Domain Analysis and Geographic Context of Historical Soundscapes: The case of Évora

Mariana Bonito
m.bonito@campus.fct.unl.pt
Advisors: João Araújo, Armanda Rodrigues
Faculdade de Ciências e Tecnologia – Universidade NOVA de Lisboa

Abstract. *Interactive and multimedia-based historical soundscape environments with geolocation* is a relatively unexplored area. But recently, this topic has started to call attention of researchers due to its relevance in culture and history. Currently, there is an ongoing project (Patrimonialization of Évora's Soundscape) between NOVA LINCS, Universidade de Évora, and other associations to develop a platform that provides access to historical soundscapes in specific locations. This work presents an initial domain requirements analysis. Tool support is provided by a developed FeatureIDE plug-in extension which allows URL associations to individual features in a feature model. Feature models are a key concept of domain analysis. The importance of historical soundscapes is clarified in this paper. It also explores in-depth several different approaches to domain analysis and the selected approach is justified. Finally, both contributions are evaluated and analyzed.

Keywords: Domain Analysis, Soundscapes, Historical Soundscapes, FeatureIDE, Feature models

1 Introduction

Soundscape is the technical term used to describe the sound in our surroundings. This "environment" can be created in many ways including music, historical recordings, or even the product of reminiscing and imagining the sound [7]. As Emily Thompson states, "soundscape is simultaneously a physical environment and a way of perceiving that environment" [19]. Hence, having access to historical soundscapes allows for a better understanding of life in the past. In addition, it acts as an indicator of the evolution of a community [7].

Given the importance of soundscapes, but more specifically, historical soundscapes, there have been many attempts by different global organizations such as the "Early Modern Soundscapes" [23] or the "Simon Fraser University" [21] to explore and offer to the public, information and access to soundscapes of the past. Recently, the EU, along with the Spanish and the Portuguese governments have taken more of an interest in providing a platform for people interested in certain historical cities. The aim is to give access to a repertory of sounds they might have heard in the past in a specific location and thus, build a historical soundscape allowing for a more immersive experience [15,16].

This work is part of the PASEV project which is developing a platform designed to share interactively the historical soundscapes of Évora with the user. It includes a prototype[1] which was the result of the thesis and work of João Rosário [17]. This investigation presents an initial domain requirements analysis for the *interactive and multimedia-based historical soundscape environment with geolocation* domain. Thus, projects in the domain, such as the PASEV project, can be part, and take advantage of the benefits, which is the reuse of information, explored for example, by Software Product Line Engineering (SPLE) [20]. The purpose of domain analysis is to establish guidelines that allow for the reuse of common components amongst projects in the same domain. It presents "an economical and reduced time-to-develop approach for providing a general model for developing similar products" [11]. There are some suggested methods on how to approach such a problem (detailed in Section 2.2). However, most domain analysts do not use them in practice as "they are too challenging and rigid to make", and "they target assets that do not have high reuse potential" [25]. Furthermore, the explored methods attempt at modeling the domain architecture which is outside the project's scope. Hence, some changes to the analyzed methods were required (further explored in Section 3.1). The challenge of shifting from software engineering to domain engineering, in this case, arises from the lack of information and developed projects within the domain as it is relatively an unexplored area.

Besides the domain model previously referred, this project also contributes with an extension of the open-source FeatureIDE plug-in for Eclipse IDE, which allows specific features in a feature model to be linked to URLs. A more detailed explanation of the plug-in, along with its extension, and the extension's purpose, is presented further in this paper.

The research problem has been explained in this section. The remaining of the paper is structured as follows. Section 2, details the state-of-the-art relevant for this project. Then, in Section 3, the selected approach for the domain analysis is detailed and the plug-in extension is explained. In Section 4 the selected content evaluation is clarified along with the results of the evaluation. Finally, in Section 5 we draw some conclusions and point out directions for future work.

2 Background and state-of-the-art

This section aims at giving the reader context on the areas relevant to this investigation.

2.1 Current projects within the domain

Associating historical soundscapes to particular geolocations is a relatively unexplored area. Hence, (to the best of the author's knowledge) there are only two projects developed within the proposed domain of *interactive and multimedia-based historical soundscape environment with geolocation*. These projects are the

[1] Available at https://pasev.di.fct.unl.pt/#brand [16].

PASEV project and the Spanish historical soundscapes project [15,16,18]. This section briefly describes these two projects as a result of the author's observations. A more in-depth analysis is provided in the separate domain analysis document under Section *4 Software Analysis* [3].

PASEV project - This project aims at associating multimedia to geolocations on a map in order to create an immersive soundscape experience for the user. It does so by having two separate platforms: one for information on the project (available at https://pasev.hcommons.org/); and one with the actual information associated with the geolocations (currently the prototype is available at https://pasev.di.fct.unl.pt/). The prototype consists of having a map with selectable locations that display information accordingly, and many filters including a timeline to filter periods and search tools to filter events and locations.

Spanish historical soundscapes project - The Spanish project has a similar purpose of associating historical soundscapes to specific geolocations. However, their display is very different. They started by having individual maps and tabs for different cities (Granada and Seville), but they moved on to incorporating "Interconnected cities" [14]. The "Interconnected cities" component is still being developed, thus, its analysis is very limited. Their project is available at http://www.historicalsoundscapes.com/. It consists of different tabs with different information about the project, maps and the historical soundscape information associated to specific locations, and finally, itineraries with their corresponding information relative to Granada and Seville.

Comparing both projects - The main difference between the PASEV and the Spanish historical soundscape project, apart from the layout, consists of the PASEV project associating multi-media to particular locations, whereas the Spanish historical soundscapes project associates the multimedia to events. In addition, the display of information leads to the belief that the target audience varies slightly. The PASEV project is targeted towards tourists visiting Évora with a mild interest in the historical soundscape of the city, whereas the Spanish historical soundscapes project seems to target people not necessarily on that location at the time, with greater interest, and willing to dedicate more time to understanding the historical soundscapes of a particular place. In addition, there is no back-office platform visible for the Spanish project, however, the PASEV project, provides one, so members of the responsible organization can add data to the system.

2.2 Approaches to domain analysis

This section describes briefly three of the most popular approaches for domain analysis. All methods agree that the domain analysis must create an output of a domain model. This domain model is intended to represent the commonalities (requirements common to the whole domain) and the variabilities (requirements the vary between systems in the same domain).

Feature Oriented Domain Analysis (FODA) - This method focuses on domain features. According to Kang et al. (1990), a feature is "a prominent or distinctive user-visible aspect, quality or characteristic of a software system or systems". The domain analysis in this process is divided into 3 phases: context analysis (determines the scope of the domain), domain modeling (clarifies requirements, features, and entities), and architectural modeling (defines the architecture of systems in a domain). The domain model, in this case, consists of a context model, a feature model, and an entity-relationship model. Furthermore, it would not include the architectural modeling as the focus is on domain requirements. Despite the completion criteria not being evident, it can be assumed that the analysis is complete once the domain model is finished.

The FODA method creates a hierarchy of abstraction from the most general features to the very specific. The variabilities are incorporated into the refinements of the abstractions but are postponed as much as possible [10]. This led to critics such as Frakes et al. (1998) to mention this method often incorrectly focuses "on the analysis of low-level domain objects".

Throughout the explored methods, there will be a clear emphasis on the importance of determining the required features. One of the best ways to represent said features and their level of abstraction (commonalities vs variabilities) is through a feature diagram (figure 1). A feature diagram can also be called a feature model that is present in the domain model of this method. This reiterates how this diagram plays a major role in domain analysis.

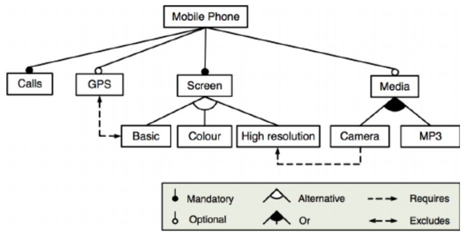

Fig. 1. Self-explanatory example of a feature diagram[1]

Domain Analysis and Reuse Environment (DARE) - In this procedure, the domain model output is referred to as the domain book. Its content is structured in 3 parts[2]: Domain Sources (all domain information available, well structured and organized); Vocabulary analysis (terms, concepts and phrases relevant to the domain, well organized); Architectural analysis (description of the architecture, the features and the code structure for projects in the domain). This method permits a thorough analysis of the domain giving it a clear structure

[2] Features also play an important role in this method as Part 1 and Part 3 contain chapters 'System Feature Table' and 'Generic Features'.

and completion criteria. The method is complete when all sections of the 'book' are thoroughly filled. For this project, only the small part of the "Architecture Analysis" about the domain features is relevant.

Another point that makes this method unique is the intense focus on vocabulary analysis. The DARE creators developed a tool which, apart from other things, clusters automatically related words and phrases[6].

This method also emphasizes the importance of a domain expert stating that they "are the most important source of information for domain analysis" due to their knowledge of the architecture of each system in the domain [6].

Product-Line Software Engineering- Customizable Domain Analysis (PuLSE- CDA) - This method begins by analyzing the topics covered by the domain (scope definition) and redefining the boundary. "The scope definition provides an initial structure for the domain information and the boundary definition limits the area to be analyzed" [2]. Then, raw/unstructured domain knowledge is gathered from various sources and "captured in the domain model work products" [2]. This establishes links between the information and the source, allowing for evolution and maintenance.

"PuLSE-CDA expects that the domain model will never be stable" [2]. Hence, it focuses on "product line infrastructure evolution". This is achieved through a constant step to re-analyze the requirements and domain knowledge. By having the link between the source and the raw information, if the source alters its data, the update on the model domain knowledge (next step) will be much more efficient. However, as this is constantly evolving, there are no completion criteria.

The third and final step is where the unstructured information from the previous stage is modeled into the two outputs: the domain model and a domain decision model. In this case, the content of the domain model is not specified. However, Bayer et al. (2000) suggest the domain model should contain meta elements. A meta element is a notation for generic work products that illustrate the variation points and allow instantiation of the work products [2].[3]

The domain decision model allows the people tasked with implementing or updating a system in the domain, to pick the right variabilities. "The variabilities are connected to decisions that, when completely resolved, specify (via instantiating the domain model) a particular system" [2]. Apart from the links and constant updates of the domain model, the decision model could be incorporated to this project by providing a guideline to help the users select between the variabilities.

2.3 FeatureIDE plug-in for Eclipse IDE

The FeatureIDE plug-in is a tool that allows its users to create feature models similar to the one present in figure 1. It was used when developing the domain model to create both feature models necessary. It has other functionalities that

[3] This concept is very similar to that which is represented in feature diagrams.

include implementing features, but this is not relevant to the problem at hand. The plug-in's official web page can be found at http://www.featureide.com/ and the user manual at https://github.com/FeatureIDE/FeatureIDE/wiki/Tutorial.

The plug-in allows the user to add features (above, bellow, or beside other features) to its model. It also allows the user to pick whether the features are optional or mandatory and pick the relationship between siblings (alternative, or). The constraints between features are represented with logical operators besides the diagram. There is also the ability to add a description to each feature through a dialog box that is then displayed as a tool-tip of the feature. The model is stored in an XML file. Hence, all properties need to have well-defined tags for the editor viewer to interpret the content (detailed in Section 3.2).

3 Selected approaches and contributions

The information in Section 2 is used as a basis and context for this section. This section explains and justifies the selected approach for both the domain analysis and the FeatureIDE plug-in extension as well as describing briefly their content.

3.1 Domain analysis

In this section, the selected approach, which is a combination of the ones analyzed in Section 2.2, is presented and justified. Then the content of the domain analysis developed is very briefly explained.

Approach - When selecting the approach, quite a few domain analysis papers were analyzed and it was clear that none of the traditional methods explained in Section 2.2 are fully followed in practice. Firstly, it is important to emphasize how this initial domain analysis focuses on domain requirements rather than architectural modeling. The main abstract steps taken in domain analysis are present in figure 2 and are self-explanatory. The only step that was not taken in this case, is the "Business decisions" as the implementation is not part of this investigation. The "Evaluation by developers" is present in Section 4. By time *b* a specific domain model has been created based on a combination of the methods presented in Section 2.2. This combination is explained bellow.

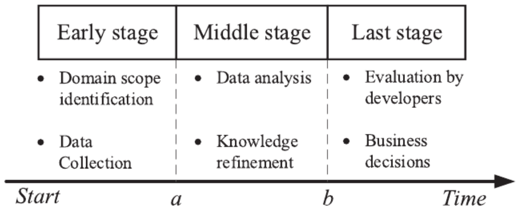

Fig. 2. Abstract stages of domain analysis [24]

The PuLSE-CDA inspired the idea of linking sources to requirements, as this will allow for a more efficient evolution of the domain model. Most of the layout is based on the Maarit Harsu survey [8]. The book metaphor and the completion criteria (when the whole outline has been filled) is based on the DARE approach. Finally, part of the FODA method phases is also included as they provide useful products for this problem.

The feature diagram plays a major role in all analyzed methods. However, the justification for choosing to develop a class diagram and a use case diagram may not be as evident. The need for a class diagram comes from the need to have some sort of representation of the entities required by the domain and their relationships with each other. It was preferred to a regular Entity-Relationship (ER) diagram as it allows for a more specific representation of the associations between classes as well as the required attributes, making the implementation of future software much simpler. The use case diagram is used in order to represent the domain requirements. As Khalique et al. (2017) emphasize, "each use case composes one or many functional requirements" and despite not explicitly representing the non-functional requirements, these either "define the product as a whole" [11] (the product is a project belonging to the proposed domain), or are viewed as "quality attributes to functional requirements and as enhancements; for use cases of the product" [11].

Hence, the method that was applied consists of creating a proposal of a domain book of sorts. Yet, as the focus is solely on the domain requirements and has no need for architectural modeling, it only contains the following:

Table 1. Domain model outline for this project

Domain scope	Detailed limits of domain scope and target users for systems in the domain
Definitions	Dictionary of technical terms and concepts the reader may not be familiar with
Software Analysis Commonalities	Structured list of assumptions that are true for all systems in the domain
Software Analysis Variabilities	Structured list of assumptions on how systems in the domain differ
Domain Requirements	Structured list of mandatory and optional, functional and non-functional requirements for all systems in the domain that already exist or may come to be developed. This will have a link to the source. It will also be supported by a use-case diagram.
Domain modeling	Class diagram and Feature diagram
Issues	A record of important decisions and their alternatives

Summary of domain analysis - When developing a domain model it is important, as Rabiser et al. mention, to keep in mind "it cannot be guaranteed that all possible requirements are covered". Hence, there is a need to scope the domain and properly define its limit. Rabiser et al. choose to implement a bottom-up approach in order to develop their domain analysis. They do so by creating an initial domain model and then comparing their domain model to a large sample of software in the domain. They base their initial model on the authors' prior knowledge and then fill in the gaps of the missing components by analyzing the rest of the software. This allows them to add components that may not necessarily be present in any of the software whilst still incorporating the requirements already present in the domain. Unfortunately, this approach was inapplicable for this investigation as the author had no prior knowledge on the domain and all the knowledge was obtained through the analysis of the available documentation on the two projects belonging to the domain and discussion with the project advisors. Nevertheless, having no prior knowledge on the domain also helped the author identify components not necessarily implemented on the current projects as there was no bias to incline towards previous works. This allowed the author to have a more critical analysis of the domain and, along with the two project advisors, come up with new ideas that could be desirable for future development.

An important factor taken under consideration when developing the domain model, was the target users. Usually, domain models are targeted at domain and system developers [10]. However, in this case, as the focus is on domain requirements, there is a need for the *clients* requesting the software and some domain experts to understand the document, to help select the requirements for the system being developed in the proposed domain. Hence, the content must be accessible for readers without experience in domain modeling. This led to the choice of representing the requirements and their sources in the form of a table rather than other options, such as goal graphs [9,12]. This is because, despite allowing a more indepth and detailed representation, it would be much more difficult for the readers to understand without prior knowledge and practice. In addition, as readers are not required to have prior knowledge on how to interpret use-case diagrams, class diagrams, and feature models, a brief explanation of their interpretation is available in the appendix. Furthermore, readers are not necessarily domain experts. This led to the necessity of Section *3 Domain Concepts* where terms and concepts relevant to the domain under analysis, the user may not be familiar with, are explained.

In brief, the domain model is a result of the analysis of both systems mentioned in Section 2.1. It also covers the mandatory and optional requirements for systems in the domain. As previously mentioned, some of these requirements were not present in either of the analyzed systems but were the product of analysis of the domain and deliberation with the project advisors. The class diagram was the part where most discussions arose as there are many ways to structure the entities and their relationships. However, all decisions for this diagram and any other conflict that was encountered during the domain analysis, are justified in Section *7 Domain Model Issues*.

Four of the main mandatory functional requirements are present in table 3. It is included so the reader can understand how the link between a requirement and its source is shown. The table has three columns: the name, a brief description of the requirement, and its source.

One of the diagrams developed for the domain model is presented in figure 3. This model presents the feature model developed for the back-office platform. Its interpretation is facilitated by the legend present in the figure. There is also another feature model present in the domain model for the front office platform.

Table 3. Domain mandatory functional requirements

Requirement	Description	Source
Temporal filter mechanism	This filter mechanism limits the data displayed on the map by showing only the data relevant to a selected period. In the PASEV example, this is achieved through an interactive timeline, whereas the Spanish historical soundscapes project achieves this by allowing the user to select the beginning and the end of the period through two drop-downs.	Analysis of the PASEV prototype [16] and the Spanish historical soundscapes project [18], along with Jõao Rosário's thesis [17].
Map displaying information	The domain requires associating multimedia data to geolocations to create a historical soundscape environment. Hence, this data is to be displayed in an interactive map with pins and polygons in the corresponding geolocations.	Analysis of the PASEV prototype [16] and the Spanish historical soundscapes project [18], along with Jõao Rosário's thesis [17].
Display of multimedia information	The developed software must be able to display videos, images, and sound recordings in order to create a soundscape environment.	Analysis of the PASEV prototype [16] and the Spanish historical soundscapes project [18], along with Jõao Rosário's thesis [17].
Back-office platform	The back-office platform should be a simple interface which allows the specialist administrator user to add relevant information and content to the system.	Jõao Rosário's thesis [17].

The feature models were developed using the FeatureIDE plug-in. This feature model is included so the reader can understand how the features in the domain model were presented.

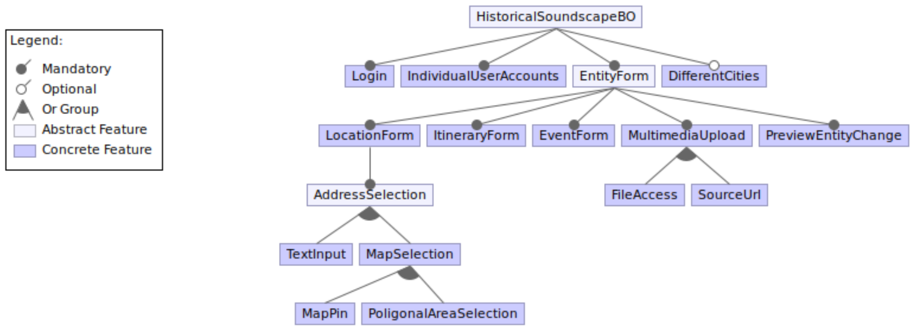

Fig. 3. Back-office feature model of the domain model

The domain model is available at: https://github.com/marianaBonito/APDC-Investigation-HistoricalSoundscapesCaseOfEvora-DomainAnalysis/blob/master/Domain_Analisis-Historical_Soundscapes.pdf.

3.2 FeatureIDE plug-in extension

This section explains the reason why this extension to the plug-in was required, and the choice of implementation. Then, it explains the approach to understanding the code and what had to be programmed. Finally, there is a very brief explanation of how the extension works.

Necessity - Feature diagrams allow some information about a specific feature to be expressed. In particular, the FeatureIDE plug-in allows each feature to have a description, as explained in Section 2.3. However, as features are user-visible characteristics, each feature can have a different level of abstraction. For example, a color can be a feature, but a Log-in page can also be a feature in the same feature model. This can lead to a certain level of ambiguity when developers want a better explanation of what the feature is meant to do. As the popular phrase states, "a picture is worth a thousand words". Hence, allowing features to have examples by associating them with multimedia, gives the developers a better understanding of what is intended by the feature.

Initially, the plan was to display the multimedia directly on the feature model. However, that seemed to be a much more complex solution than the time available for its development, could make the diagram harder to understand (by overpopulating it), and would limit its application for other problems. The chosen solution was to allow associating URLs with each feature. Then when a feature is selected, a list of all the associated URLs can be displayed and each URL accessed. As long as the multimedia is online and the user has internet access, by adding the URL to the feature, the objective is still achieved. The main disadvantage is that the multimedia the user wishes to link needs to be online However, nowadays this is not very complicated as there are various tools such as YouTube accounts that allow uploading multimedia to the public. Furthermore, this solution can be applied to many other situations for example linking specific features to open-source repositories containing the implementation code for a feature that may have a very low level of abstraction.

Approach - Figure 4 is a brief explanation of the steps taken when developing the plug-in extension code. First, the open-source code was obtained from https://github.com/FeatureIDE/FeatureIDE. Then, its structure as well as other Eclipse IDE plug-in structures were researched and analyzed. The plug-in code was then run and analyzed to understand what functionalities could be emulated. In this case, the extension was based primarily on the "description" component described in Section 2.3. This component was thoroughly analyzed with debugging as to copy and adapt all steps, mainly the XML tags and XML parsing. Then, the component was adapted and new functionalities such as changing context menus, listing URLs, and pop-up errors had to be studied and implemented. Once it was functional and tested by the author, a user manual for the installation and usage of the plug-in was developed. This manual was then given to a user without prior knowledge or experience with feature models, Eclipse IDE, or any sort of software engineering concepts, to be followed and tested. The user then provided feedback on what could be improved. This feedback was integrated, and finally, it was ready for evaluation which is presented in Section 4.

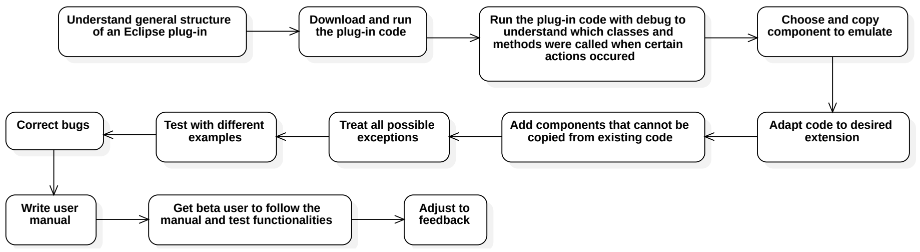
Understand general structure of an Eclipse plug-in
Download and run the plug-in code
Run the plug-in code with debug to understand which classes and methods were called when certain actions occured
Choose and copy component to emulate
Adapt code to desired extension
Add components that cannot be copied from existing code
Treat all possible exceptions
Test with different examples
Correct bugs
Write user manual
Get beta user to follow the manual and test functionalities
Adjust to feedback

Fig. 4. Process used when developing the plug-in extension

Summary of functionalities - The user manual for the plug-in extension is available at https://github.com/marianaBonito/APDC-Investigation-HistoricalSoundscapesCaseOfEvora-DomainAnalysis/blob/master/FeatureIDE_Plugin_Extension_Manual.pdf. In summary, the plug-in extension, allows the user to associate one or more URLs to each feature. The association is done through a dialog box similar to the "description" component and each line must follow a designated structure. Then when the user wishes to access the URL, this can be done by either right-clicking on the feature and selecting "List URLs", or by selecting the feature and then pressing *Ctrl* + *U*. The extension is also ready to deal with exceptions such as loss of internet, or invalid URLs. Nevertheless, a better explanation of all the functionalities the extension covers and the exceptions it is ready to deal with, is available in the user manual.

4 Evaluation

This section presents the evaluation protocol for the contributions described in Section 3, its challenges, and its results. To be able to evaluate the content, the goals must be previously established. In this case, the objective of the domain analysis was to model the domain with particular focus on its requirements in a way anyone involved in the development of a system in the domain could understand and make the implementation/ planning simpler. For the FeatureIDE extension, the goal was to be similar to the current plug-in structure and then easy to use after an adaption period is over.

4.1 Approach

The domain analysis is targeted for people involved in developing a project belonging to the proposed domain. Whereas, the FeatureIDE plug-in extension is targeted for people involved in planning/developing projects using SPLE techniques. Hence, the evaluation should rely on target users' opinions.

To develop the selected evaluation approach, three methods were studied. The first method analyzed was the approach taken by João Rosário in his thesis

[17]. This approach was considered because Rosário's thesis is one of the main basis for the domain analysis. Then the paper on "Cultural Domain Analysis for Soundscape Assessment" by Dhamelia and Dalvi was also studied. They collect data and understand the perception of sounds, by administering a survey "to the participants with Likert-like questions" [5]. In addition, they point out that "Although the qualitative data of the people's opinions are collected, they are not analyzed qualitatively" [5]. This is also true when reading Rosário's interpretation of his results. The final method that was taken into consideration was Roberto Veloso's approach to evaluating his thesis. His thesis is taken into consideration as he presented a new language for modeling interface requirements for WebGIS [22]. Despite not being very similar to the contributions of this investigation, his evaluation was designed to measure the ease of understanding of the models he presented and get participants' feedback [22].

The evaluation methods studied rely on user feedback. Hence, the selected method for evaluating both the domain model and the FeatureIDE extension consisted of creating a form[4] and sending it to the target users.

In this case, the form was sent to all the members of the PASEV team, Bruno Ponte and André Ferreira (master students developing their thesis on the PASEV project), and both members of the Spanish historical soundscapes project. These participants were chosen as they cover all areas of the target users mentioned above, and belong to different projects which should ideally reduce the bias of only representing one software in the domain.

The form has three main sections. First, there is an open answer question section for user information, whilst still maintaining anonymity. This gives context to the answers and allows for an understanding of which areas should be better explained/altered due to users not having the same background knowledge as the author [17,22]. Then, there are open answer questions about the domain analysis, these questions are followed by Likert-like questions where the participant is asked to rate statements on a scale of one-to-five depending on how much they agree with it. The final section is relative to the plug-in extension and its manual. It has the same structure as the previous form section and for the same purpose. The open answer questions are designed to evaluate whether the participant read and understood the documents (domain model and featureIDE plug-in extension manual) [22]. The Likert-like questions are meant to facilitate the analysis of the qualitative evaluations [5,17,22].

4.2 Results

This section analyzes briefly the form evaluation results[5].

[4] The form questions are available at https://github.com/marianaBonito/APDC-Investigation-HistoricalSoundscapesCaseOfEvora-DomainAnalysis/blob/master/Domain%20Analysis%20and%20FeatureIDE%20Extension%20Manual%20Evaluation%20-%20Google%20Forms.pdf

[5] The full form results can be observed with google sheets at https://docs.google.com/spreadsheets/d/1vNGe06-UjMUZo7OuAIuwmOVIhHPYxurEji6L_VMulMU/edit?usp=sharing.

Unfortunately, the form has a lot of questions and takes quite a lot of time to answer. In addition, reading both the domain model and following the featureIDE plug-in extension manual can be very time-consuming. Hence, there were only three answers to the form. The implications caused by this lack of answers are further explored in Section 4.3.

From the context answers given to the form, it is evident that only one of the participants has a software developer background and none have experience with SPLE. The software developer was the only participant with previous knowledge of use-case diagrams and class diagrams. However, none of the participants had any previous experience with feature models.

The open answer questions for the domain model were mostly right, and when asked their opinion, the feedback was all positive. In addition, when asked about the interpretation of the diagrams, all the replies stated they were able to understand the diagrams with the help of the explanation in the appendix. Then, the Likert-like questions for the domain model also had very positive replies with all of the statements, except for one, having answers within the 4 to 5 range (5 represents "Completely Agree" with the statement, and 1 represents "Completely Disagree"). The statements were all done with an optimistic approach to keep consistency and avoid confusing the participants. The optimistic approach refers to stating for example "The terms relevant to the domain are clearly defined", rather than "The terms relevant to the domain are not clearly defined".

The only exception to the previously mentioned range is the "The use-case diagram helps the understanding of how certain requirements are to be enforced" statement. Here, one of the participants responded with a 3 which represents "Moderately Agree". This participant is one who did not have any previous experience with use-case diagrams and replied to the question on use-case diagram interpretation stating it was "a bit confusing at a first glance". This reply indicates that a possible area of improvement, is to establish a clear link between use-cases and the requirements they represent. This would give readers with less experience in software development and UML diagrams, a better understanding of how they are connected.

The section about the featureIDE extension was not as straight forward as the domain analysis evaluation. Firstly, one of the participants said they were unable to follow the steps of the manual. However, the participant also informed, via email, that they only replied to the domain analysis part of the form as "the computer details have to be with other team members". This response was likely due to the participant not having enough time to follow the manual and then answer the questions rather than being unable to follow the manual. Hence, this section only had two proper participants.

Both participants agree that the URL association benefits and enriches the FeatureIDE plug-in. In addition, there were no suggestions made on how to improve the association. On the other hand, there is a clear difference in the answers to questions about the user experience of the plug-in. The software developer participant that had previous experience with git hub repositories, did not find the plug-in or its extension difficult to understand or use. Whereas the participant which is a historical soundscapes expert found them “more or less” difficult. However, when analyzing the rest of the responses, the historical soundscapes expert participant found the plug-in extension manual to be clear and stated the problem encountered was “the plugin use is not intuitive”.

The Likert-like questions for this section of the form were, again, presented with an optimistic approach and the results range from 4 to 5 (with the same scale as before). This indicates the plug-in extension reached its objectives.

The context given for the answers led to the understanding that despite being able to follow and do what is required, the FeatureIDE plug-in and its extension are not very suited for people outside the area of computer science. Nevertheless, as this plug-in was targeted for domain and software developers, it is not a problem.

4.3 Threats to validity

As mentioned, there is a lack of evaluation answers. Despite attempting to engage with a large spectrum of participants to cover all possibilities of the target audience mentioned in Section 3.1, the outcome was not as expected. This limits the validity of the evaluation as the margin of error increases, individual participants’ biases have greater weight on the results, and in turn the evaluation is less reliable. In an ideal situation, there would be more time for the participants to answer the form. This could eventually lead to different results than the ones observed.

5 Conclusions and future works

There are a few points that could be improved upon for future work. These include, but are not limited to: exploring the solution of linking code to the individual features; when the domain has enough information, switching to the approach suggested by Liu et al. [13] to update domain knowledge based on app descriptions and user reviews; constantly updating the domain model when new information is made available; and finally, allowing more time for the participants to answer the form and adapt the documents to the feedback given.

From the information presented in this document, one can conclude both contributions made by this investigation had good evaluations and consequently reached their goals effectively. Some points could be improved or modified for the next update of the domain model. These include having a direct link between use-cases and requirements and adding a role to the system users suggested by Lizáran Rus. Nevertheless, this does not affect the usefulness of the contributions as it was mentioned many times the need for a systematic update of the domain model.

6 Acknowledgements

I would like to thank both of my advisors Professor Armanda Rodrigues and Professor João Araújo for helping and giving me feedback throughout the duration of this investigation. I would also like to thank João Cambeiro as the development of the plug-in extension would not have been possible without his help.

References

1. Arcaini, Paolo & Gargantini, Angelo & Vavassori, Paolo. (2015). Generating Tests for Detecting Faults in Feature Models. 2015 IEEE 8th International Conference on Software Testing, Verification and Validation, ICST 2015 - Proceedings. 10.1109/ICST.2015.7102591.
2. Bayer, J., Muthig, D., & Widen, T. (2000). "Customizable Domain Analysis." Generative and Component-Based Software Engineering: First International Symposium, GCSE'99, Erfurt, Germany, September 28-30, 1999: Revised Papers, by Krzysztof Czarnecki and Ulrich Eisenecker, Springer,(G. Goos, J. Hartmanis, & J. van Leeuwen, Eds.) pp. 178–194.
3. Bonito, M. (2020). Domain Analysis - Historical Soundscapes with Geolocation (Tech.). Retrieved from https://github.com/marianaBonito/APDC-Investigation-HistoricalSoundscapesCaseOfEvora-DomainAnalysis/blob/master/Domain_Analisis-Historical_Soundscapes.pdf
4. Bonito, M. (2020). FeatureIDE URL Plugin Extension - Manual
5. Dhamelia, M., & Dalvi, G. (2019). Cultural Domain Analysis for Soundscape Assessment. (A. Chakrabarti, Ed.). Springer Nature Singapore.
6. Frakes, W., Prieto-Diaz, R. & Fox, C. (1998). DARE: Domain analysis and reuse environment. Annals of Software Engineering 5, 125–141. doi: https://doi.org/10.1023/A:1018972323770
7. Guzi, M. (2017, May 4). The Sound of Life: What Is a Soundscape? Retrieved March 2, 2020, from https://folklife.si.edu/talkstory/the-sound-of-life-what-is-a-soundscape
8. Harsu, M. (2002). A surveu on domain engineering (pp. 0–27).
9. Ifuku, M., Kushiro, N., & Aoyama, Y. (2018). Requirements Definition with Extended Goal Graph (2018 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 210-218, Tech.). doi:10.1109/ICDMW.2018.00039
10. Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., & Peterson, A. S. (1990). Feature-oriented domain analysis feasibility study (pp. 1-147). SEI Technical Report CMU/SEI-90-TR-21.
11. Khalique, F. H., Butt, W. A., & Khan, S. U. (2017). Creating Domain Non-Functional Requirements in Software Product Line Engineering using Model Transformations (2017 International Conference on Frontiers of Information Technology, Rep.). Dept. of Computer Engineering National University of Sciences and technology Islamabad, Pakistan. doi:10.1109/FIT.2017.00015
12. Kushiro, N., Shimizu, T., & Ehira, T. (2016). Requirements Elicitation with Extended Goal Graph (20th International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES2016, 5-7 September 2016, York, United Kingdom, Rep.). doi:10.1016/j.procs.2016.08.217

13. Liu Y, Liu L, Liu H, Yin X. App store mining for iterative domain analysis: Combine app descriptions with user reviews. Softw: Pract Exper. 2019;1–28. https://doi.org/10.1002/spe.2693
14. Lizarán Rus, I. J. (2020, June 24). Domain Analysis on Historical Soundscapes [E-mail to the author].
15. PASEV • Patrimonialization of Évora's Soundscape • 1540-1910. (2019). Retrieved from https://pasev.hcommons.org/
16. Plataforma PASEV. (2019). Retrieved from https://pasev.di.fct.unl.pt/#brand
17. Rosário, J. D. B. A. (2019). Uma plataforma responsiva para o Atlas Auditivo de Évora (Unpublished master's thesis). Faculdade de Ciências e Tecnologia - Universidade Nova de Lisboa.
18. Ruiz Jiménez, J., & Lizarán Rus, I. J. (2015). Paisajes sonoros históricos (c.1200-c.1800). Retrieved from http://www.historicalsoundscapes.com/
19. Sterne, J. (2013). Soundscape, Landscape, Escape. De Gruyter. Retrieved from https://www.degruyter.com/downloadpdf/books/9783839421796/9783839421796-010/9783839421796-010.pdf
20. Thurimella, A. K., & Padmaja, T. M. (2014). Economics-driven software architecture. (Mistrík Ivan, R. Bahsoon, R. Kazman,& Y. Zhang, Eds.). doi: https://doi.org/10.1016/C2012-0-02842-6
21. Truax, B., Westerkamp, H., Woog, A. P., Kallmann, H., & Mcintosh, A. (206AD, February 7). World Soundscape Project. Retrieved March 2, 2020, from https://thecanadianencyclopedia.ca/en/article/world-soundscape-project
22. Veloso, R. J. M. (2019). Uma linguagem para modelação de requisitos de interface para WebGIS (Unpublished master's thesis). Faculdade de Ciências e Tecnologia - Universidade Nova de Lisboa.
23. Willie, R., & Murphy, E. (2020). Soundscapes in the early modern world. Retrieved March 2, 2020, from https://emsoundscapes.co.uk/
24. Y. Liu et al., Mining domain knowledge from app descriptions, The Journal of Systems and Software (2017), http://dx.doi.org/10.1016/j.jss.2017.08.024
25. Zaragoza, M. G., & Kim, H.-K. (2018). Mobile Application Development on Domain Analysis and Reuse-Oriented Software (ROS) (Computer and Information science, Tech.) (R. Lee, Ed.). Springer International Publishing AG. doi:10.1007/978-3-319-60170-0_14