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

Mariana Bonito

m.bonito@campus.fct.unl.pt

Orientadores: João Araujo, Armanda Rodrigues
Faculdade de Ciências e Tecnologia – Universidade NOVA de Lisboa

Abstract. Historical soundscapes are very underrated and disregarded. Until recently, there has not been a lot of investigation in understanding the concept. However, currently there is an ongoing project (Patrimonialization of Évora's Soundscape) between NOVA LINCS, Universidade de Évora, and other associations to develop a platform which provides access to historical soundscapes in specific locations. The aim of this work is to initialize a domain analysis for historical soundscapes in the context of Évora. The importance of historical soundscapes is clarified in this paper. It also explores in depth several different approaches to domain analysis, selecting one for future work whilst justifying the choice. The result of the domain analysis will be presented in the following report.

Keywords: Domain Analysis, Soundscapes, Historical Soundscapes

1 Introduction

Sound helps us understand and contextualise what is happening in our environment as well as allowing for other forms of expression (it being oral communication, music, etc.). Due to sound being mainly ephemeral - "it does not endure long past its production" [1] - it can many times be forgotten or disregarded, especially where historical resources are concerned. This is of relevance as it plays a "cultural role within our acoustic ecology" [1]. Soundscape is the technical term used in order 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 [1]. As Emily Thompson states, "soundscape is simultaneously a physical environment and a way of perceiving that environment" [2]. 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 [1].

Given the importance of soundscapes, but more specifically, historical soundscapes there have been many attempts by different global organisations such as the "Early Modern Soundscapes" or the "Simon Fraser University" to explore

and offer to the public, information and access to soundscapes of the past [3,4]. Recently, the EU, along with the Spanish and the Portuguese governments have taken more of an interest in providing a platform for people curious about 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 [12,13,14].

This work is part of the PASEV project which is developing a platform[1] designed to share interactively the historical soundscapes of Évora with the user. The first prototype was the result of the thesis and work of João Rosário [5]. Jõao Rosário's work included a partial domain withdraw. This work will obtain the domain knowledge required from the current prototype, the partial domain withdraw from Rosário's work, and other documentation supplied (such as the forms provided by the project's team and literature). The aim of this work is to gather the domain knowledge available, determine a domain analysis approach and finally initiate a domain analysis following the selected approach. This will include a class diagram (to demonstrate in more detail relationships between concepts), a feature diagram (to display the variability of the concepts), and if time permits, a use case diagram (to illustrate possible functionalities assigned to stakeholders). To do so, first, the domain must be well analysed to capture the fundamental characteristics. For now, let us consider the domain to be: Historic Soundscapes. This will be scrutinized on the next report as it is part of the domain analysis.

The purpose of domain analysis is to establish guidelines which allow for the reuse of common components amongst projects in the same domain. Frake et al. (1998) state that the shift from software engineering to domain engineering and consequently domain analysis "is based on the observation that quality and productivity can be significantly increased" when implementing projects in the domain. Hence, the results of this project should facilitate the development of platforms within the established domain. It is a relatively recent research area. Thus, there are some suggested approaches on how to develop a domain analysis (explored in section 2), but there is no defined method of asserting the best option.

The research problem has been presented in this section. On the following sections, the various approaches to the domain analysis are explored and the chosen solution, justified. The tools and methodologies used to reach the goal of this project are also described in section 3, and a detailed work plan is presented in section 4.

2 State of the art

As previously stated, there is no answer for how to develop the best domain analysis. In this section some of the most popular methods are explored in detail and evaluated in the context of this project.

[1] There is already a published prototype which can be found at: https://pasev.di.fct.unl.pt/#brand [13].

All methods here studied 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). However, there are "two vexing problems of domain analysis" [8]: What should the domain model be? How does one know that the analysis is complete? These are the points of interest when comparing the analysed methods.

2.1 Feature Oriented Domain Analysis (FODA)

Feature Oriented Domain Analysis was one of the first established protocols for domain analysis. In this case, as the name indicates, the 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". As outlined in Figure 1 the domain analysis in this process is divided in 3 phases: context analysis (determines the scope of the domain), domain modeling (clarifies requirements, features, and entities), and architectural modelling (defines architecture of systems in domain). The domain model in this case consists of the column 'Product' in Figure 1. Despite the completion criteria not being evident, it can be assumed that the analysis is complete once the domain model is finished.

Phase	Inputs	Process	Product	Description
Context analysis	Operating environments, Standards	Context analysis	Context model	Environments in which the applications will be used and operated
Domain modelling	Features, Context model	Features analysis	Features model	End-user's perspective of the capabilities of the applications in a domain
	Application domain knowledge	Entity-relationship modelling	Entity-relationship model	Developers' understanding of the domain entities (objects) and their relationships
	Domain technology, Context model, Features model, Entity-relation model, Requirements	Functional analysis	Data flow model	Requirements analyst's perspective of the functionality of the applications
			Finite state machine model	
Architectural modelling	Implementation technology, Context model, Features model, Entity-relation model, Design information	Architectural modelling	Process interaction model	Designer's perspective of the high-level structure (architecture) of the applications
			Module structure charts	

Fig. 1. Overview of the FODA process [10]

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 2). A feature diagram can also me called a feature model which is present in the domain model of this method. This reiterates how this diagram plays a major role in domain analysis.

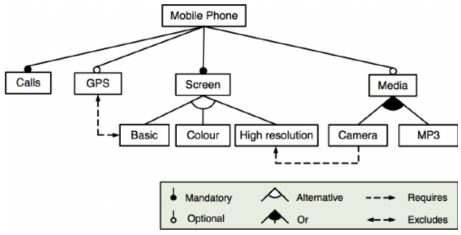

Fig. 2. Self explanatory example of a feature diagram[11]

2.2 Domain Analysis and Reuse Environment (DARE)

The DARE approach is probably one of the best known methods for domain analysis. In this procedure the domain model output is referred to as the domain book. Its content is described in detail in Figure 3 [2]. The book metaphor is used to represent the domain information gathered from the various sources. This method permits a thorough analysis of the domain giving it a clear structure and completion criteria. The method is complete when all sections of the 'book' are thoroughly filled.

Sections	Chapters	Entries	How created
Table of Contents	–	Headings	Automatic
Part 1: Domain Sources	Source Documents	Document files	User input & ordering
	Source Code	Code files	User input & ordering
	System Descriptions	Structured text	Directed user input
	System Architectures	Architecture specs	Graphical user input
	System Feature Table	Feature table	Assisted user input
	Source Notes	Notes	User input & ordering
Part 2: Vocabulary Analysis	Basic Vocabulary	Keywords & phrases	Assisted text analysis
	Facets	Facet table & editor	Assisted text analysis
	Synonyms	Synonym table	Assisted user input
	Templates	Template table	Assisted user input
	Thesaurus	Terms & definitions	Assisted user input
	Vocabulary Notes	Notes	User input & ordering
Part 3: Architecture Analysis	Generic Architecture	Architecture spec	Graphical user input
	Generic Features	Feature table	Assisted user input
	Code Structure	Hierarchies	Automatic code analysis
	Architecture Notes	Notes	User input & ordering
Glossary	–	Words & definitions	User input, auto ordering
Bibliography	–	Citations	User input, auto ordering
User Index	–	Words & locators	Automatic
Appendix	Analysis Parameters	Parameters & values	Automatic
	Activities Log	Event descriptions	Automatic

Fig. 3. Content of domain book [8]

Another point which makes this method unique is the intense focus on vocabulary analysis. The DARE creators developed a tool (also called DARE) which

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

apart from allowing for the domain book to be filled, also has a functionality (DARE cluster editor) which clusters automatically related words and phrases[8].

This method also gives great emphasis on the importance of a domain expert stating that they "are the most important source of information for domain analysis". A key aspect of high significance to the DARE method is their knowledge in the architecture of each system in the domain [8]. Unfortunately, for this work there is no domain expert and all research must be done from the available documentation/ other resources.

2.3 Product-Line Software Engineering- Customizable Domain Analysis (PuLSE- CDA)

When it comes to software engineering, there are two main types. Product-Line Engineering (PLE) and Single-System Engineering (SSE). As the name suggests, this method is aimed at Product-Line Engineering. The domain scope for PLE is a family of similar projects etc. whereas the domain for SSE is focused solely on a particular project. The overview of the PuLSE-CDA process is described in Figure 4.

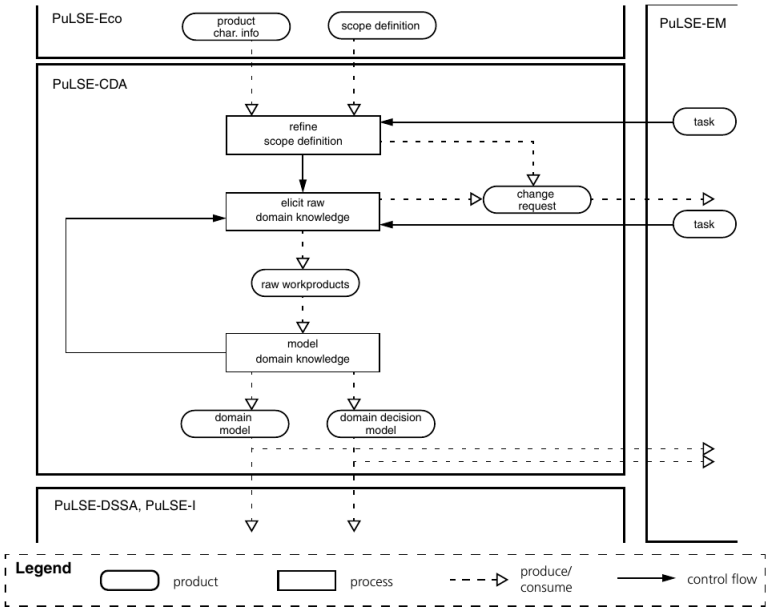

Fig. 4. PuLSE-CDA Process Overview [7]

This method begins by analysing 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 analysed" [7]. Then, raw/unstructured domain knowledge is gathered from various sources and "captured in the domain model workproducts" [7]. 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" [7], hence, it gives great focus on "product line infrastructure evolution". This is achieved through a constant step to re-analyse the requirements and domain knowledge. Hence, 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. On the down side, as this is constantly evolving, there is no completion criteria as the analysis is never over.

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 use of a tool they developed called Diversity/CDA and introduce the idea that the domain model must contain meta elements. A meta element is a notation for generic workproducts that illustrate the variation points and allow instantiation of the workproducts [7].[3]

The domain decision model is the second aspect which most differentiates this procedure from the rest of the analysed methods. It 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" [7]. These decisions are structured based on constraints amongst them and may have different levels of abstraction.

2.4 Selected Approach

The approach taken in this work will be a mixture of all analysed and will follow the outline presented on the paper "A survey on domain engineering" [9].

The FODA approach has the advantage of having very clear phases, benefits of both generic components and refinements of said components, and due to it being one of the first, most other methods have been partly based on it. However, this work, does not require as much detail as this method provides, and despite it producing a clear result, it does not provide completion criteria.

The DARE method "can be used to support" [8] other methods. Its main advantages are that it has a clear output and completion criteria. In addition, its structure is easy to follow and covers in much detail all the concepts behind the domain (such as vocabulary and platform architecture). On the other hand, this work, does not require such an in depth analysis of the domain vocabulary, nor does it have the ability/need to involve a domain expert who plays such an important role in this method.

The PuLSE-CDA's main advantage is the constant reanalysis of the domain knowledge and the way it is prepared to evolve with the links between the information and the sources. However, it does not clearly specify what the domain model should be, nor does it suggest any completion criteria.

[3] It is worth noting that this concept is very similar to that which is represented in feature diagrams.

To summarise, 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 [9]. 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 are also included as they provide useful products for this problem.

The feature diagram plays a major role on all analysed 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 to 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 will be used in order to help justify the features presented. It will aid developers to identify which features their software will require. In addition, it will help understand the limits of the domain as it indicates what is covered by the analysis.

Hence, the method that will be applied will consist in creating a proposal of a domain book of sorts. Yet, this will only contain the following:

Overview	Detailed limits of domain scope and requirements along with links to their sources
Definitions	Dictionary of technical terms
Commonalities	Structured list of assumptions that are true for all systems in the domain
Variabilities	Structured list of assumptions on how systems in the domain differ
Issues	A record of important decisions and their alternatives
Domain modeling	Feature diagram and Class diagram
Architectural modeling	Process interaction model (Use case view)

Table 1: Domain model outline for this project

3 Tools and Methodologies

In order to achieve the goals of this project, several tools will be required. In order to do the domain analysis, the documentation and forms provided by João Rosário and the client when developing his thesis must be analysed, as they will be the main source for domain knowledge.

As mentioned on section 2.2, there is a domain analysis tool called 'DARE' which helps implement all required sections of the DARE method. Despite the chosen approach being highly based on the DARE method, it does not require the same depth of analysis, making the main benefits of the platform (such as the vocabulary clusters) non applicable. Hence, to develop the domain analysis,

the document will be typed in overleaf and the diagrams will be developed using the StarUML software (Class Diagram and Use Case Diagram) and featureIDE for eclipse (Feature Diagram).

This project aims at initializing a domain analysis approach not solely focused on the particular case of the PASEV platform. Hence, the effectiveness of the analysis approach, in facilitating future reuse of components for a range of other projects in the domain, is of utmost importance.

4 Work Plan and Scheduling

The schedule outline can be found in the appendix in section 5.1. The work plan for this project is divided in five main topics:

1. **Research** about the domain of historic soundscape and different approaches to domain analysis. This requires the use of the available documentation from João Rosário's work, and extra literature about the domain.
2. Develop the **class diagram** using the knowledge gathered in the research part and the StarUML software.
3. Develop the **feature diagram** using the domain knowledge from the research part and the FeatureIDE for eclipse. There will be an attempt to include multimedia content in this diagram which is not usually part of the content and hence there is no certainty it will be accomplishable.
4. Develop the **use case diagram** using the knowledge gathered in the research part and the StarUML software.
5. Write the **final report** which will contain the domain model (its outline is in section 2.4)

References

1. 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
2. Sterne, J. (2013). Soundscape, Landscape, Escape. De Gruyter. Retrieved from https://www.degruyter.com/downloadpdf/books/9783839421796/9783839421796-010/9783839421796-010.pdf
3. Willie, R., & Murphy, E. (2020). Soundscapes in the early modern world. Retrieved March 2, 2020, from https://emsoundscapes.co.uk/
4. 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
5. Rosário, J. D. B. A. (2019).
6. 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

7. 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.
8. 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
9. Harsu, M. (2002). A surveu on domain engineering (pp. 0–27).
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. 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.
12. PASEV • Patrimonialization of Évora's Soundscape • 1540-1910. (2019). Retrieved from https://pasev.hcommons.org/
13. Plataforma PASEV. (2019). Retrieved from https://pasev.di.fct.unl.pt/brand
14. Jiménez, J. R., Rus, I. J. L. (2015). Paisajes sonoros históricos (c.1200-c.1800). Retrieved from http://www.historicalsoundscapes.com/

5 Appendix

5.1 Schedule outline