected studies align with the focus and objectives of our research.

3.1.4 Snowballing

Traditional snowballing technique starts with a seed set – which is a set of papers related to a topic – and recursively looks at relevant papers on the seed set references' list (a.k.a backward snowballing, or simply BSB), as well search for additional papers looking at the papers citing the seed set (a.k.a forward snowballing, or simply FSB) (8). The recursion ends when an iteration results in no new relevant paper.

However, Mourão et al.(Mourão et al. 2017) introduced the concept of Parallel Snowballing (PSB), which, as the name suggests, conducts BSB and FSB in parallel. In parallel snowballing papers obtained with BSB are not eligible for FSB, and vice-versa. Given the proven effectiveness of the PSB approach as demonstrated in previous studies (Mourão et al. 2020), we decided to adopt this approach for our study.

The initial seed set for our study consists of 24 papers selected through the selection procedure. The BSB demanded 2 iterations, and in the end added 7 papers. While the FSB demanded 3 iterations, and added 9 papers. Overall, both the BSB and FSB contributed with 16 new papers to the initial seed set, resulting in a total of 40 relevant papers for our mapping study. The spreadsheet we used to conduct the snowballing is available online[1].

3.1.5 Quality assessment

To ensure the inclusion of high-quality papers with substantial empirical evidence, Kitchenham and Charter (Kitchenham and Charters 2007) recommend that researchers develop an instrument to assess the quality of studies to be included in a secondary study.

The quality assessment instrument we created for this study is based on recommendations of Kitchenham and Charter's (Kitchenham and Charters 2007), and also on other similar instruments for secondary studies published recently (19; Amna and Poels 2022; Lewowski and Madeyski 2022). Table 1 presents the criteria we defined for the quality assessment instrument.

To evaluate each criterion, we have adopted a scale consisting of three categories: Yes (Y) assigned 1.0 point, No (N) assigned 0.0 points, and Partially (P) assigned 0.5 point. However, different criteria have different weights. The weight of each criterion is shown in Table 1. The total score of a paper is the weighted average of the criteria. We defined 0.7 as the quality threshold, which means that any paper scoring below 0.7 in the quality assessment was excluded from the study. After conducting the quality assessment, we excluded 22 papers, resulting in 18 papers that, indeed, were included in this mapping study. The spreadsheet we used to conduct the quality assessment is available online[1].

Table 1 – Quality criteria with attributed weights.

Quality Evaluation	Weight
Q1. Is the problem well defined?	1
Q2. Does the paper have clearly stated aims and objectives?	1
Q3. Does the paper provide a clear statement of findings?	2
Q4. Is there adequate discussion of related studies?	1
Q5. Does the paper follow a research method/design that supports the aims?	2
Q6. Does the paper provide a clear context (e.g., industry or laboratory setting)?	1
Q7. Is the knowledge claim validated empirically?	2
Q8. Does it follow any guideline?	1
Q9. Does the paper explicitly discuss its limitations?	1

Source: Elaborated by the author.

3.1.6 Extraction procedure

To ensure the accuracy and reliability of the extracted data from the selected studies, a two-step verification process was implemented. The first author of this paper was responsible for initially extracting the data, including bibliographical information, research questions (RQs)-related data, and complementary information. Following this, the third author of this paper conducted a comprehensive review of the extracted data, serving as a double check to ensure its accuracy and consistency. This rigorous verification process helps minimize errors and enhances the reliability of the data used in this study. The extracted items are presented in Table 2, with further details available in the replication package.

3.1.7 Synthesis procedure

Based on the the data extracted from the selected studies, the first author of this paper conducted an open coding aligned with a focused coding and constant comparison approach (Merriam and Tisdell 2015) to synthesize the findings and answer our research questions. We identified the codes and analyzed the papers, gathering the data that was directly related to

Table 2 – Data extraction form.

Bibliographical	**Information about the paper identification**
DOI	DOI of the paper
Title	The tilte of the paper
Year	The publication year
Authors	The authors list
Org./country	Institution and country of the paper authors
Research Questions	**Challenges, strategies, tooling suport, and context**
Challenges	What are the main challenges faced by i18n and l10n testing? (RQ1)
Strategies	What are the testing strategies most used for i18n and l10n testing? (RQ2)
Tooling	What is the current tooling support for i18n and l10n testing? (RQ3)
Context	Are there differences between i18n and l10n testing techniques depending on the context in which they are applied (e.g., mobile, web, desktop applications)? (RQ4)
Extra	**Future work and additional relevant information**
SW Type	What is the type of software tested under i18n and l10n strategies?
Future Work	What is planned as future work?
Extra Info	Any relevant further information.

Source: Elaborated by the author.

the codes. Again, the third author reviewed the synthesis procedure to mitigate any personal bias.

3.1.8 Discussion

In this Section, we provide answers to the posed research questions.

3.1.8.1 RQ1: Challenges

We have identified four main challenges faced by localization and internationalization testing: i) lack of scientific research conducted on the topic; ii) high manual labor and very limited automation support; iii) the cost of fixing internationalization faults; iv) and testing right-to-left (RTL) languages.

Among the surveyed primary studies, eight papers cited the lack of scientific research

Table 3 – Selected studies. Column "Type" classifies the studies into (J)ournal, (C)onference, or (W)orkshop.

ID	Year	Authors	Title	Type
PS1	2021	Schindler et al.	Towards continuous deployment of a multilingual mobile app	J
PS2	2021	Mahajan et al.	Effective automated repair of internationalization presentation failures in web applications using style similarity clustering and search-based techniques	J
PS3	2020	Santos et al.	Bug! Falha! Bachi! Fallo! Défaut!! What about internationalization testing in the software industry?	C
PS4	2020	Escobar-Velásquez et al.	An Empirical Study of i18n Collateral Changes and Bugs in GUIs of Android apps	C
PS5	2019	Ali	Behavior-driven development as an error-reduction practice for mobile application testing	J
PS6	2019	Alameer et al.	Efficiently repairing internationalization presentation failures by solving layout constraints	C
PS7	2017	Ramler and Hoscheck	Process and Tool Support for Internationalization and Localization Testing in Software Product Development	C
PS8	2017	Ramler and Hoscheck	How to Test in Sixteen Languages? Automation Support for Localization Testing	C
PS9	2016	Alameer et al.	Detecting and Localizing Internationalization Presentation Failures in Web Applications	C
PS10	2016	Varouqa and Hammo	WIT: Weka interface translator	J
PS11	2016	Martinez et al.	Automated Localisation Testing in Industry with Test*	C
PS12	2013	Xia et al.	Software internationalization and localization: An industrial experience	C
PS13	2011	Ping et al.	Integration of cultural dimensions into software localisation testing of assistive technology for deaf children	C
PS14	2009	El-Kadi and Badreddin	Grup - A globalized approach to software engineering	J
PS15	2009	Zou and Liu	Chinese localisation of Evergreen: An open source integrated library system	J
PS16	2008	Abufardeh and Magel	Software localization: the challenging aspects of Arabic to the localization process (arabization)	C
PS17	2004	Hogan et al.	key chalenges in software localization	W
PS18	1998	Carey	Creating global software: a conspectus and review	J

Source: Elaborated by the author.

conducted on the topic and, consequently, the lack of empirical results available in the literature (PS4; PS5; PS7; PS8; PS3; PS1; PS12; PS17). The shortage of research leads to a lack of understanding of best practices, challenges, and solutions for i18n and l10n testing. This lack of understanding, in its turn, can result in inefficient and ineffective i18n and l10n testing processes. Although L10n/i18n testers need highly specialized skills (PS17), it is extremely challenging for them to be proficient in all the languages supported by the software, and the lack of scientific research and empirical results in this field poses a significant problem for training activities, as professionals find it challenging to obtain necessary resources, tools, and materials (PS3).

Another challenge identified was the high amount of manual labor required and limited automation support. The primary studies (PS7; PS8) call attention to the low number of automated approaches currently available. Since most of i18n/L10n failures (e.g., overlapping, ellipsis, truncation) are only visible at the user interface level, manual validation may still be necessary in some contexts. However, given that the number of L10n test cases can greatly increase depending on the number of *locales* supported by the software, the availability of automation support would be highly desirable.

The cost of fixing internationalization faults is also a challenge raised by the primary study (PS5). As reported in (PS5), the cost of fixing a i18n fault can be as high as 30 times the cost of handling the potential fault up-front. This may be due to the fact that when translating a string from English to the target language, the length of the translated string can often be longer than the original English string, sometimes increasing in size from 100% up to 200% (IBM). When the UI is not designed to handle the increased length of the localized string, it can cause overlap with other UI elements, display ellipses, or truncate important information. A way to address this type of issue is by sending the translated string to a linguist specialized in the target language to help reduce its size. However, this can be a costly process. Although this challenge was only mentioned in one of the primary studies surveyed, it is consistent with other research on software testing that has shown that the earlier a fault is identified, the less expensive it is to fix it.

Another challenge identified was that of testing Right-to-Left (RTL) languages (PS1; PS5; PS16; PS18; PS10). Despite Arabic being the sixth most spoken language in the world (The 12 most spoken languages in the world), the authors of (PS16) claim that the quality of Arabic localization (or "*arabization*") is still very poor and does not meet minimum quality standards found in other localized software. The authors mention that even in big tech companies, such as IBM and Microsoft, the quality level for the Arabic language is barely adequate. The authors attribute this poor quality to the neglect of particularities of the Arabic language during the internationalization and localization process, which makes it difficult for developers and testers to produce high-quality RTL localized variants.

We identified four main challenges faced by localization and internationalization testing: the lack of scientific research conducted on the topic; high manual labor and very limited automation support; the cost of fixing internationalization faults; and testing right-to-left (RTL) languages.

3.1.8.2 RQ2: Strategies

For answering RQ2, the primary studies were classified according to their test target, namely system, integration, or unit testing. Considering the primary studies that explicitly mention the test target, the vast majority of the studies (11 out of 18) focus on system testing (PS1; PS2; PS4; PS7; PS8; PS9; PS3; PS10; PS11; PS12; PS13). One primary study concentrates only at the unit testing level (PS5); and one study reported the adoption of i18n and l10n testing at both unit and system levels (PS12). The results observed are in line with the common practice of performing i18n and l10n testing at later stages of development, when the functional requirements and user interface have been finalized. This is because failures identified through i18n and l10n testing are typically highly visible on the UI.

Considering the primary studies that explicitly mention the test target, the vast majority of the studies focus on system testing, which is aligned with the common practice of performing i18n and l10n testing at later stages of development.

3.1.8.3 RQ3: Tooling

To answer this question, we have collected all the tools reported by the primary studies for the automation of internationalization and localization testing. The reported tools can be divided into two categories: tools developed specifically for i18n and l10n testing (PS4; PS8; PS9; PS11), and generic tools that were applied in the context of i18n and l10n testing. Below we describe each of these tools:

- Test* (PS11): a testing tool that automatically and dynamically generates, executes and verifies test sequences based on a tree model that is derived from the software's user interface through assistive technologies. In (PS11) the authors use Test* to find failures such as words that are inconsistent between Spanish from Spain and Latin America.

- ITdroid (PS4): a tool for automatically translating strings and detecting bad practices and collateral changes introduced in the GUI of Android apps after translation. The tool is also able to indicate hard-coded strings by performing static analysis on the application.

- Gwali (PS9): A tool to detect Internationalization presentation failures in web applications. It works by comparing the user interface of an application in different languages and detecting inconsistencies in the layout. (PS6) and (PS2) reported the use of Gwali to find `i18n` failures and the use of that information for repairing the identified `i18n` issues.

- In (PS8) the authors report the use of scripts that browse the UI, take screenshots of the visited screens and extract data such as text strings, size and position of its elements. The scripts are executed for the English version of the software and later for the localized version. After that, a comparison is made in order to find UI inconsistencies.
- Cucumber: an open source tool for test automation in the context of behavior-driven development (BDD). In (PS5) Cucumber was used to automate concrete executable specifications to evaluate an app with Righ-to-Left languages.
- Espresso, Robotium, Junit: testing frameworks for the Android platform. In (PS5) these frameworks were used to automate the UI testing of an internationalized app.

The tooling support for i18n and l10n testing is very limited, which could indicate a lack of attention given to this type of testing. Out of the surveyed primary studies, only four tools were identified as specialized tools for the context of i18n and l10n testing. This suggests that more research and development are needed.

3.1.8.4 RQ4: Context

RQ4 is concerned with the context where i18n and L10n testing are applied, in order to understand if the strategies of this type of testing change based on the context. We identified three main contexts: Web (PS2; PS6; PS9; PS11), Desktop (PS7; PS8; PS10; PS12; PS13), and Mobile (PS1; PS4; PS5; PS3).

While the basic principles of i18n and L10n testing apply to all the identified contexts, there are nuances and unique challenges for each type of application. For example, mobile devices often have touchscreens with virtual keyboards, while web and desktop applications typically use physical keyboards. This can affect the way text is inputted and displayed, and can affect how the application handles different languages.

We identified three main contexts for i18n and L10n testing: web, desktop, and mobile. Given the nuances and unique challenges of each context there is a need for specialized tools and techniques that are tailored to each context to ensure effective i18n and L10n testing.

3.1.9 Challenges and opportunities

In this section we summarize the main challenges and opportunities we identified in our systematic mapping study.

(CH1) Challenge #1: Lack of tooling support. The process of i18n and L10n testing typically involves testing a large number of supported *locales*, which can result in the need for large test suites. Therefore, automation can be a strong candidate for ensuring effective and efficient testing. However, the current tooling support for this type of test is still limited and needs further development.

(CH2) Challenge #2: Lack of support material. As highlighted by (PS7; PS5; PS3; PS8), there is a notable lack of empirical studies and practical guidance on the topic of i18n and L10n testing. This gap in research and empirical results leads to a lack of understanding of best practices, challenges, and solutions for i18n and L10n testing. Such gap can result in inefficient and ineffective i18n and L10n testing processes, hindering the development of high-quality localized software products.

(CH3) Challenge #3: Testers training and qualification. The field of i18n and L10n testing is not as widely recognized or studied as other types of testing, which can make it difficult for new professionals to find relevant resources and guidance. In i18n and L10n testing, it is often necessary to reference data/hour standards, linguistic and cultural aspects, and other information that may not be easily accessible or consolidated in a single resource. Another important aspect of i18n and L10n testing is the need for empathy and understanding of different cultures and user perspectives. Testers must be able to put themselves in the shoes of users from different cultural backgrounds and use the localized software as if they were the end users. For example, a tester used to a left-to-right interface must be able to test a right-to-left software with consideration for its target users. This cultural aspect should be addressed in the training and development of i18n/ testers.

(CH4) Challenge #4: Unbalanced support for different locales. The support for i18n and L10n testing varies across different locales. For instance, even though Arabic is the sixth most spoken language in the world, the available support material for right-to-left

(RTL) locales is not as comprehensive as for other languages (Abufardeh and Magel 2008). This imbalance can be attributed in part to the dominance of certain languages in the global economic scenario. However, it is important to strive for a more balanced support for i18n and L10n testing across different locales, in order to ensure that all users, regardless of their language and cultural background, have access to high-quality and localized software.

(OP1) Opportunity #1: Tools development. This opportunity is related to the challenge **CH1**. Given the limited tooling support for i18n and L10n testing, the development of specialized tools represents an opportunity for both academics and practitioners to contribute to the field and improve the efficiency and effectiveness of i18n and L10n testing processes.

(OP2) Opportunity #2: Producing support material. There is a clear research gap in the field of i18n and L10n testing. Some of the primary studies reviewed highlight the need for further research. A quick search on academic search engines can show that the number of studies focused on i18n and L10n testing is relatively low compared to other testing topics. More research is needed to advance the field of i18n and L10n testing and this represents an opportunity to researchers.

(OP3) Opportunity #3: Exploring new technologies. The emergence and proliferation of large language models (LLMs) have the potential to revolutionize the field of i18n and L10n testing. Some companies currently rely on machine translation (MT) to translate software strings, which can save costs but sometimes leads to inaccurate translations (Mulero et al. 2012). If LLMs can be developed as a more reliable alternative for localizing strings, they could represent a significant opportunity for both practitioners and researchers in the field.

3.1.10 Threats to Validity

Secondary studies play a crucial role in synthesizing existing knowledge and identifying research gaps in a particular domain. However, these studies may suffer from validity threats that can compromise the accuracy and reliability of their findings. To ensure the validity of this mapping study, we followed the guidelines proposed by (Ampatzoglou et al. 2019) to mitigate the three main categories of validity threats to secondary studies: study selection validity, data validity, and research validity.

Study Selection Validity: To avoid biases in the selection of primary studies, we adopted a state-of-art search strategy evaluated by (Mourão et al. 2020), which combines automatic search in the Scopus search engine with parallel Snowballing. We also conducted the selection procedure in pairs to reduce personal subjectivity during the analysis of papers. Additionally, we applied systematic voting, discussion among authors during conflict resolution meetings, random paper screening, and inclusion/exclusion criteria in the review protocol.

Data Validity: To ensure the accuracy and reliability of the extraction and synthesis procedures, we assessed the quality of relevant studies and conducted reliability checks (Ampatzoglou et al. 2019). The extraction and synthesis procedures were also reviewed by another researcher to reduce the risk of errors.

Research Validity: To mitigate threats to the overall research, we involved more than one researcher, made the data available in the replication package, developed a protocol, and held discussions among authors (Ampatzoglou et al. 2019). These strategies were adopted to enhance the transparency, replicability, and rigor of the study.

Despite our efforts to mitigate the validity threats, we acknowledge that the topic of this mapping study has limited research, which reduces the confidence and generalizability of our results. We also observed many low-quality papers lacking empirical evidence and poorly executed from the perspective of scientific methodology.

3.2 CONCLUDING REMARKS

This chapter reports the results of a SMS on i18n and l10n testing. We searched primary studies on the topic by querying the Scopus database, and by performing a backward snowballing cycle. We started with a set of 1073 papers and after applying our inclusion/exclusion and quality evaluation criteria, we selected a final set of 18 primary studies that were used as a basis for answering the posed research questions, and for identifying the main challenges and opportunities in the field. Based on these findings, we decided to address the demand for training novice `L10n` and `i18n` testers. In Chapter 4 our proposed approach to contribute with the `L10n` and `i18n` testing community is described.

4 L10N-TRAINER

From the results of the SMS we could further understand the main challenges faced by the practitioners of `L10n` and `i18n` testing, as well as their needs. We decided to contribute with the community by facing the challenge of training novice `L10n` and `i18n` testers. Since the tests cases are developed and executed by testers, a good starting pointing is to aid the training of the novice `L10n` and `i18n` testers in order to have hard skilled professionals. We sought a company that professionally conducts `L10n` and `i18n` testing to carry out our study in order to understand how novice `L10n` and `i18n` testers are trained in their environment and how we could contribute to the improvement of this process. In this chapter we describe the context where the study was performed and we introduce L10N-TRAINER[1], the tool we developed to assist in the training of novice `L10n` and `i18n` testing.

4.1 CONTEXTUALIZATION

The company where this study was conducted provides testing services to a smartphone manufacturer. The products tested are designed to be marked worldwide, thus one of the company's tests team is fully dedicated to `L10n` and `i18n` testing, which has about 11 members and has a testing scope of approximately 100 locales. In this context, the locales are prioritized by tiers, so some locales receive more attention than others during testing activities.

The `L10n` and `i18n` testing team assures the quality of the software from new products as well as maintenance and upgrades, from operational system versions (Android platform) to mobile applications. The team is sometimes involved with the testing of products/software that other teams from this testing company are not, mostly due the particularity of the work performed by them. One of these activities is the quality assurance of marketing applications and validations of texts from Play Store. Thus, the work performed by `L10n` and `i18n` testing team is easily noticed by the end user.

Usually, for new products and Android upgrades there is a specific testing suite to verify the functionalities from the software for all supported locales by the software, 101 locales. This testing suite has about 2500 test cases, from which about 200 tests are manual and one of them is an exploratory test. This testing suit is usually executed 2 times by product, once before the product market release and another before the product receives a new Android version.

[1] https://raquelcouto.github.io/L10N-TRAINER/

The automated test cases from this suite have scripts that are executed with the aid of tool developed in house. Besides this test suite, there is also suites for regression and exploratory tests. However, execution frequency of them is lower, usually once a year. Applications also have specific test plans, the frequency of execution is higher due the maintenance of these apps on Play Store. The test plan for apps, includes manual validation for app description used in Play Store and validations related to the functionalities of the application for all supported locales.

Annually, all testing scripts used by the `L10n` and `i18n` testing team need to be reviewed for each Android released version in order to check if they are still functional. Besides that, frequently, the automated test case scripts need to be updated due the variation in the features from product to product or layout changes in the applications. When this happens during a test suite execution, sometimes the time used to fix a script is higher than executing the test cases manually. In these cases, the tester occasionally needs to execute the same test case for several locales. This situation is a double-edged sword: the script validations are only functional, which means that the script only checks if the applications are localized, verifies if no crashes are observed under a specific locale, and checks data format issues. The manual validation by a human can expose issues that are only observed by visual verification, such as ellipsis, truncation, overlapping, term inconsistency, grammar related issues and RTL related failures. On the other hand, the tester needs to be well-trained to easily identify these issues in a language with which they are not familiar.

Besides the test suits mentioned previously, the `L10n` and `i18n` testing team also executes further tasks with more manual interaction, such as visual validation of the longest strings on the UI and retest of the reported failures.

4.2 PROBLEM

Santos et al. (Santos et al. 2020) state that internationalization tests are not as popular as other types of tests, which makes it difficult for professionals to obtain resources, tools and study material to rely upon. Quoting the authors, "*training activities are usually affected by the lack of information available about this practice and even skilled testers who started to work in this particular context might face difficulties to understand the required validations*". Normally, novice `L10n` and `i18n` testers first encounter this practice during the training stage. Is common that the novice `L10n` and `i18n` testers have never heard about this type of testing before and even associate the name “localization” to tests related to map positioning.

In the context of the real industrial setting studied by us in this work, the training of novice L10n and i18n testers typically involves their active participation in daily activities and the execution of real test suites. A training test suite has about 800 testing cases, from this total 100 are meant to be executed manually whilst the others are automated. The main objectives of the training are to familiarize the novice with the test cases executed by the team, teaching them about the team tools/process and how to look into references for checking data format issues. The problem with this approach is that very often the software version used during the training phase is stable and the novice L10n and i18n testers might complete the training without observing any real i18n/L10n failure. Besides that, since most of the test cases are automated, the novice might not be well trained to perform manual testing, which is crucial for L10n and i18n testing due the fact that there are issues that cannot be checked without manual interaction.

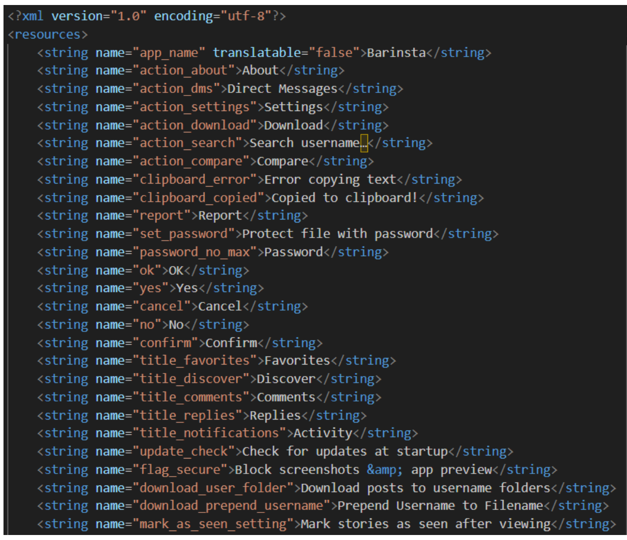
Therefore, to ensure the quality of the localized software it is necessary to ensure the quality of the training of novice L10n and i18n testers. It is important that the testers are well trained so that they can distinguish real L10n/i18n faults from expected behavior. Motivated by this problem faced by our industrial partner we developed L10N-TRAINER: a tool to assist in the training of localization (L10n) and internationalization (i18n) testers.

4.3 SOLUTION

The tool developed in this work is proposed to be used for the training of novice i18n/L10n testers. The potential users of L10N-TRAINER are testing practitioners, both experienced testers (in the role of trainers) and novice L10n and i18n testers (in the role of trainees). The main idea behind the tool is that of seeding realistic L10n/i18n faults to assist in the training of novice L10n and i18n testers. Because real apps are used in the training, the novice L10n and i18n testers get exposed not only to i18n/L10n failures, but also to manual testing and exploratory testing concepts.

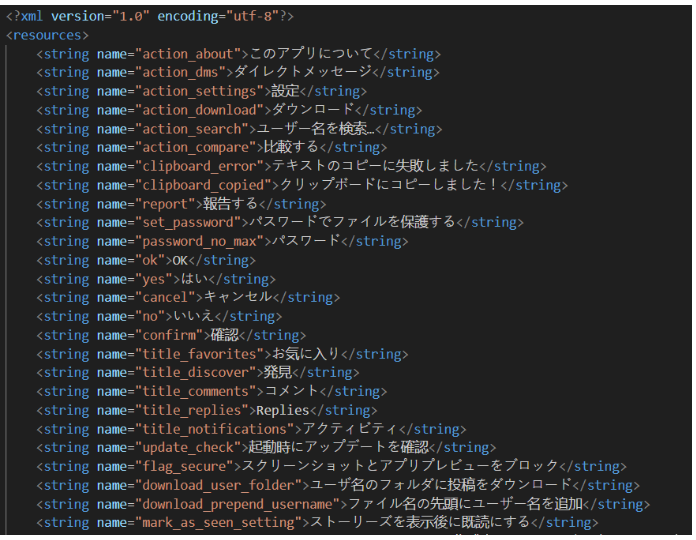
In globalized applications it is generally the case that the source language is English, since most of the applications are firstly developed in this language. When all strings from the application are final, their localization is done from English to the target languages that shall be supported by the SUT. eXtensible Markup Language (XML) files are usually used to store the string values for the source and target languages. The data stored in the XML

Figure 17 – Example of XML files for an application.



```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name" translatable="false">Barinsta</string>
    <string name="action_about">About</string>
    <string name="action_dms">Direct Messages</string>
    <string name="action_settings">Settings</string>
    <string name="action_download">Download</string>
    <string name="action_search">Search username…</string>
    <string name="action_compare">Compare</string>
    <string name="clipboard_error">Error copying text</string>
    <string name="clipboard_copied">Copied to clipboard!</string>
    <string name="report">Report</string>
    <string name="set_password">Protect file with password</string>
    <string name="password_no_max">Password</string>
    <string name="ok">OK</string>
    <string name="yes">Yes</string>
    <string name="cancel">Cancel</string>
    <string name="no">No</string>
    <string name="confirm">Confirm</string>
    <string name="title_favorites">Favorites</string>
    <string name="title_discover">Discover</string>
    <string name="title_comments">Comments</string>
    <string name="title_replies">Replies</string>
    <string name="title_notifications">Activity</string>
    <string name="update_check">Check for updates at startup</string>
    <string name="flag_secure">Block screenshots &amp; app preview</string>
    <string name="download_user_folder">Download posts to username folders</string>
    <string name="download_prepend_username">Prepend Username to Filename</string>
    <string name="mark_as_seen_setting">Mark stories as seen after viewing</string>
```

(a) XML file for English strings.



```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="action_about">このアプリについて</string>
    <string name="action_dms">ダイレクトメッセージ</string>
    <string name="action_settings">設定</string>
    <string name="action_download">ダウンロード</string>
    <string name="action_search">ユーザー名を検索…</string>
    <string name="action_compare">比較する</string>
    <string name="clipboard_error">テキストのコピーに失敗しました</string>
    <string name="clipboard_copied">クリップボードにコピーしました！</string>
    <string name="report">報告する</string>
    <string name="set_password">パスワードでファイルを保護する</string>
    <string name="password_no_max">パスワード</string>
    <string name="ok">OK</string>
    <string name="yes">はい</string>
    <string name="cancel">キャンセル</string>
    <string name="no">いいえ</string>
    <string name="confirm">確認</string>
    <string name="title_favorites">お気に入り</string>
    <string name="title_discover">発見</string>
    <string name="title_comments">コメント</string>
    <string name="title_replies">Replies</string>
    <string name="title_notifications">アクティビティ</string>
    <string name="update_check">起動時にアップデートを確認</string>
    <string name="flag_secure">スクリーンショットとアプリプレビューをブロック</string>
    <string name="download_user_folder">ユーザ名のフォルダに投稿をダウンロード</string>
    <string name="download_prepend_username">ファイル名の先頭にユーザー名を追加</string>
    <string name="mark_as_seen_setting">ストーリーズを表示後に既読にする</string>
```

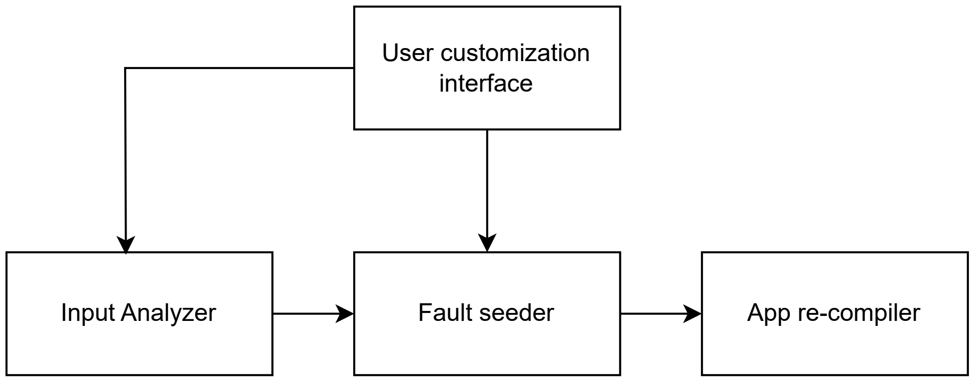
(b) XML file for Japanese strings.

Source: Created by the author.

files usually consists of the *ids* (unique for each string) and the *values* of the strings. For instance, a software that supports French, Portuguese, Spanish and Greek, will have an XML file containing the translated strings for each of these locales. In Figure 17 there is an example of XML file for English (Figure 17a) and Japanese (Figure 17b) strings, the content the double quotes are the *ids* for each string, the content highlighted in white are the values for these ids. Depending on the language chosen by the user, the system will load the corresponding XML data. In addition, there is also other types of strings files, such as JavaScript Object Notation (JSON), which can be used for web and flutter applications.

For the development of L10N-TRAINER we decided to use Python as the programming language due to its extensive library support and ease of handling different types of files. An overview of the tool modules is displayed in Figure 18. Module ***user customization interface*** is the interface with the end user, the trainer, by mean of it is possible to configure the parameters for seeding an application to simulate L10n/i18n faults, later we describe the steps and parameters available in L10N-TRAINER. From the user's inputs the module ***input analyzer*** verifies the provided parameters and validates if the XML file is adequate for inserting the faults. Further explaining, the file might not contain translatable strings or have a different format from XML. The main module of the tool is the ***fault seeder*** module, it seeds the provided file with the number of faults chosen by the trainer, this module is more detailed in the next subsections. To recompile the applications with the seeded faults, the module ***app re-compiler*** is triggered, it will provide to the user a new version of the application containing the `L10n/i18n` failures.

Figure 18 – L10N-TRAINER modules.



Source: Created by the author.

4.3.1 Fault Seeding

The fault seeding process can be either manual or fully automated. In the case of manual seeding, trainers have the flexibility to choose which faults to seed and their respective locations. The fully automated process is based on value mutation operators. Mutation testing is a fault-based testing technique, the principle underlying it is that the faults used by mutation testing represent the mistakes that programmers often make (Jia and Harman 2010). In this sense, there are some types of mutation operators, such as: ***value operators*** that changes the value of the code variables, ***decisions mutations*** that are logical or arithmetic operators are changed to detect errors and ***statement mutations*** that occurs when a statement is deleted or it is replaced by some other. In the case of our tool we used the value operator in order to modify the values from the applications strings. This way it is possible to play around with the strings size causing ellipsis, truncation, overlapping or even changing the string value to another language in order to cause a missing translation. This way, trainers can control the quantity and types of faults while leaving the decision of where to seed the faults to our tool.

Regardless of the chosen fault seeding strategy, L10N-TRAINER receives as input: a target app, an XML file, and a list of faults to be seeded. For the fully automated fault seeding process an additional parameter is a target language. This is required for translating the original app language to the desired target language.

4.3.1.0.1 Manual Fault Seeding

L10N-TRAINER receives the required input, that is the pair variables and values to be altered, then it modifies the necessary files and compiles a new version of the target app containing the i18n/L10n faults defined by the trainer.

4.3.1.0.2 Automated Fault Seeding

The process is analogous to that of manual fault seeding. The only difference is that L10N-TRAINER will make use of the Google Translate API[2] to translate some of the original app strings to the desired target language. The tool will randomly choose the n (where n is chosen by the trainer) strings to be modified, translate them and then overwriting their values into the string file (XML or JSON). In this process, L10N-TRAINER can generate different types of i18n/L10n failures, from mistranslated strings to overlap and ellipsis (since the UI may no longer be able to support the size of the new strings).

After seeding the string file, the tool generates a modified version of the application. For React Native apps a QR code is displayed on the screen, by reading this QR code with a mobile phone is possible to load the seeded application. For Android apps the tool generates a new apk by communicating with Android Studio, a path for the modified apk is provided on the screen. Then, the application needs to be installed on the mobile Android phone. Finally, the trainer delivers the mobile device with the seeded applications to the novice tester starts the exploration in the searching the faults.

4.3.1.0.3 Dummy Applications

For supporting the training task, our tool is delivered with a set of dummy applications that can be used by trainers to seed i18n/L10n faults and conduct training sessions with novice L10n and i18n testers. The "dummy" apps are actually simplified versions of real apps with limited functionality. They are specifically designed for training purposes to enhance development skills and familiarize individuals with various aspects of app development. Due to the open-source nature of many dummy apps, integrating them with L10N-TRAINER requires minimal or no adaptation. L10N-TRAINER can seamlessly work with these open-source dummy apps, leveraging their existing structure and functionality.

[2] <https://pypi.org/project/googletrans/>

Currently, L10N-TRAINER is delivered with two dummy apps: *Vinted* (Vinted), a react-native open source application for clothing sales; and *Barinsta*(masuvern), an Android Instagram client.

React-native apps can be tested with the aid of Expo Go (Expo Go), an open-source framework for apps that run nativity on Android, iOS and the web. Expo Go allows us to use the application remotely by creating a server and sharing the QR code generated specifically for the target project. The use of the application anywhere is an interesting feature that can enabled the remote training of novice `L10n` and `i18n` testers.

Therefore, any internationalized Android or React Native open-source application can be seeded with L10N-TRAINER. The use of an internationalized application is needed since the tool uses the strings files to generate the faults.

Figure 19 displays an example of `i18n`/`L10n` failures seeded on the Vinted app with the use of our tool. The picture on the left contains an example of overlap and missing translation issues. In the right picture an issue of `i18n` is shown since the money symbol is displayed in Euro (€) and the application was set to `pt-BR` locale, that uses $R\$$ as currency symbol. Missing translation issues are illustrated in the strings 'see more', 'home' and 'inbox' that were left in English. Please note that the inclusion of a high number of seeded faults in a single screen app is for illustrative purposes only. In actual use scenarios, trainers may prefer to seed a single fault strategically placed within the app screens.

4.3.2 Walkthrough

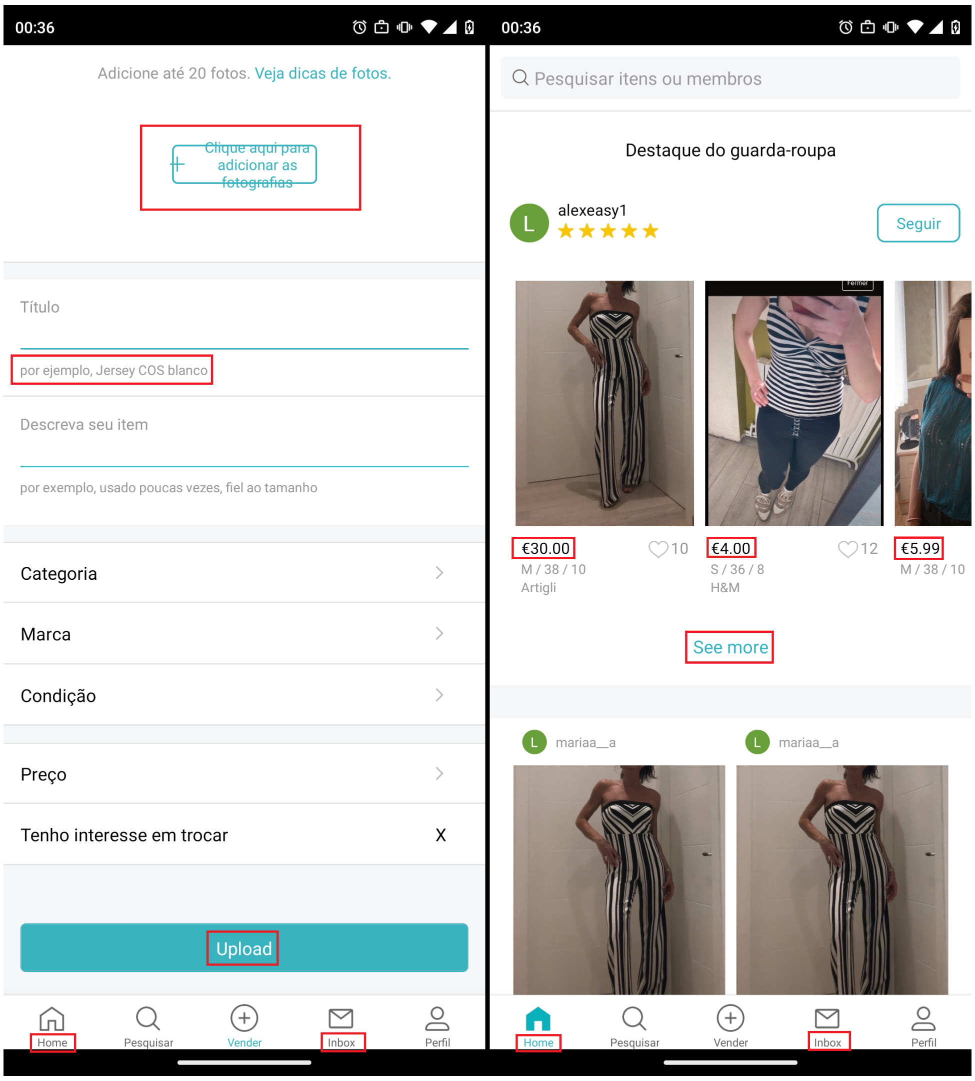
In this section we present the L10N-TRAINER UI and describe how to use it. Figure 20 displays the main screen of the tool.

Before conducting a new training session, the trainer should choose a mobile application to be seeded, and configures L10N-TRAINER to seed it with `L10n`/`i18n` faults. The trainer plans the type of failures (e.g Ellipses, truncation, overlapping) that will be inserted in the app and where in UI they are going to be displayed.

The trainer provides the path for the chosen application's string file that will be modified, for example choosing the file containing the German strings from an application($\#1$ in Figure 20, `Path to string file`).

Next, the trainer defines the number of random faults to be seeded in the training app ($\#2$ in Figure 20, `Number of random faults`). By default, if the trainer leaves this field

Figure 19 – `i18n/L10n` failures in the application Vinted.



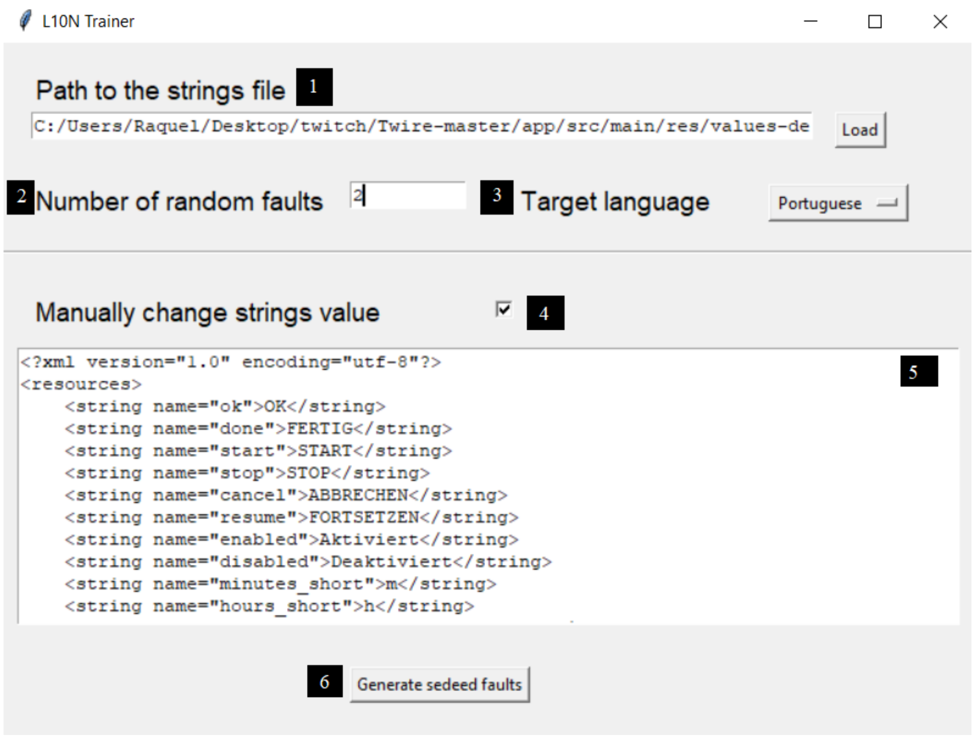
Source: Created by the author.

empty the number of random faults will be zero and no random faults will be inserted into the application.

In case the trainer wants to seed random faults, the next step is to select the language into which the strings will be translated to(#3 in Figure 20, `Target language for translation`). A list of languages is loaded, allowing the trainer to choose the specific language to be used for translation (Figure 21). For instance, if the trainer has chosen the German string file and wants that the n random faults be generated in French, this language should be selected in this field.

Finally, the trainer can insert faults manually (#4 in Figure 20, `Manually change string values`). In that case, the manual edition of the strings will be allowed (#5 field in Figure 20), when the chosen file is loaded all the strings from the file are loaded in that field.

Figure 20 – L10N-TRAINER UI.



```
C:/Users/Raquel/Desktop/twitch/Twire-master/app/src/main/res/values-de
```

Number of random faults: 2

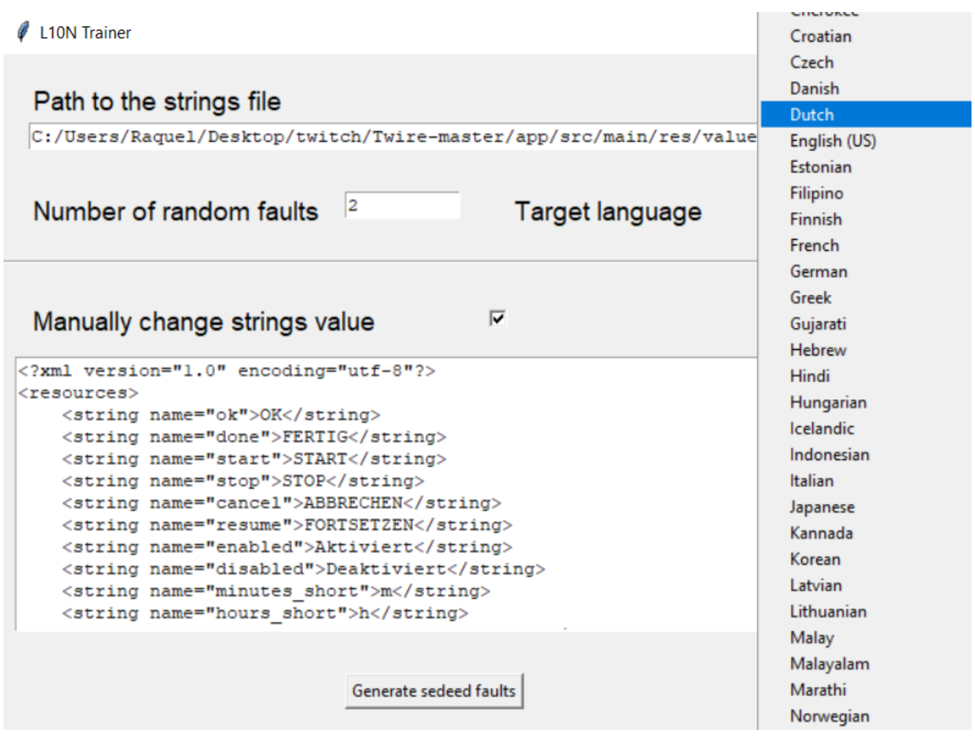
Target language: Portuguese

Manually change strings value [✓]

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="ok">OK</string>
    <string name="done">FERTIG</string>
    <string name="start">START</string>
    <string name="stop">STOP</string>
    <string name="cancel">ABBRECHEN</string>
    <string name="resume">FORTSETZEN</string>
    <string name="enabled">Aktiviert</string>
    <string name="disabled">Deaktiviert</string>
    <string name="minutes_short">m</string>
    <string name="hours_short">h</string>
```

Source: Created by the author.

Figure 21 – L10N-TRAINER languages selector.



```
C:/Users/Raquel/Desktop/twitch/Twire-master/app/src/main/res/value
```

Number of random faults: 2

Manually change strings value [✓]

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="ok">OK</string>
    <string name="done">FERTIG</string>
    <string name="start">START</string>
    <string name="stop">STOP</string>
    <string name="cancel">ABBRECHEN</string>
    <string name="resume">FORTSETZEN</string>
    <string name="enabled">Aktiviert</string>
    <string name="disabled">Deaktiviert</string>
    <string name="minutes_short">m</string>
    <string name="hours_short">h</string>
```

Source: Created by the author.

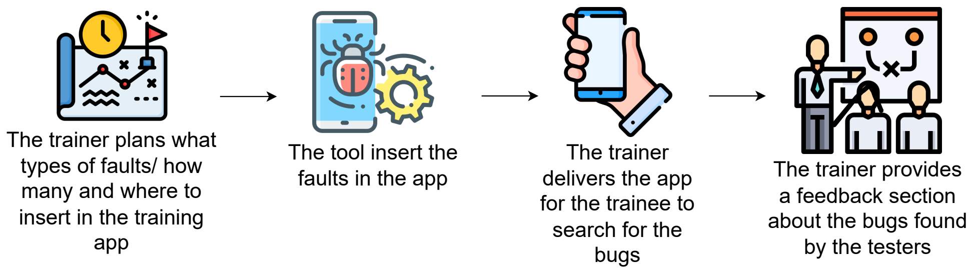
The trainer can decide to seed the faults automatically, manually, or by combining the two approaches. After performing all the configurations the trainer can click in the button

'Generate seeded faults' (#6 in Figure 20) to generate a new version of the app with the selected seeded L10n/i18n faults.

If the application selected for the user is a React-Native application, a UI containing the QR code that leads to the application will be loaded (Figure 23. For Android application, a path for the generated apk file will be provided.

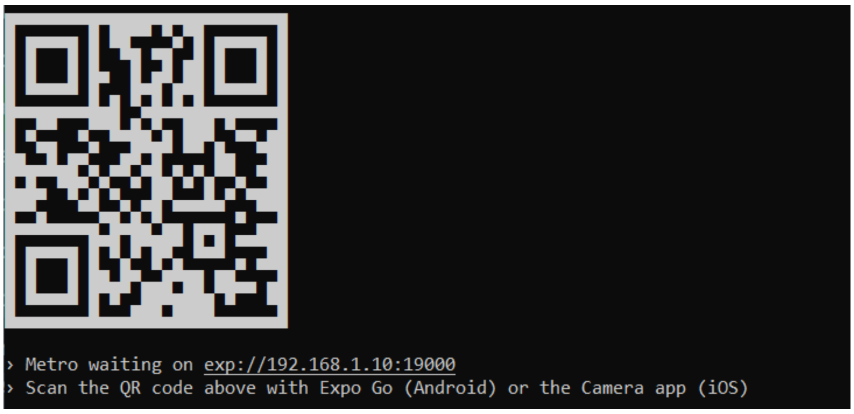
Finally, when the seeded application is generated, the trainer schedule a training section with the novice L10n and i18n testers and then a feedback section to discuss about the failures found or not found, the full process is illustrated in Figure 22.

Figure 22 – Training steps with L10N-TRAINER approach.


The trainer plans what types of faults/ how many and where to insert in the training app
The tool insert the faults in the app
The trainer delivers the app for the trainee to search for the bugs
The trainer provides a feedback section about the bugs found by the testers

Source: Created by the author.

Figure 23 – QR code generated for React Native applications.



```
› Metro waiting on exp://192.168.1.10:19000
› Scan the QR code above with Expo Go (Android) or the Camera app (iOS)
```

Source: Created by the author.

4.4 CONCLUDING REMARKS

In the previous Chapter, we presented a SMS in the field of (L10n) and (i18n) testing. One of the main challenges identified was the lack of tools for supporting the practice. None of the few tools presented in the Chapter was target to aid in the training of novice L10n and i18n testers.

This chapter has presented L10N-TRAINER, a novel tool designed to assist in the training of (L10n) and (i18n) testers. The tool seeds (L10n) and (i18n) faults in Android or react-native applications in order to simulate real L10n and i18n testing failures, allowing that the novice L10n and i18n testers trained with applications seeded by it to be presented these faults during their training stage. Our tool addresses the challenge of effectively training novice L10n and i18n testers to ensure that the software meets the requirements of diverse cultures and languages.

Since we could not find in the literature any tool to compare L10N-TRAINER against, we could not include a related work section in this work. However, in the next Chapter, we present the results of an empirical study with L10N-TRAINER in a real industrial setting.

5 EVALUATION

For assessing the usefulness of L10N-TRAINER we conducted an evaluation in the context of manual testing in a real industrial setting. In this chapter we describe the empirical study design and the results obtained. The replication package of our empirical study design is available online[1].

5.1 PREPARATION

5.1.1 Study design

The objective of our evaluation was to assess the effectiveness of L10N-TRAINER for training novice localization and internationalization testers. In particular, we defined the following research question:

> **RQ1:** How effective is L10N-TRAINER in the training of novice L10n and i18n testers in the context of manual testing?

To answer the posed research question we aimed to design an empirical study replicating the target real-world scenario where the L10N-TRAINER would be employed: an industrial environment engaged in L10n and i18n testing and tasked with training novice L10n and i18n testers. For this empirical study, we had to recruit inexperienced testers and we had the cooperation of a real industrial setting that performs L10n and i18n testing on mobile devices.

To evaluate the effectiveness of L10N-TRAINER when compared with the traditional methodology applied by our industrial partner, we followed these preparation steps:

1. **Seeding faults in the training app.** We used L10N-TRAINER to seed L10n faults in a *dummy* application, hereafter referred as *training app*.

2. **Seeding faults in the testing app.** We asked the development team of the partner company to intentionally introduce analogous L10n faults into one of their official applications, hereafter referred as *testing app*.

[1] <https://github.com/RaquelCouto/Msc-L10N-Trainer>

3. **Recruiting and dividing participants into groups.** The participants of our study were employees from the industrial partner. They were recruited through a Google Forms and divided into two groups.

For selecting the training application (step 1), we considered the criteria of the number of screens of the app. We needed the application to be neither too extensive nor too short to be explored within the available time for the empirical study.

For selecting the official application to be seeded for the empirical study (step 2), we opted for an application that was accessible on all the available devices, ensuring that it had identical features on all the smartphones. The smartphones used in the empirical study were provided by the industrial partner.

Concerning step 3, the participants of the two groups were asked to perform exploratory tests looking for `L10n` issues on both apps. In order to minimize order effects and ensure a balanced distribution of conditions, we used the Latin square design, which involved changing the order of exploration of the test applications. The first group was initially introduced to the training application and then introduced to the testing application. The second group, on the other hand, was first trained with the testing application and then introduced to the training application. We followed this approach to validate the effectiveness of the training application against the traditional methodology (testing application) adopted by the company. In theory, our tool enables testers to learn about localization testing and identify issues by exposing them to failures found during this type of testing in their training stage. Therefore, our approach facilitates the assimilation of failures in production software for novice `L10n` and `i18n` testers. In contrast, the traditional training methodology may not allow testers to identify failures since the software is stable. This implies that experienced testers have been testing this software, and failures are not easily identified by novice `L10n` and `i18n` testers.

5.1.2 Seeded applications

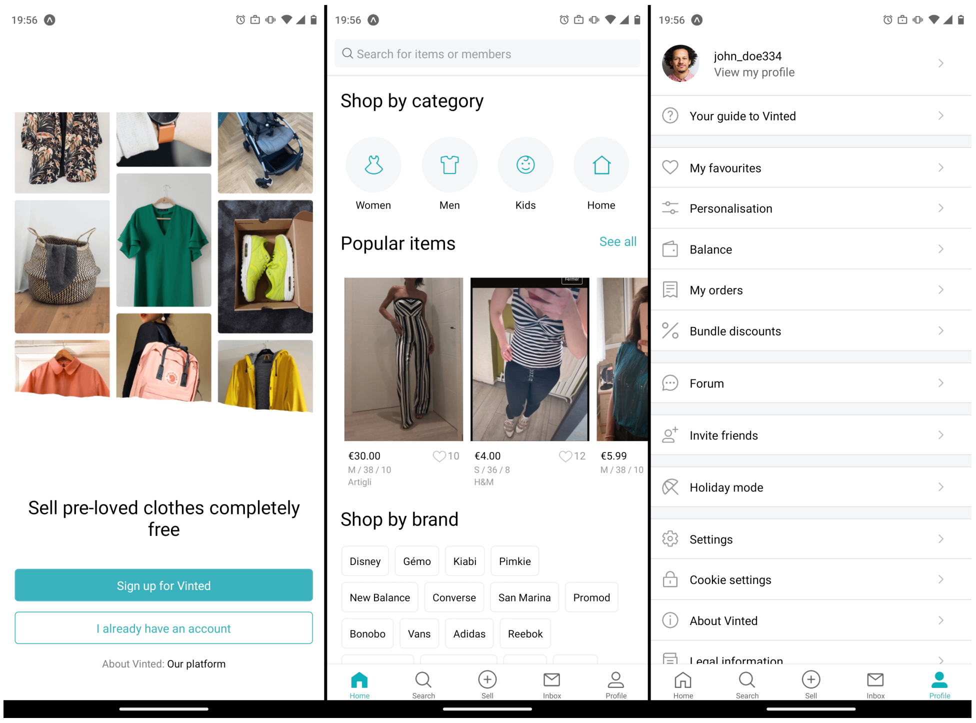
The usual training methodology in the context where the empirical study was executed consists in executing about 100 manual test cases related to multiple applications available in the mobile devices. Since the time available for the empirical study execution was limited due to the participants' availability, we could not reproduce the exact same scenario. Thus, we decided to use only one of the applications tested during the traditional training methodology.

The reason to choose Moto application (detailed later) was to have an application within the similar size of the training application: the app should not be too extensive in order to allow the novice testers to perform an exploratory test session in the available time. Next we briefly describe the applications used in the evaluation.

5.1.2.1 *Vinted*

Vinted (Vinted) is a react-native open source application for clothing sales. Figure 24 displays some screenshots of Vinted application. This application was used in the empirical study as the training app for training novice `L10n` and `i18n` testers.

Figure 24 – Screenshots from Vinted application.

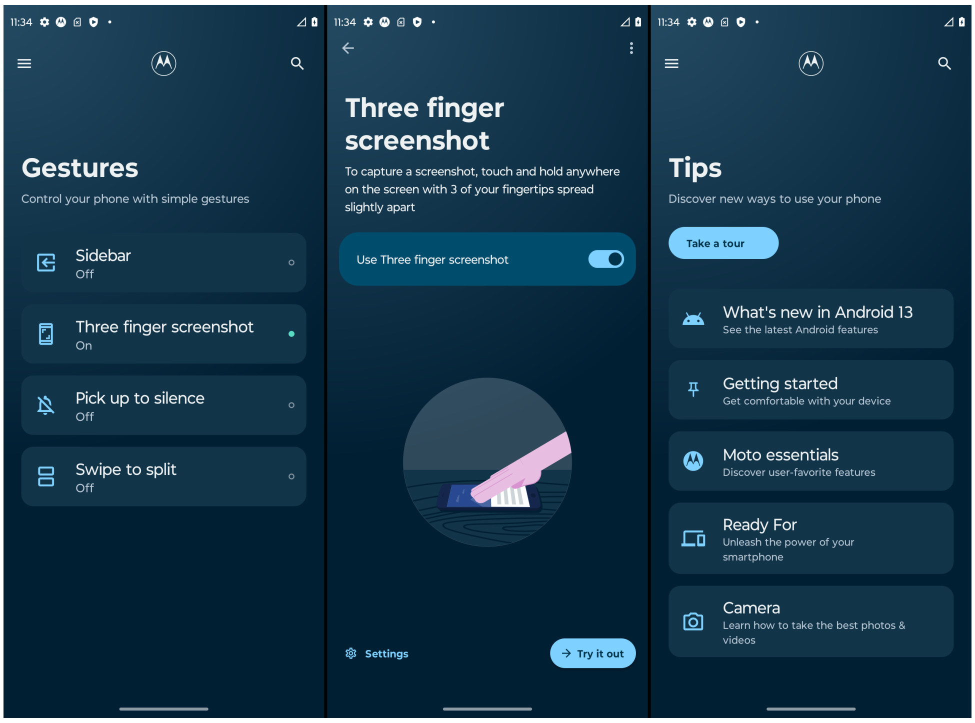


Source: Created by the author.

5.1.2.2 *Moto*

Moto is an Android application developed by our industrial setting collaborator. The app is targeted for mobile devices and provides tutorials and settings related to the experiences offered by the mobile phone. In Figure 25 it is possible to see some screens from the Moto application.

Figure 25 – Screenshots from Moto application.



11:34

Gestures

Control your phone with simple gestures

Sidebar
Off

Three finger screenshot
On

Pick up to silence
Off

Swipe to split
Off

11:34

Three finger screenshot

To capture a screenshot, touch and hold anywhere on the screen with 3 of your fingertips spread slightly apart

Use Three finger screenshot

Settings

Try it out

11:34

Tips

Discover new ways to use your phone

Take a tour

What's new in Android 13
See the latest Android features

Getting started
Get comfortable with your device

Moto essentials
Discover user-favorite features

Ready For
Unleash the power of your smartphone

Camera
Learn how to take the best photos & videos

Source: Created by the author.

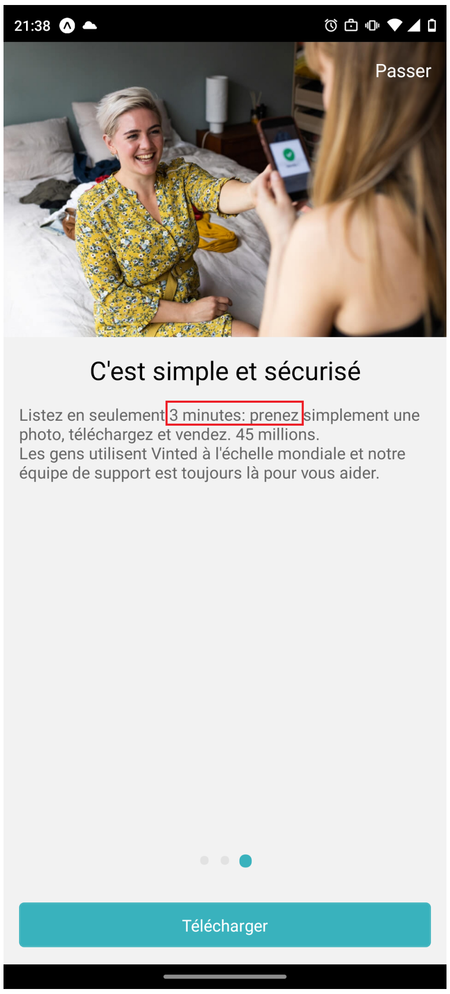
5.1.3 Seeded faults

For the empirical study we decided to seed both applications with two faults. The seeded faults needed to be similar in both apps in order to facilitate a more accurate assessment of the effectiveness of L10N-TRAINER for the task of training of novice testers.

For our evaluation the applications were explored with either French or Portuguese (the native language of the participants). We chose to avoid complex languages with intricate alphabets and complex characters as we did not want to subject the participants to a highly challenging task in their first contact with L10n and i18n testing. Choosing a language with non-American Standard Code for Information Interchange (ASCII) alphabet characters, for example, would had made the empirical study way more difficult than needed to be since the participants were not familiar with the training application. In fact, in that industrial setting, until the novice testers get used to the UI of the applications they tend to use their native language to perform exploratory tests.

Concerning the types of faults seeded, we decided to seed the applications with a **inconsistency of terms** and a **language related** issue. Such a decision was mainly motivated by the limited time available for the execution phase of the evaluation. For more complex issues

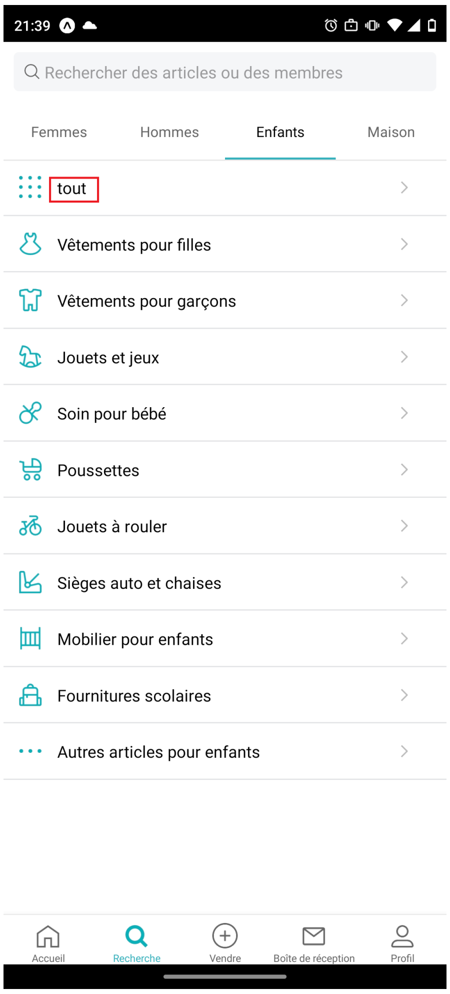
Figure 26 – Issues seeded in Vinted application.



C'est simple et sécurisé

Listez en seulement 3 minutes: prenez simplement une photo, téléchargez et vendez. 45 millions.
Les gens utilisent Vinted à l'échelle mondiale et notre équipe de support est toujours là pour vous aider.

(a) Vinted seeded issue 1.



(b) Vinted seeded issue 2.

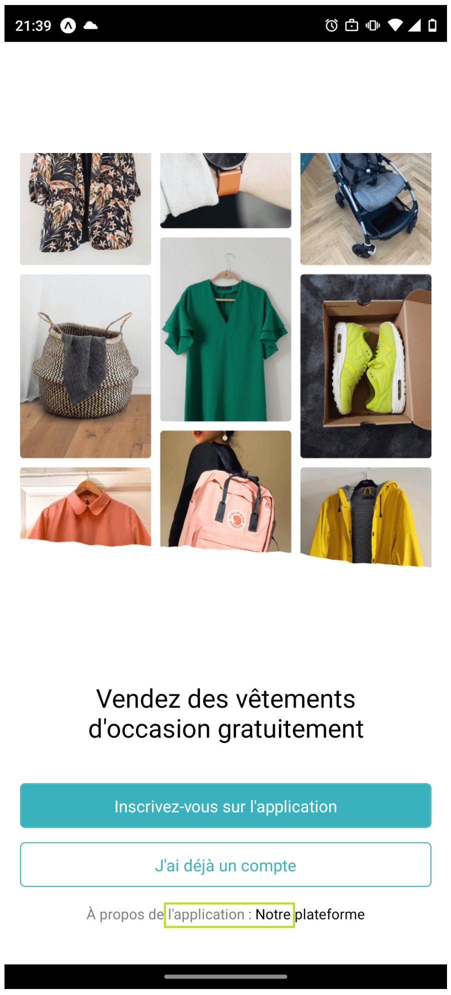
Source: Created by the author.

like data-format we would need more time to show how to check the correct date pattern in the references. Naturally, in a normal training section with more time available, this would be addressed.

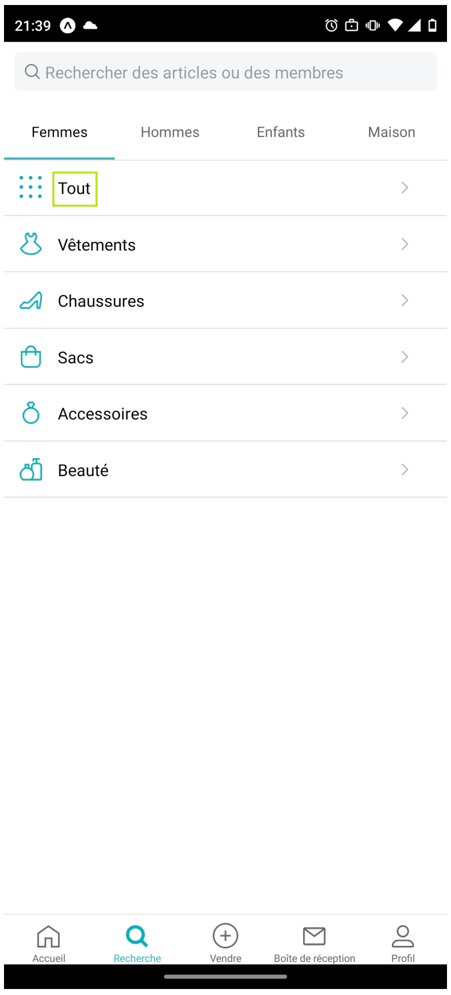
Figure 26 displays the faults that were seeded in the training app. In French there is a rule to have space between the word and its punctuation, and such a space is missing between the word 'minutes' and the colon in Figure 26a. In Figure 26b, there is an inconsistency. The word 'tout' does not start with a capitalized letter whilst all the other elements from the screen are following this pattern. In Figure 27 it is possible to see screens from Vinted application that have the expected behaviour for both faults.

Since we wanted to evaluate the effectiveness of the training app in the training of novice `L10n` and `i18n` testers, we seeded the Moto application with similar issues. Figure 28 displays highlighted in the red color the seeded faults for this application, 28-a similar to 26a and 28b

Figure 27 – Expected behaviour for the faults seeded in Vinted app.


21:39
Vendez des vêtements d'occasion gratuitement
Inscrivez-vous sur l'application
J'ai déjà un compte
À propos de l'application : Notre plateforme

(a) Vinted expected behaviour for issue 1.



(b) Vinted expected behaviour for issue 2.

Source: Created by the author.

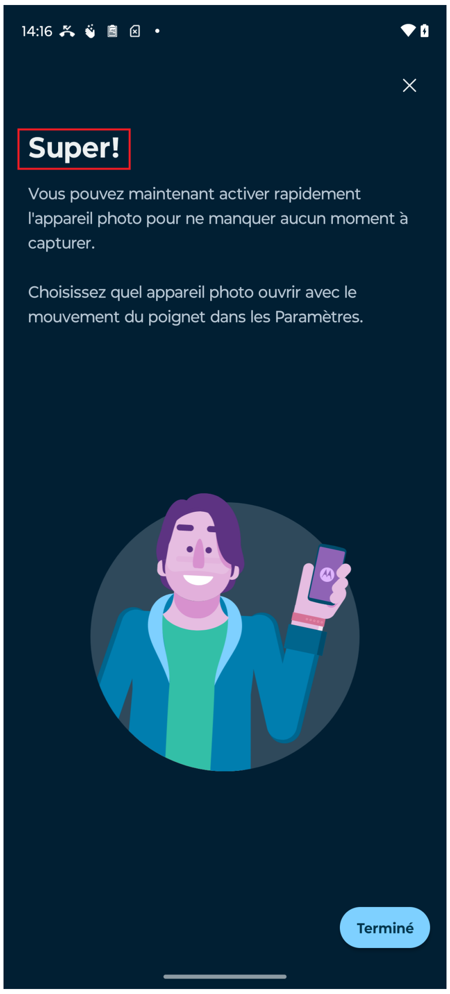
similar to 26b. The expected results for the screens are highlighted in green in Figure 29.

5.1.4 Participants

To conduct our evaluation, we recruited participants from our partner's industrial setting. Specifically, our focus was on novice testers who had no prior experience with `L10n` and `i18n` testing. To invite participants, we created a recruitment form with a total of eight questions. Four of these questions were demographic, while the remaining were aimed at assessing the participant's suitability for our study. The questions from the recruitment form are displayed in Table 4.

Regarding the eligibility questions, EQ1 and EQ2 were employed to assess the participants' proficiency in software testing and exploratory testing, respectively. If there were discrepancies

Figure 28 – Faults seeded on Moto app.

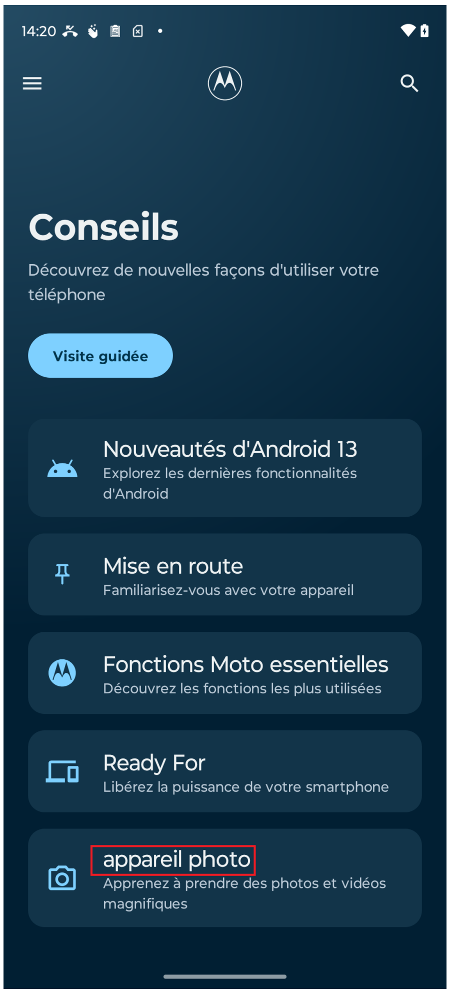

14:16

Super!

Vous pouvez maintenant activer rapidement l'appareil photo pour ne manquer aucun moment à capturer.

Choisissez quel appareil photo ouvrir avec le mouvement du poignet dans les Paramètres.

(a) Moto seeded issue 1.


14:20

Conseils

Découvrez de nouvelles façons d'utiliser votre téléphone

Nouveautés d'Android 13
Explorez les dernières fonctionnalités d'Android

Mise en route
Familiarisez-vous avec votre appareil

Fonctions Moto essentielles
Découvrez les fonctions les plus utilisées

Ready For
Libérez la puissance de votre smartphone

appareil photo
Apprenez à prendre des photos et vidéos magnifiques

(b) Moto seeded issue 2.

Source: Created by the author.

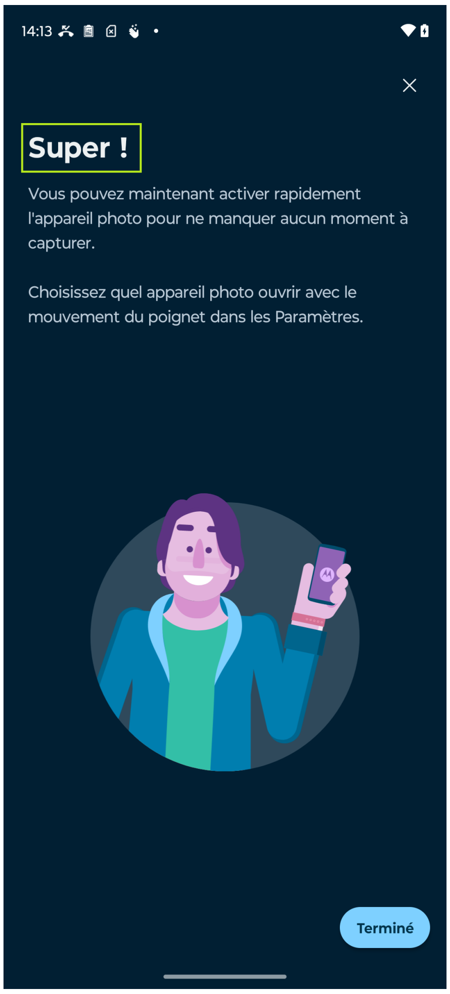
Table 4 – Recruitment form with Demographic questions (DQ) and Eligibility questions (EQ)

ID	Question	Type
DQ1	Name	Text
DQ2	Age	Numeric
DQ3	Gender	Multiple-choice
DQ4	Educational background	Multiple-choice
EQ1	How do you evaluate your knowledge in software testing?	Likert scale
EQ2	How do you evaluate your knowledge in exploratory testing?	Likert scale
EQ3	Have you ever executed `L10n` and `i18n` testing professionally?	Yes/No
EQ4	Have you ever studied French?	Yes/No

Source: Elaborated by the author.

in their proficiency levels, it would be necessary to segment the groups to ensure level calibration. EQ3 and EQ4 served as an exclusion criterion for our study, as participants should not

Figure 29 – Expected behaviour for the faults seeded in Moto app.



14:13

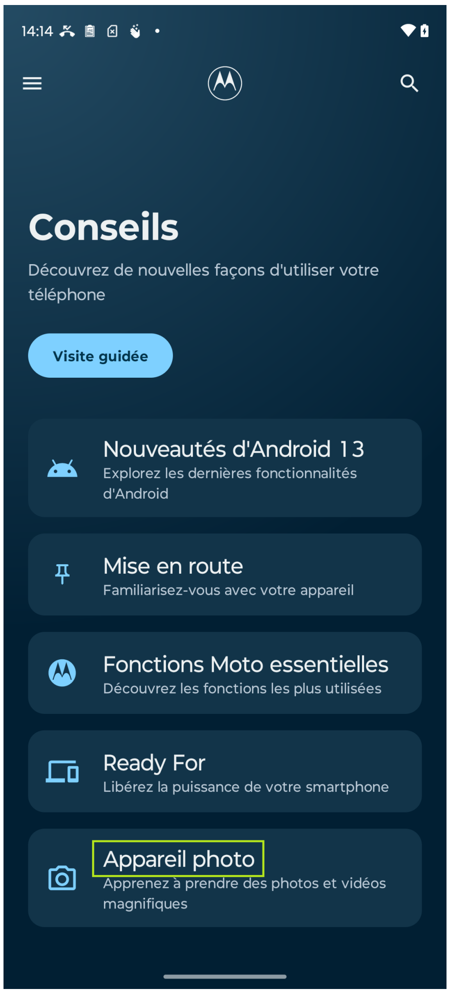
Super !

Vous pouvez maintenant activer rapidement l'appareil photo pour ne manquer aucun moment à capturer.

Choisissez quel appareil photo ouvrir avec le mouvement du poignet dans les Paramètres.

Terminé

(a) Moto seeded issue 1.



14:14

Conseils

Découvrez de nouvelles façons d'utiliser votre téléphone

Visite guidée

Nouveautés d'Android 13
Explorez les dernières fonctionnalités d'Android

Mise en route
Familiarisez-vous avec votre appareil

Fonctions Moto essentielles
Découvrez les fonctions les plus utilisées

Ready For
Libérez la puissance de votre smartphone

Appareil photo
Apprenez à prendre des photos et vidéos magnifiques

(b) Moto seeded issue 2.

Source: Created by the author.

have had prior professional experience with `L10n` and `i18n` testing or previous knowledge in the French language.

Ten participants completed the recruitment form and voluntarily agreed to participate in our study. The age of the respondents ranged from 23 to 42. 80% identified themselves as from the masculine gender. Concerning the educational background, only 3 respondents had a background related to Information Technology (IT), the others had different backgrounds. Furthermore, the participants had no previous knowledge in the French language, which an important aspect for the study since it would lead the groups to have different performances. Table 5 displays the demographic information of the participants. To preserve the participants identity, their names were replaced by identifiers (*P1* to *P10*).

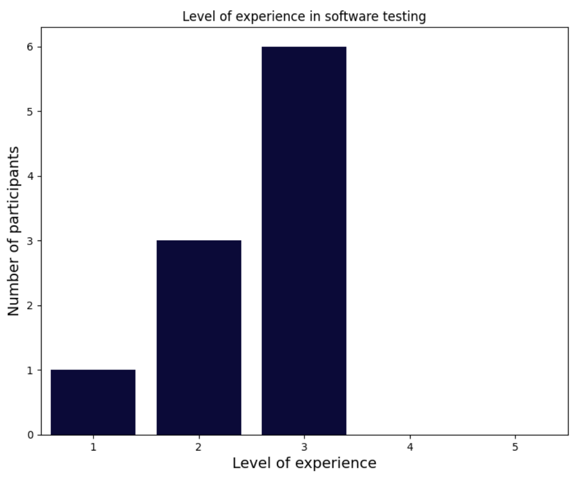
Regarding the eligibility questions, the participants considered themselves as beginner to intermediate in software testing and exploratory testing. Thus, we decided to drawn the groups

Table 5 – Participants Demographic information

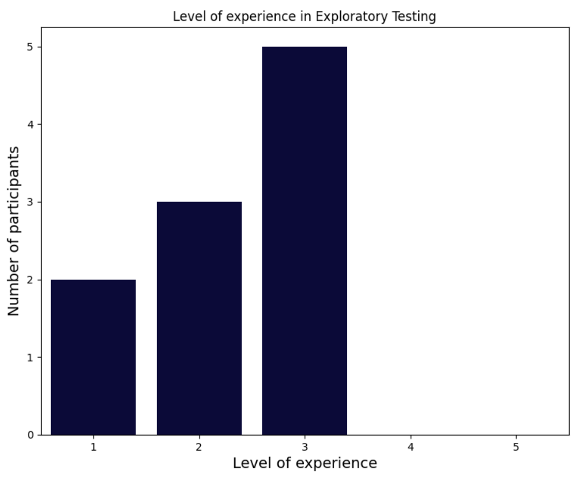
Participant	*Age*	*Gender*	*Educational background*
P1	23	F	Law
P2	25	M	Electrical Engineering
P3	30	F	Graphic Design
P4	32	M	Computer Science
P5	42	M	Biomedical Engineering
P6	31	M	Civil Engineering
P7	27	M	Computer Science
P8	29	M	Civil Engineering
P9	29	M	Digital Games
P10	31	M	Production Engineering

Source: Elaborated by the author.

Figure 30 – Level of experience of the participants in Software testing and exploratory testing.



(a) Experience in Software testing.



(b) Experience in Exploratory testing.

Source: Created by the author.

randomly instead of considering their testing levels. The graphs from Figures 30a and 30b display the overall answers of the participants' proficiency. With respect to EQ3, no participant claimed to be previously involved with `L10n` and `i18n` testing. Hence, all of the respondents were eligible to participate in our study.

5.1.5 Procedures

For the study execution, we prepared 10 smartphones: 5 of them with the training app Vinted and the other 5 with the testing app Moto. This was done to ensure that participants would only interact with one app at a time, either the training or testing app, during the study phases. All smartphones were configured with French as the system language.

Our study was planned to run for a total of 3 hours, following the schedule displayed in Figure 31. The schedule was divided into four main sessions: an opening session, two testing rounds, and a closing session.

In the opening session, the participants are gathered in room 1. The first 30 minutes of the empirical study are allocated to initial training. This initial presentation has the purpose of introducing them to the practice, as the participants have no prior knowledge in `L10n` and `i18n` testing; therefore, the most common failures found during the execution of this type of test need to be presented. The presentation used during this step is documented in the empirical study replication package.

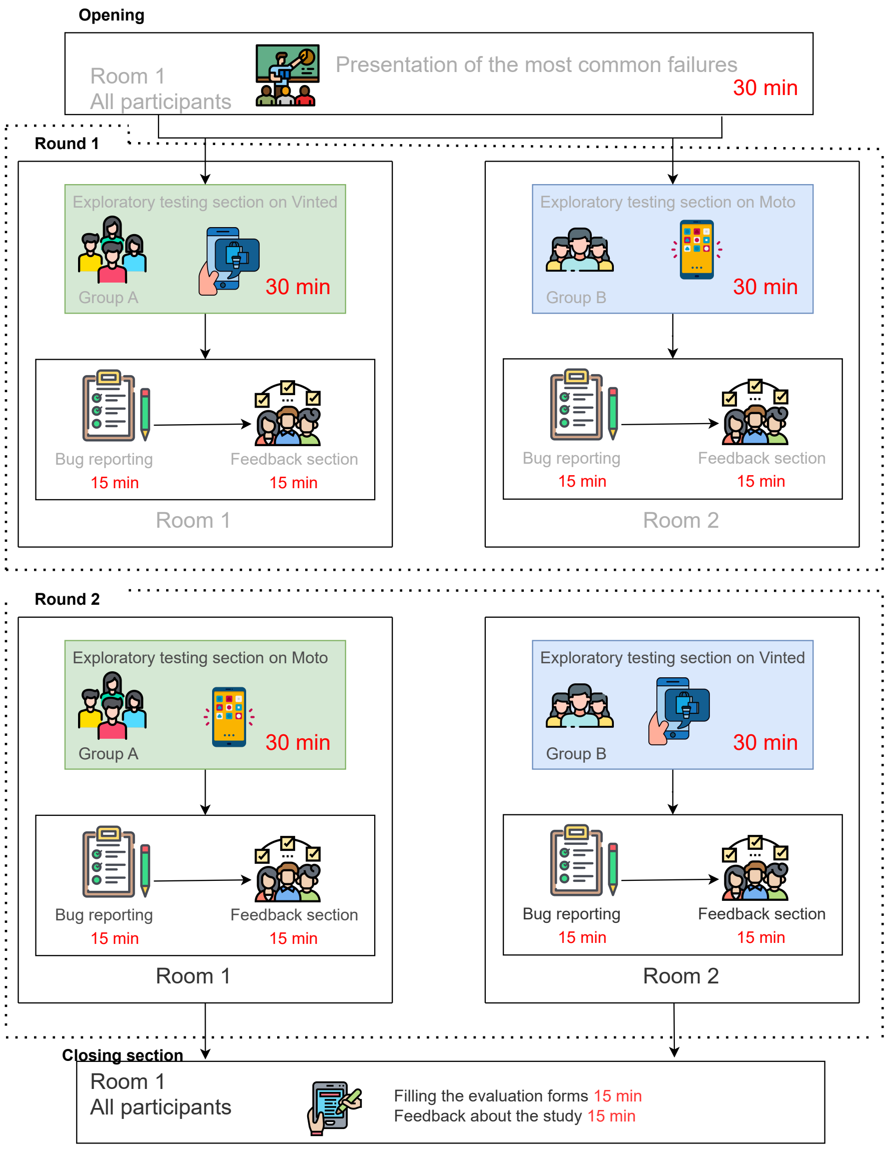
Subsequently, the initial training groups are drawn. Group A stays in room 1, and group B is sent to room 2. The empirical study is guided by two trainers in each room.

Next, in the first testing session (Round 1), the participants from group A are instructed to perform exploratory testing on the training application to find `i18n` and `L10n` failures. For every failure found, they have to take a screenshot of the issue. In total, the time available for this task is thirty minutes. Meanwhile, group B is instructed to start the exploration on Moto with the same objective, and they also have thirty minutes to explore the app. It is important to highlight that exploratory testing is conducted individually by each participant in each group.

After the exploratory testing, the participants have fifteen minutes for bug reporting. To easily identify the bugs found by each participant, they need to log into their email accounts on the smartphone and send a screenshot of each failure found along with a brief description of the issue via chat. If the participants need more time for bug reporting, we would give them additional minutes to ensure no possible true bug found by them is missed. While the participants are reporting the bugs, the author of this dissertation is analyzing the failures reported to classify the true or false bugs found by the participants.

When the bug reporting section ends, we start the feedback section. During this step, we provide feedback about the true positive bugs (the ones that were seeded by us) and the

Figure 31 – Empirical study schedule.



Source: Created by the author.

false positive bugs found in each app (failures that they thought were issues but were not). In this opportunity, the participants have the chance to clarify any doubts they have about the failures.

Following the feedback section for both groups, the participants are instructed to log out of their accounts used for bug reporting and to delete all screenshots from the devices, due to the fact that the groups need to exchange the smartphones.

In the second testing session (Round 2), group A is instructed to perform an exploratory test on the Moto app looking for `i18n` and `L10n` failures, while group B starts the exploration on the training application. Once again, the groups have thirty minutes in total for this task.

The bug reporting of Round 2 starts after the thirty minutes exploratory section. The participants follow the same protocol as in Round 1: logging into their email accounts on the smartphones, reporting the failures in the chat with a brief description of the issue. After the bug reporting from each group, we have another feedback section to discuss the true positive and false positive bugs found.

Finally, in the closing session, we reunite all the participants in room 1 and ask them to answer a form about the empirical study. When they complete the form, we explain the purpose of L10N-TRAINER and the usage of the applications seeded by it. We also invite the participants to provide any feedback about the empirical study and the training application. Furthermore, we relate the elaborated questions and results for this last step in Subsection 5.3.2 of this Chapter.

5.2 EXECUTION

5.2.1 Pilot

Before the official execution we ran a pilot of the empirical study to analyze the level of difficult of it, find holes and flaws in order to adjust those elements for the official execution. For the pilot we invited the two novice testers that were involved with the `L10n` and `i18n` testing team activities since Apr/2023 (3 months of experience). Since the testers had been involved with the practice it wasn't needed to explain the practice and its most common failures. The two testers performed the Pilot following the same steps describe in Table **??**.

Discussing about the performance of the testers, both of the them were able to find only one of the seeded faults on the training application (Figure 26b). An interesting discussion topic is the fact that the testers did not find the bug from Figure 26a, they mentioned to don't know that French language had this rule to use space between the word and the punctuation. This fact highlights that the testers were able to learn something new with the training app during the pilot section. Although being involved with the `L10n` and `i18n` testing activities for three months this was not addressed in their activities during this period.

Regarding the faults on Moto the testers were not able to find any of the faults seeded.

Initially, for the pilot empirical study we had seeded the failure illustrated in Figure 28a and another one that was related to capitalization in an application name. However, during the pilot execution, the participants considered this last one very difficult to find. Thus, with this feedback we decided to replace it for the one from Figure 28b. To achieve the failure from Figure 28a, the tester needed to perform a tutorial and be attentive to the punctuation during the tutorial screens.

Another important consideration regarding the pilot is that we made some adjustments to the training application based on the failure reports provided by the testers during this initial study. This helped us reduce the number of false positives found in the official empirical study.

After finishing the pilot we take a time to discuss with the testers about the purpose of the empirical study and enjoyed the opportunity to ask their opinion about the app, they declared that the app would have been very helpful in their initial training when they first started they activities on the team.

5.2.2 Official execution

In the execution day 9 participants showed up in the scheduled time. The tenth participant (P10) gave up of the empirical study without notify us, due this under-reporting we did not had time to find another participant for replacement. Thus, we decided to follow up the execution with the nine volunteers: 5 participants for group A and 4 for group B. The groups draw and number of participants for each was conducted through a random draw. The schedule of the official execution followed the steps from Table **??**

5.3 RESULTS

In order to evaluate the effectiveness of L10N-TRAINER in training novice `L10n` and `i18n` testers, the number of true bugs (TB), the ones that we seeded on the app, found per group in each app was considered as measure metric. Table 6 contains the number of true bugs found from all the participants by app, for a better visualization of groups, the results from group A was highlighted in blue color. The latest 2 rows of the Table contains the results per group.

Table 6 – Number of bugs found by participants in each group and for each app

Group A			Group B		
Participant	Round 1 (Vinted)	Round 2 (Moto)	Participant	Round 1 (Moto)	Round 2 (Vinted)
P1	1	2	P3	0	1
P2	0	0	P4	1	1
P6	0	1	P5	1	0
P7	0	1	P9	1	1
P8	1	1			
	2	**5**		**3**	**3**

Source: Elaborated by the author.

5.3.1 Results Discussion

Following, we discuss our findings about the empirical study and the key observations from each group based on the results from Table 6.

5.3.1.1 Key observations about group A

(KO1) Key observation #1: Stability or improvement. From Table 6, is possible to note that group A, that was firstly trained with Vinted application, showed an improvement in the number of faults found by the team, increasing from two true bugs found on the training app to five bugs on the Moto application.

(KO2) Key observation #2: Improvement of 150% in the total number of true positive bugs found. From K01 we can estimate that the performance of the group that used the training application as first contact with `L10n` and `i18n` failures was improved in 150% by increasing the number of true bugs found from two to five in total.

5.3.1.2 Key observations about group B

(KO4) Key observation #3: Stability or worsening. In contrast with group A, the second group showed stability in the total number of true bugs found.

(KO5) Key observation #4: No alteration in the total number of true positive bugs found. Additionally to K04, the results from Table 6 shows an stability in the total of bugs found by the group: 3 bugs on Moto and then 3 bugs on Vinted. That indicates that initially training the novices with a more complex application like Moto as the first contact with `L10n` and `i18n` failures might be harder for the tester to assimilate the issues as using a more simpler playful application as the initial contact with the practice.

General observation

(KO5) Key observation #5: No participant found 100% of the fails in the training app. None of the participants were able to find the 2 failures seeded on Vinted application. The only issue found was the one from Figure 26b. The failure from Figure 26a was unnoticed, which is a curious fact, since Brazilian Portuguese is the native language from all participants, in this idiom there is no space between punctuation and word, this French rule should be odd to their eyes.

5.3.2 Qualitative Evaluation

Beyond the quantitative evaluation we also elaborated a form to qualitatively evaluate the use of the training application seeded with L10N-TRAINER, for assessing this, four questions were elaborated. We used the Likert scale with answers varying in a range from 1 to 5, where 1 means strongly disagree and 5 means strongly agree. Following the questions:

1. (Q1) Do you consider that the L10N-trainer app used during the empirical study helped you find faults more easily? If you haven't found any bugs in the Moto app, reflect on whether the faults encountered during the training with the L10n-trainer are similar to the issues found in the real context of localization testing

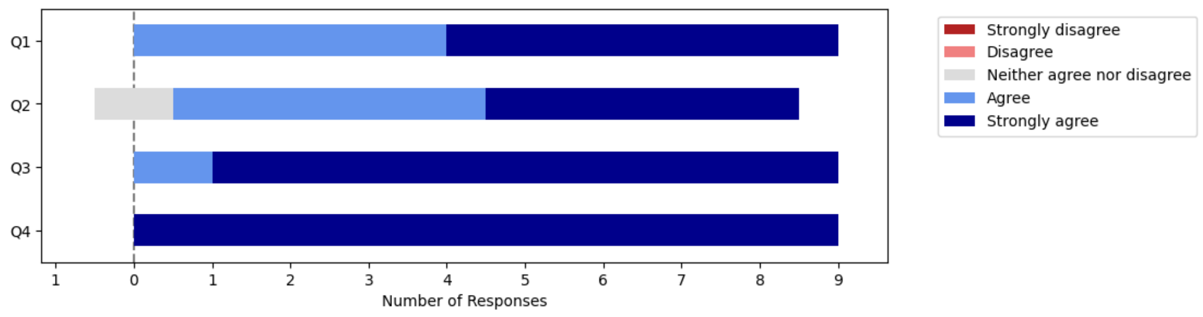
2. (Q2) Do you consider that the bugs found in the L10n-trainer were similar to those encountered during the execution of the exploratory testing in the Moto app? If you haven't found any bugs in the Moto app, reflect on whether the faults encountered during training with the L10n-trainer are similar to the issues found in the real context of localization testing

3. (Q3) Do you consider that the use of the L10N-trainer app along with the feedback received from the trainer helped you better discern actual Localization issues?

4. (Q4) Do you think the L10N-TRAINER tool should continue to be used in training novice `L10n` and `i18n` testers?

Figure 32 shows the Likert scale plot with the answers for all questions. In the first question we wanted to evaluate the feeling of the participants about L10N-TRAINER in their training. Five of the participants answered that they strongly agreed with the affirmation from Q1 and four of them showed an agreement level. Thus, considering the participants' feedback, the use of Vinted app was relevant to their training.

Figure 32 – Results from the qualitative evaluation.



Source: Created by the author.

The second question was elaborated in order to investigate if the similarity of the bugs seeded on the training application was in fact close with the ones from Moto. Most of the participants "strongly agreed" or "agreed" that the failures were similar, only one participant indicated to neither agree or disagree with the similarity between the bugs.

To investigate the effectiveness of the trainer's feedback aligned with the use of the Vinted app during the training, we elaborated the third question. Eight participants answered that strongly agreed with the affirmation showing that the feedback of the trainer during this stage is important to better distinguish the failures. This assertion also reinforces that the trainer can construct a more directed training, making sure that some topics that might be missing during the traditional training are addressed in the training stage. For example, testers from the pilot empirical study affirmed that they did not know about the French punctuation particularities, even after being on the team for 3 months. This topic could be addressed in future training sessions for novice testers using the training app.

Lastly, we sought the participants' opinions on the viability of L10N-TRAINER for the task of training of novice `L10n` and `i18n` testers. Thus, we developed the fourth question. The answer for this question was unanimity, all participants strongly agreed that the tool should be used on the training of future novices `L10n` and `i18n` testers. Which indicates their satisfaction with the training application used in their training.

We also left a space for optional feedback to allow the participants sharing their thoughts about the application and the empirical study in general. The participants were very participating in this field, in total 6 of them provided feedback. P9 wrote: *"I think the training was very comprehensive, it was possible to cover all the points, I can't think of any suggestions. It definitely helped me get a more specific idea of what the Globalization team does."* Discussing about the last statement of P9, as mentioned before in Chapter 2, `L10n` and `i18n` testing is not as popular as other types of testing which is clearly noticed in the participant's words, the practice is sometimes unknown by collaborators within a company itself.

Following, three reviews were related to the number of screens from the training application, P2 declared: *"I believe that developing more screens related to the profile menu options of the app would allow to adding more details to be discovered. Overall, the idea and conception of the app are very good, it fulfilled the purpose of the application."* In addition, P3 wrote: *"I found the app quite nice, actually. Maybe a few more tabs and more errors that were present in this version"*. In the same line of thinking P4 said: *"The app could display more localization errors to help testers identify the most common faults where it might appear"*. All three participants suggested for the training app to have more screens, in future we plan to have more open-source applications with more complex operations and screens to seed with L10N-TRAINER. Discussing about the last statement from P4, in order to make the empirical study more real, we preferred to not introduce too many faults on the application, that would make the empirical study unrealistic, since usually the applications do not have a higher number of `L10n` and `i18n`, unless when they are under initial development stage.

Beyond the suggestions about expanding the application, the remaining two comments included a similar observation about their experience with the application. P8 declared: *"I think the app is very good and intuitive, so the trainer can add various specific faults for the person in training. If it's improved with more available and interactive screens, it will become a very useful and facilitating tool in training. My biggest difficulty was that I didn't have much knowledge in the field of globalization, so I encountered possible issues but wasn't sure if it was actually an issue or not"*. In this line of thinking of the last phrase of P8, P5 commented: *"I believe that the implementation of a virtual environment for sharing the results found by each participant would be very valuable for the purpose of self-assessment and individual self-correction."* Both participants expressed challenges in confidently distinguishing true bugs from false positives and desired a form of self-correction feedback.

At last, in the end of the empirical study, we chatted with the participants about their initial training process experience in the industrial settings and the possibility of using this same approach on other testing teams, not only for `i18n` and `L10n` testing team. They showed excitement with the idea and related that in some testing teams they face the same problem in being trained with stable applications and concluding the training phase without observing any failure related to their testing area.

5.3.3 Threats to Validity

Following the assertions on how valid the results are and whether we can generalize them to a larger population are discussed.

Internal validity: With a small number of participants, the results may not be generalized to a larger population. The observed effects could be due to random variations rather than representing a real trend. We need to perform the empirical study with more participants to make our results even more reliable. Besides that, the fact that the group A had more participants than group B could influence on the overall results of the empirical study, even considering the fact that the participant P2 from group A had favorable results to the group. In order to avoid any bias related to that, the number of participants of the groups and their members were chosen by random draw. Another important threat is the individual characteristics of the participants, this could influence in their performance, giving more advantages for a team over another. To avoid that, in a new experiment we could balance the levels of the groups based in the participants of each.

External validity: Since our approach can be used not only for mobile applications, but as well for system and web apps. The replication of this study is not limited to the industrial environment were this empirical study was carried out, the study can be adapted to others software and devices.

Conclusion validity: A common conclusion validity threat is the violation of the assumptions of the statistical methods used. In our empirical study, due time constraints and size of apps restrictions, we used only 2 bugs, by finding one of then the participant had already found half of the bugs seeded. Although the observed significant results in our study are in line with our expectations, an empirical study with more complex applications and more days available for the empirical study should be conducted to minimize this thread.

5.4 CONCLUDING REMARKS

In this Chapter we described the planning, execution and results of our evaluation to assess the effectiveness of L10N-TRAINER for the task of training novice L10n and i18n testers. We designed a study with two groups: one used a training application seeded with L10N-TRAINER and then executed exploratory tests in an official application developed by the company partner. The other group did the opposite, by starting the training on the traditional official app and then exploring the training application. The group that was first exposed to the app produced by L10N-TRAINER improved their performance by 150% in the official app when considering the number of true bugs found. Besides that, the only participant that was able to find 100% of the true bugs in the official app was a participant that was first exposed to L10N-TRAINER.

6 CONCLUSION AND FUTURE WORK

Our research contributes to the field of software testing by bridging a gap between theory and practice in L10n and i18n testing education. L10N-TRAINER, the tool developed in this work, offers a practical solution for addressing the challenges faced in training novice testers in L10n/i18n. In addition, the SMS conducted and reported in Chapter 3 contributes with the testing community by being a guideline of the main challenges faced by L10n and i18n testers, highlighting opportunities for further research.

The development and continuous improvement of L10N-TRAINER is an important step forward in assisting the training of L10n and i18n testers, ultimately contributing to the quality of globalized software. By providing hands-on experience on real apps with carefully seeded L10n/i18n faults, L10N-TRAINER helps testers to effectively identify potential faults related to language and cultural conventions, thus enhancing their overall skills and knowledge in the domain of L10n and i18n testing.

6.1 MAIN FINDINGS

From the SMS reported in this work we could identify the main challenges faced by L10n and i18n testers: lack of support material, lack of tooling support, testers training & qualification and unbalanced support for different locales. Aligned with these challenges, we identified also some opportunities: development of tools for supporting the practice, producing support material and exploration of the use of new technologies, such as Large Language Models (LLMs) in the context of L10n. Particularity, our work embraces these opportunities by developing a tool for training novice L10n and i18n testers and by delivering the SMS itself.

The empirical evaluation of L10N-TRAINER showed that the use of our tool improved the performance of the group of participants that was first introduced to L10n and i18n testing with an application seeded by the tool. The group that was first introduced to the practice using the traditional methodology used on the company demonstrated stability or worsening in the results, whilst the group training first with the training application improved their number of true bugs found, from 2 to 5 true bugs. In addition, the participants showed interest in the adoption of a similar approach in the training for their testing teams. The adoption of our approach in the training of novice testers, not only for L10n and i18n testing, but for other teams (by seeding faults related to their practices) could enhance the learning experience of the novice tester and consequently professionals skills.

6.2 FUTURE WORK

Moving forward, our future work will focus on expanding L10N-TRAINER capabilities, incorporating additional dummy applications, and refining its user interface. Additionally, broader evaluation of our tool can provide valuable insights into its real-world effectiveness and potential for widespread use. In particular, we want to conduct an evaluation from the perspective of the trainers to assess their experience using our tool.

In our efforts to expand our approach we plan to do the "gamification" of the training app, to make it more didactic. We plan that by clicking on the spot of a possible i18n or L10n failure on the screen, the application instantly provides a feedback about the issue informing if it is a true or false bug, also providing information about it.

We also plan to continue contributing with the L10n and i18n testing community by researching and developing solutions for the challenges not addressed in this work, in particular the development of tools to aid the execution of automated L10n and i18n testing as proposed in (Couto and Miranda 2022; Felipe and Miranda 2023).

REFERENCES

ABUFARDEH, S.; MAGEL, K. Software localization: the challenging aspects of arabic to the localization process (arabization). *IASTED Proceeding of the Software Engineering SE*, p. 275–279, 2008.

ALAMEER, A.; CHIOU, P. T.; HALFOND, W. G. Efficiently repairing internationalization presentation failures by solving layout constraints. In: IEEE. *2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)*. [S.l.], 2019. p. 172–182.

ALAMEER, A.; MAHAJAN, S.; HALFOND, W. G. Detecting and localizing internationalization presentation failures in web applications. In: IEEE. *2016 IEEE International Conference on Software Testing, Verification and Validation (ICST)*. [S.l.], 2016. p. 202–212.

ALI, Z. Behavior-driven development as an error-reduction practice for mobile application testing. *International Journal of Computer Science Issues (IJCSI)*, International Journal of Computer Science Issues (IJCSI), v. 16, n. 2, p. 1–10, 2019.

AMNA, A. R.; POELS, G. Ambiguity in user stories: A systematic literature review. *Information and Software Technology*, v. 145, p. 106824, 2022. ISSN 0950-5849.

AMPATZOGLOU, A.; BIBI, S.; AVGERIOU, P.; VERBEEK, M.; CHATZIGEORGIOU, A. Identifying, categorizing and mitigating threats to validity in software engineering secondary studies. v. 106, p. 201–230, 2019. ISSN 0950-5849.

AWWAD, A. M. A.; SLANY, W. Automated bidirectional languages localization testing for android apps with rich gui. *Mobile Information Systems*, 2016.

BADAMPUDI, D.; WOHLIN, C.; PETERSEN, K. Experiences from using snowballing and database searches in systematic literature studies. In: *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2015. (EASE '15). ISBN 9781450333504.

CAREY, J. M. Creating global software: a conspectus and review. *Interacting with Computers*, Oxford University Press Oxford, UK, v. 9, n. 4, p. 449–465, 1998.

CONFLUENCE. *Non-ASCII Characters display as square characters in Confluence*. Https://confluence.atlassian.com/confkb/non-ascii-characters-display-as-square-characters-in-confluence-317948797.html. 2023-04-14.

COUTO, M.; MIRANDA, B. Towards improving automation support for internationalization and localization testing. In: *Anais Estendidos do XXI Simpósio Brasileiro de Qualidade de Software*. Porto Alegre, RS, Brasil: SBC, 2022. p. 9–14. ISSN 0000-0000. Available at: <https://doi.org/10.5753/sbqs_estendido.2022.227653>.

COUTO, M.; MIRANDA, B. An industrial experience report on the challenges in training localization and internationalization testers. In: *Proceedings of the 8th Brazilian Symposium on Systematic and Automated Software Testing*. New York, NY, USA: Association for Computing Machinery, 2023. (SAST '23), p. 96–98. ISBN 9798400716294. Available at: <https://doi.org/10.1145/3624032.3624045>.

COUTO, M.; MIRANDA, B. l10n-trainer: a tool to assist in the training of localization (l10n) and internationalization (i18n) testers. In: *Proceedings of the XXXVII Brazilian Symposium on Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2023. (SBES '23). ISBN 979-8-4007-0787-2/23/09. Available at: <https://doi.org/10.1145/3613372.3613420>.

DESIGN, M. *Bidirectionality*. Available at: <https://m2.material.io/design/usability/bidirectionality.html>.

EL-KADI, A.; BADREDDIN, O. B. grup–a globalized approach to software engineering. *Journal of Computational Methods in Sciences and Engineering*, IOS Press, v. 9, n. s2, p. S201–S210, 2009.

ESCOBAR-VELÁSQUEZ, C.; OSORIO-RIAÑO, M.; DOMINGUEZ-OSORIO, J.; AREVALO, M.; LINARES-VÁSQUEZ, M. An empirical study of i18n collateral changes and bugs in guis of android apps. In: IEEE. *2020 IEEE international conference on software maintenance and evolution (ICSME)*. [S.l.], 2020. p. 581–592.

EXPO Go. <https://expo.dev/client>. Accessed: 2023-04-30.

FELIPE, L.; MIRANDA, B. Supporting localization testing through automated application navigation. In: *Anais Estendidos do XXII Simpósio Brasileiro de Qualidade de Software*. Brasília, DF, Brasil: SBC, 2023. p. –. ISSN 0000-0000.

GARCIA, L. A.; OLIVEIRAJR, E.; MORANDINI, M. Tailoring the scrum framework for software development: Literature mapping and feature-based support. *Information and Software Technology*, v. 146, p. 106814, 2022. ISSN 0950-5849.

GEOGRAPHIC, N. *Globalization*. Available at: <Globalization>.

GITHUB. *Cosmetic bug: Should we add more space between options in What were affected page?* Https://github.com/cds-snc/report-a-cybercrime/issues/2288. 2023-04-14.

HOGAN, J. M.; HO-STUART, C.; PHAM, B. Key challenges in software internationalisation. In: *Proceedings of ACSW Frontiers '04, Volume 32*. [S.l.: s.n.], 2004. p. 187–194.

IBM. *IMB Guidelines to Design Global Solutions*. Available at: <http://www-01.ibm.com/software/globalization/guidelines/a3.html>.

JESUS, C. M. d. *Provérbios*. [S.l.: s.n.], 1963.

JIA, Y.; HARMAN, M. An analysis and survey of the development of mutation testing. *IEEE transactions on software engineering*, IEEE, v. 37, n. 5, p. 649–678, 2010.

KITCHENHAM, B.; CHARTERS, S. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. [S.l.], 2007.

KRUKOWSKI, I. *Internationalization vs. localization (i18n vs l10n): What's the difference?* Available at: <https://lokalise.com/blog/internationalization-vs-localization/>.

LEWOWSKI, T.; MADEYSKI, L. How far are we from reproducible research on code smell detection? a systematic literature review. *Information and Software Technology*, v. 144, p. 106783, 2022. ISSN 0950-5849.

LINGUAE, S. *Localization QA Testers: Guardians of Quality*. Https://summalinguae.com/localization/localization-qa-testers/. 2021-10-26.

LISA. The localization industry primer. n. 2, 2003. ISSN 1420-3693. Available at: <https://www.immagic.com/eLibrary/ARCHIVES/GENERAL/LISA/L030625P.pdf>.

MAHAJAN, S.; ALAMEER, A.; MCMINN, P.; HALFOND, W. G. Effective automated repair of internationalization presentation failures in web applications using style similarity clustering and search-based techniques. *Software Testing, Verification and Reliability*, Wiley Online Library, v. 31, n. 1-2, p. e1746, 2021.

MARTINEZ, M.; ESPARCIA, A. I.; RUEDA, U.; VOS, T. E.; ORTEGA, C. Automated localisation testing in industry with test^*. In: SPRINGER. *Testing Software and Systems: 28th IFIP WG 6.1 International Conference, ICTSS 2016, Graz, Austria, October 17-19, 2016, Proceedings 28*. [S.l.], 2016. p. 241–248.

MASUVERN. *barinsta*. Https://github.com/masuvern/barinsta. 2021-07-13.

MERRIAM, S. B.; TISDELL, E. J. *Qualitative research: A guide to design and implementation*. [S.l.]: John Wiley & Sons, 2015.

MICROSOFT. *Globalization testing approach*. Available at: <https://learn.microsoft.com/en-us/globalization/testing/testing-approach>.

MICROSOFT. *Testing localizability*. Available at: <https://learn.microsoft.com/en-us/globalization/testing/localizability-testing>.

MOURãO, E.; KALINOWSKI, M.; MURTA, L.; MENDES, E.; WOHLIN, C. Investigating the use of a hybrid search strategy for systematic reviews. In: *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. [S.l.: s.n.], 2017. p. 193–198.

MOURãO, E.; PIMENTEL, J. F.; MURTA, L.; KALINOWSKI, M.; MENDES, E.; WOHLIN, C. On the performance of hybrid search strategies for systematic literature reviews in software engineering. *Information and Software Technology*, v. 123, p. 106294, 2020. ISSN 0950-5849.

MULERO, V. M.; ADELL, P. P.; BONET, C. E.; VILLODRE, L. M. Context-aware machine translation for software localization. In: *Proceedings of EAMT 2012: Trento, Italy, 2012*. [S.l.: s.n.], 2012. p. 77–80.

PACKEVIČIUS, Š.; RUDŽIONIENĖ, G.; BAREIŠA, E. Automated localization testing of mobile applications method. *International Journal of Software Engineering and Knowledge Engineering*, World Scientific, v. 32, n. 05, p. 769–790, 2022.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic mapping studies in software engineering. *12th International Conference on Evaluation and Assessment in Software Engineering*, v. 17, 06 2008.

PING, T. P.; CHAN, C. P.; SHARBINI, H.; JULAIHI, A. A. Integration of cultural dimensions into software localisation testing of assistive technology for deaf children. In: IEEE. *2011 Malaysian Conference in Software Engineering*. [S.l.], 2011. p. 136–140.

PIXAR. *A Peek Inside Pixar's Localization Process For 'Monsters University'*. Available at: <https://pixarpost.com/2015/01/laura-meyer-localization.html>.

RAMLER, R.; HOSCHEK, R. How to test in sixteen languages? automation support for localization testing. In: IEEE. *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*. [S.l.], 2017. p. 542–543.

RAMLER, R.; HOSCHEK, R. Process and tool support for internationalization and localization testing in software product development. In: SPRINGER. *Product-Focused Software Process Improvement: 18th International Conference, PROFES 2017, Innsbruck, Austria, November 29–December 1, 2017, Proceedings 18*. [S.l.], 2017. p. 385–393.

SANTOS, R. E.; CORDEIRO, J. R.; LABICHE, Y.; MAGALHÃES, C. V.; SILVA, F. Q. da. Bug! falha! bachi! fallo! défaut! ! what about internationalization testing in the software industry? In: *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. [S.l.: s.n.], 2020. p. 1–6.

SCHINDLER, C.; LUHANA, K. K.; SLANY, W. Towards continuous deployment of a multilingual mobile app. *International Journal*, v. 9, n. 7, 2021.

SURVEY: Consumers prefer their Own Language. Https://csa-research.com/Blogs-Events/CSA-in-the-Media/Press-Releases/Consumers-Prefer-their-Own-Language. 2023-04-14.

TARASIEWICZ, J. *How Much Does Software Localization Cost?* Available at: <https://www.atltranslate.com/articles/how-much-does-software-localization-cost>.

THE 12 most spoken languages in the world. Https://blog.busuu.com/most-spoken-languages-in-the-world. 2023-04-14.

VAROUQA, M.; HAMMO, B. Wit: Weka interface translator. *International Journal of Speech Technology*, Springer, v. 19, p. 359–371, 2016.

VIERA, A. J.; GARRETT, J. M. Understanding interobserver agreement: The kappa statistic. *J. of the Society of Teachers of Family Medicine*, v. 37, n. 5, 2005.

VINTED. <https://github.com/vinted>. Accessed: 2023-04-30.

XIA, X.; LO, D.; ZHU, F.; WANG, X.; ZHOU, B. Software internationalization and localization: An industrial experience. In: IEEE. *2013 18th International Conference on Engineering of Complex Computer Systems*. [S.l.], 2013. p. 222–231.

YNION, J. C. *Using AI in Automated UI Localization Testing of a Mobile App*. Master's Thesis (Master's Thesis) — Metropolia University of Applied Sciences, 2020.

ZOU, Q.; LIU, G. Chinese localisation of evergreen: an open source integrated library system. *Program*, Emerald Group Publishing Limited, v. 43, n. 1, p. 49–61, 2009.