Domain Analysis - Historical Soundscapes with Geolocation

Mariana Bonito

m.bonito@campus.fct.unl.pt

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

Abstract. There is a new approach to visiting historical sites emerging. It consists of associating multimedia to geolocations on software to create a historical soundscape environment. The ongoing Patrimonialization of Évora's Soundscape (PASEV) project is attempting to do just that. This paper is part of an investigation and presents the results of domain analysis for software similar to the one being developed by the PASEV project. The domain under analysis is an *interactive and multimedia-based historical soundscape environment with geolocation*. There are currently only two software available in this domain hence, the analysis addresses that limitation.

Keywords: Domain Analysis, Soundscapes, Historical Soundscapes

1 Introduction

This paper presents a domain model output generated by an initial domain analysis of the historical soundscapes domain, which is part of an investigation project. It was created to help model future application development of the ongoing project *Patrimonialization of Évora's Soundscape* (PASEV) as well as the conception of further similar software. As Rabiser et al. define, this paper uses the term domain model to refer to a model that "describes the requirements relevant to characterize all systems in a domain"[23].

A "domain is the total body of information for which, ideally, an answer to that problem is expected to account"[27]. In this particular case, the "problem" faced is finding a way to model and convey the information required to begin developing software in the domain (better specified in Section 2).

An important point that must be mentioned about this domain analysis, is that it requires a systematic update whenever there is an alteration or new software is developed in the domain. This is because, as Hjørland points out, "domains are not ready-made divisions of the world but are dynamic" and developing [10]. This means, that despite defining a domain now, with its requirements, features, etc., the domain model will become obsolete unless there is an ongoing effort to update it. There might even be a necessity to redefine the limits of the domain.

There has been a study by Liu et al.[16] that re-enforces how this concept is exceedingly relevant when it comes to developing mobile applications as there are "about 3% new features appearing in the domain every half-month" and "mainstream apps release a new version within 15 days on average" [16]. They suggest a way to extract, iteratively, relevant information from app descriptions in the domain and their reviews, and then creating a feature-based domain state model. This allows developers to track how certain features evolve and enables a better critical assessment of said features. However, as (to the best of the author's knowledge) there has not yet been any app, or domain analysis, developed for this domain, for now, this approach is still not a viable option. However, once there is more software developed, this is a method worth implementing.

Nevertheless, this domain model will implement a structure described in the author's investigation report [3]. This structure incorporates part of the PuLSE-CDA domain analysis approach by linking the requirements to their source. With this method, whenever a source is altered/added the requirements can be tracked through the diagrams presented and their content altered correspondingly.

The rest of this paper is organized as follows. In Section 2, the domain for this paper and its limits are defined. It also clarifies the target users for software development in this domain. Section 3 defines some of the concepts relevant to the domain a reader may not be acquainted with. The following Section 4, outlines both the features that can be observed in all the software currently developed in the domain (commonalities) and the ones that may appear in some software but not all (variabilities). Section 5 explains in detail all software requirements in the domain and provides a use case diagram to illustrate the uses of the functional requirements. Section 6, contains the class diagram and the feature models for both the front-office platform and the back-office platform. Then, Section 7, outlines some of the problems and options encountered when developing the domain model, the selected approach for each, and the corresponding alternatives. Finally, in Section 8, a summary of the content is presented and final remarks are made. For the evaluation of the domain model and the conclusions of the author's investigation, please refer back to the investigation report [3].

Important: Please be aware that some diagrams may not be readable if this domain model is not read with a PDF editor that allows zooming in. If this is the case, the diagrams are also available in the author's investigation git repository: https://github.com/marianaBonito/APDC-Investigation-HistoricalSoundscapesCaseOfEvora-DomainAnalysis/tree/master/Diagrams.

2 Domain Scope

Defining the domain is one o the most important tasks when developing a domain analysis. However, defining one is quite complex, and "in order to examine a domain, you must know something about that domain"[10]. Hence, to define the domain, the forms provided by the PASEV experts, the PASEV website, the current PASEV prototype, and João Rosário's thesis, along with the Spanish

project *Historical soundscapes (c.1200-c.1800)* had to be thoroughly analyzed [24,21,22,25,26].

After much deliberation and consultation with the two project advisors, it was determined that the domain which best incorporates the desired PASEV platform is: ***Interactive and multimedia-based historical soundscape environment with geolocation***. It is worth noting that this domain supports both front-office and back-office platforms. The front-office platform is intended for client interaction such as having access to the historical soundscape. Whereas the back-office platform is provided in order to manipulate the information presented to the clients, and should only be available for members of the responsible organization. Furthermore, the content of this domain analysis should not limit the type of software developed. This is to say, the models presented should allow for the software developed to be an app, a website, or any other type of software as long as it belongs to the established domain.

2.1 Domain Users

As this domain is focused on historical soundscapes and their perception, one can assume that software developed in this domain will target users who are not required to have an extensive grasp of technology. As previously mentioned, this initial domain analysis is based mostly off of the PASEV project and the Spanish *historical soundscapes* project. From there, two main users can be extracted: a specialist administrator user, and a client user. These users are by no means the only users allowed for the domain, but from the information available currently, they are the only ones that can be observed.

The specialist administrator user is a user that is part of the responsible organization for the software and has access to the back-office platform. This user should be a specialist in the history, more specifically soundscape history, of the location the software focuses on. In addition, the user is responsible for providing and populating the historical soundscape data in the system. In the PASEV example, this user would be a member of the PASEV team that knows the historical-cultural events of Évora. Hence, there is no requirement for the user to have any expertise in technology.

The client user is a user that has access to the historical soundscape data through the front-office platform (a website, an app, etc.). This user is a common person with an interest in the history of the location the software focuses on. It is intended that anyone with access to the platform, could be represented by this user. The developers cannot assume any prior knowledge of technology, the history of the location the software focuses on, or the domain itself. In the PASEV case, this user represents a tourist in Évora that wishes to experience the historical soundscape of cultural events around town.

When planning the software, the developers must keep in mind these users' limitations and design platforms suitable for their capabilities.

3 Domain Concepts

In this section, the "standard vocabulary of domain experts"[11] considered essential, are defined for the reader to fully comprehend the content of the domain model presented.

Soundscape - Dhamelia and Dalvi present two definitions for *soundscape*[7]:

1. "An acoustic environment consisting of events heard, rather than objects seen".
2. "An acoustic environment as perceived or experienced and/or understood by a person or people, in context".

Their paper emphasizes the "importance on the perception of the sound by people rather than physical measurement of sound levels in a place"[7]. Sterne's paper includes a definition for soundscape as being "an auditory or aural landscape" which is "simultaneously a physical environment and a way of perceiving that environment"[28].

Hence, when referring to the soundscape, the reader should focus on the two key concepts: *acoustic environment* and *perception of the sound*.

Historical soundscape - Having *soundscape* defined previously, helps the understanding of *historical soundscape*. If we focus on the two key concepts presented above, *historical soundscape* can be viewed as the acoustic environment of the past, but might also require context to be given as the perception of the sounds will likely differ over time.

Lee et al. define historical soundscapes as being an "approach to representing the past through sound"[14]. They also defend that *historical soundscapes* are important as they "can inform our understanding of the past by opening new spaces for examining, communicating, and understanding the past using sound"[14].

Itineraries - For this domain, an *itinerary* is defined based on João Rosário's definition: a textual description of an ordered set of locations[25]. In other words, an *itinerary* can be thought of as a predetermined route between specific locations that have a description and are represented on the platform as a line connecting the locations on the map.

Events - In this context, *events* only refer to culturally relevant historical events. In addition, they are associated with one or more types such as a procession or protest.

Locations - The terms location and place are used interchangeably in this domain model. They refer to a location in a city that is of historical relevance. Events are associated with one or more locations. Hence, the locations allow for specific sounds to be heard creating a more diverse, realistic, and captivating portrayal of the historical soundscape of the area the software developed focuses on.

Period vs duration - To understand the concepts of this domain model and more specifically the class diagram presented in Section 6.1 the difference between what is referred to as a *period* and what is referred to as a *duration* must be clear for the reader.

The *duration* refers to how long a specific event took. It is meant to represent whether it is a cyclical event or if it only happened once. It is also intended to allow the specialist administrator user to indicate which calendar its recurrence refers to. The *duration* registers the day an event started, along with the start time (can indicate if it took place in the morning, afternoon, evening, or the specific time), the number of days it lasted, and the time it finished at.

On the other hand, a *period*, for this domain analysis, refers to "a portion of time in the life of a nation, civilization, etc. characterized by the same prevalent features or conditions"[20]. The *periods* are to be defined by the specialist administrator users, which can then help the client users filter the information they desire. They can be labeled however the responsible organization decides, as long as there is consistency throughout each software. Two examples of labels are "Medieval Period" and "16th century".

Map layer - There are various definitions for *map layers*. These depend on what platform is being used in the software. Two of these definitions are:

1. *Map layers* "are the mechanism used to display geographic datasets"[1].
2. A *map layer* is a group of "features representing a particular class or type of real-world entities" it "contains both the visual representation of each feature and a link from the feature to its database attributes"[5]

Figure 1 shows an example of how *map layers* can be placed over each other to show a single image.

In this domain, the *map layers* are used for many things including:

- Displaying specific locations/pins.
- Allowing the client user to select whether the platform should display the historical cartography or other relevant *layers* over the *basemap layer*.
- Display the route of a selected itinerary.

A *basemap layer* is a "type of map layer that provides an optimized map display on which you display your dynamic operational information"[2]. In other words, it is the main *layer* presented on the platform, where all other *layers* are

Fig. 1. Example of combining map layers [5]

placed over. The platform may wish to present options for this *layer* to the client user, allowing it to change depending on the user's selection.

Figure 2 demonstrates the use of *layers* and the interaction a client user may have with them. It is an example from the PASEV platform illustrating what happens when multiple *layers* are requested to be piled over the *basemap layer*.

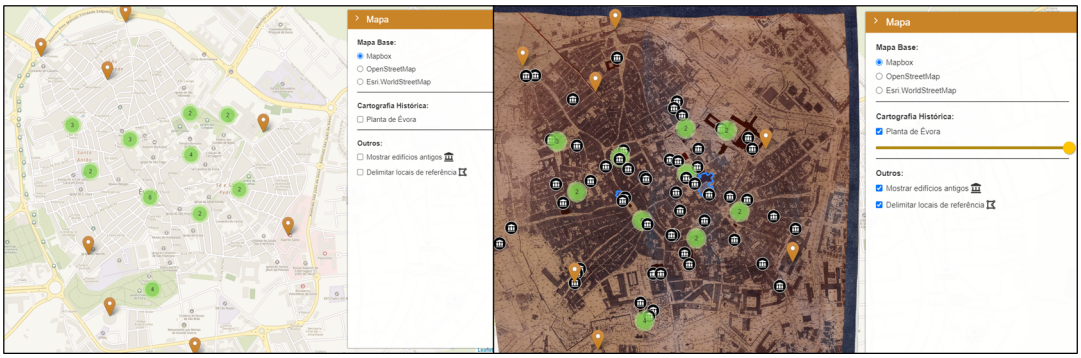

Fig. 2. The image on the right serves as an example of adding 3 layers to the image on the left in the PASEV platform [22]

Institutions - In this domain analysis *institutions* refer to their regular definition. However, they are mentioned in this section as they can play three different roles described below:

1. **Source -** When an *institution* is registered with the role of a *source* it serves to inform where a particular media or map layer was retrieved from. It differs from the *Reference* entity as a reference is usually a mention of a particular *source*.
2. **Participant -** When referred to as a *participant*, it means the *institution* was involved in the execution/ participated in a particular event. However, it does not imply it organized the event.

3. **Organizer -** If an *institution* is considered an *organizer*, it indicates it played a role in the organization of a particular event. Again, it does not imply the *institution* participated in the carrying out of the event.

4 Software Analysis

This section, along with the rest of the domain model, uses the concepts defined in Section 3 to convey information about the domain. One of the main steps in product-line engineering is analyzing the current software developed in the proposed domain, to extrapolate the mandatory and optional requirements described in Section 5[8,9,16,31].

Due to the domain under analysis being relatively unexplored, we analyse two software developed in the domain. The PASEV project software, and the Spanish historical soundscapes software [22,26]. Hence, any attempt to analyze the software in the domain to later abstract into domain requirements will be very limited. Nevertheless, a thorough analysis of both software was conducted and is detailed in sections 4.1 and 4.2. Section 4.1 is used to list the observable qualities present in both software. Whereas Section 4.2 lists the observable qualities present in only one of the software.

4.1 Developed Software Commonalities

Commonalities are "a structured list of assumptions that are true for every member of the family"[9]. In this case, the family refers to the *interactive and multimedia-based historical soundscape environment with geolocation* domain. Unfortunately, as it is a relatively new concept, the extraction of said assumptions are limited to the two developed software in the domain [22,26]. In this section, the observable commonalities between the two projects are listed and detailed. These client observable components are known as features and are better explored in Section 6.

The observed commonalities are as follows:

- Both projects developed a website as their front-office platform. This is to say, when a client user wishes to access the historical soundscape data, they do so by accessing the website and browsing it. In addition, both platforms seem to have been adapted to mobile view with different display sizes.
- Both projects have specific pages dedicated to explaining the purpose of the project as it is a relatively new concept. However, the PASEV project has two separate websites, whereas the Spanish historical soundscapes project has all the information on one website.
- In order to access data on the historical soundscapes of specific locations/events, this must be done through the interaction with the map and its pinned locations.
- There are multiple map layers the client user can select on both platforms.
- The available pins/displayed area on the map can be filtered based on a selected period, a selected location, or a selected event.

- Selecting a location to display is available through an input of the location (including a drop-down list that suggests the locations based in what the user has typed so far), or by interacting with the map (dragging across it, and zooming in and out).
- Both platforms allow the user to select an event from a predetermined list (which can be filtered with user input) and see the locations and corresponding media data relevant to that event.
- Both websites present a list of available itineraries that the user can select from. Once selected, the platform displays the map with the locations relevant to such itinerary linked with a line tracing the route between them.

4.2 Developed Software Variabilities

Variabilities are "elements and relationships [...] found in only some of the systems in the family"[8]. As mentioned in Section 4.1, this family is the domain under analysis. Again, the scope of software under examination is limited to the two platforms developed so far in the domain[22,26].

The observable differences (variable features) between the two systems are listed and detailed below:

- A back-office platform is only present on the PASEV project (that the author is aware) and has not been optimized for mobile view.
- The back-office platform has a login page and 4 tabs, one for adding a file, one for adding a location, one for adding an itinerary, and one for adding an event. João Rosário mentioned in his thesis that the implementation of editing and deleting each of the entities previously listed are not yet implemented but already have REST endpoints associated with each operation [25]. Hence, it can be interpreted as part of the PASEV platform.
- In the PASEV case, the information is linked to the specific locations whereas the Spanish historical soundscape project links the multimedia to specific events and not to the locations.
- In the Spanish historical soundscapes project, there are two separate tabs, one with the map and information regarding Granada's historical soundscape and the other with the map and information regarding Seville's historical soundscape. In addition, there is a clear intention in adding more cities to the system. In the PASEV case, this is not so.
- Despite both platforms allowing for a temporal filter, the PASEV platform does so through implementing an interactive timeline filter whereas the Spanish historical soundscapes software achieves it through selecting the beginning of the period and the end of the period from a predetermined list of years.
- The PASEV website requests access to the user's location to display it on the map and to eventually allow for storytelling.
- The layout of the websites is also very different. The steps necessary in order to display specific information in the Spanish historical soundscapes project are not the most intuitive. This is to say, the PASEV prototype is much more

user friendly. However, the disposition of the Spanish historical soundscapes platform allows more in-depth information to be displayed. This suggests a slight difference in the target client users:

1. For the PASEV project, the client users are tourists with some interest in the historical soundscape of Évora but do not require such in-depth information about the context nor prior knowledge on how to use the platform.
2. For the Spanish historical soundscapes project the client users are more likely academics or people with a greater interest in the Spanish historical soundscapes as the disposition of the website suggests it is more aimed at being a source of information than necessarily being user friendly (something tourists/ users with a mild interest would require). For the client users to use this platform efficiently they would likely require a greater period of adaption to explore how the different aspects of the website behave in comparison to the PASEV platform.

– A QR Code scanner that allows a user to scan a QR code on a specific location and, through augmented reality, display the information relevant to said location. Despite not being present on either of the developed software, this feature is discussed in João Rosário's thesis as being a desired feature for the PASEV project that is to be implemented. Thus, it is considered as a feature of the platform regardless of not being complete.

5 Domain Requirements

The observations made in Section 4 are used as the basis for obtaining the domain requirements. The observed components are abstracted and are analyzed in order to determine their importance in the domain. There are two levels of importance for requirements, mandatory and optional. As the names indicate, mandatory requirements are requirements that all systems in the domain must fulfill, whereas optional requirements are requirements desirable only to some systems in the domain. Not all commonalities presented in 4.1 are mandatory requirements, and some of the variabilities are considered mandatory despite not being present on both systems. Some requirements were not obtained from either of the developed software. Rather they are solely the product of discussions with the project advisors.

In this section, both functional and non-functional requirements are presented in the form of tables with three columns: the name, a brief description of the requirement, and its source. The last column is based on the PuLSE-CDA domain analysis approach and facilitates the update of the domain model whenever there is an alteration in a source.

Furthermore, these requirements are split into two sections: *5.1 Mandatory Requirements*, and *5.2 Optional Requirements*. There is a third sub-section 5.3 which includes a use case diagram that supports the functional requirements of software designed in this domain.

Note: All requirements are the product of analysis of the available documents referenced in the source column of the respective row and discussion with the two project advisors.

5.1 Mandatory Requirements

Tables 1 and 2 contain the mandatory requirements for the whole domain. Hence, they only include the components every single software developed in the domain *must* have.

Table 1. Domain mandatory functional requirements

Requirement	Description	Source
Spatial filter mechanism	This filter mechanism limits the displayed locations. This location filter should be possible by zooming in and out and dragging across the map.	Analysis of the PASEV prototype [22] and the Spanish historical soundscapes project [26], along with Jõao Rosário's thesis [25].
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 [22] and the Spanish historical soundscapes project [26], along with Jõao Rosário's thesis [25].
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 [22] and the Spanish historical soundscapes project [26], along with Jõao Rosário's thesis [25].
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 [22] and the Spanish historical soundscapes project [26], along with Jõao Rosário's thesis [25].
Display routes on the map	In order to be able to display itineraries, and narratives of historical events, the map must be able to support having a path between specific locations layered on it.	Jõao Rosário's thesis [25]
Multiple map layers	The software developed must allow there to be multiple map layers displayed either one at a time or overlapping them. An example can be observed in the PASEV prototype [22]. This is also required to show the locations and the routes on the map.	Analysis of the PASEV prototype [22] and Jõao Rosário's thesis [25].
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 [25].
Individual user accounts and credentials for back-office	This is necessary to ensure only specific users can alter the system's information.	Analysis of Jõao Rosário's thesis and other requirements [25]

Table 2. Domain mandatory non-functional requirements

Requirement	Description	Source
Usability	As mentioned in Section 2.1, the users of this software are not required to have expertise in technology. Hence the developed platforms must be user friendly and easy to use after a determined period for adaption.	Analysis of target users and Jõao Rosário's thesis [25].
Security	Since the domain includes back-office to populate the system data, it must be able to verify the credentials of those who wish to manipulate the system data. Furthermore, there may be software such as the PASEV platform that also require user authentication for the front-office platform.	Jõao Rosário's thesis [25].
Persistence	This should be acquired through the implementation of a relational database capable of storing all the required data (specified in Section 6). This database should ensure concurrency and avoids all information being lost if the server fails.	Jõao Rosário's thesis [25].
Performance and scalability	This requirement will vary for different systems in the domain. However, as it's targeted for end-users a very slow software response is not acceptable.	Conference with project advisors.
Data integrity	As the software in the domain can have multiple users accessing and altering data at the same time, data integrity is important, to avoid synchronization conflicts. In addition, no user would like to invest time adding data to the system to later find out there was a problem.	Conference with project advisors.

5.2 Optional Requirements

Tables 3 and 4 represent the optional requirements of the whole domain. This means they contain components that may be desirable for *some* software developed in the domain but may not be beneficial for others.

Table 3. Domain optional functional requirements

Requirement	Description	Source
Display user location	This can improve the user experience as it helps the user identify the surrounding points of interest.	Jõao Rosário's thesis [25].
Audio recording	The software may wish to allow the client users to add voice notes or other sound recordings to certain locations in their profile (supported by user authentication).	Jõao Rosário's thesis [25].
Individual user accounts and credentials for front-office	This is necessary if allowing the customization of data such as adding personal sound recordings is desirable.	Analysis of Jõao Rosário's thesis and other requirements [25]
360 multimedia display	This can be achieved through access to a gyroscope (if available on the technology running the software) or dragging the multimedia presented on the software (an example would be Google Maps Street view).	Consultation with Bruno Ponte and André Ferreira.
QR Scanner	The software may require support for scanning and interpreting QRCodes to access location information.	Consultation with Bruno Ponte and André Ferreira.
Storytelling	If the software has access to the user's location, a requirement may be to display the data of said location automatically as the user travels between places/ on a predefined route.	Consultation with Bruno Ponte and André Ferreira.
Polygonal area representation for a location	In some cases, there may be a desire to represent certain locations as polygonal areas in the map rather than only pinpoints. This is not to say the pinpoint locations will cease to exist on systems that implement this requirement. Rather, both options should be available.	Jõao Rosário's thesis [25].
Filter/search different types of content	The system might require users to be able to narrow down displayed information with the help of the software's search tools. These may have different formats but the user must be able to apply a filter for different types of content. In the PASEV example, this includes searching for a particular event, or itinerary.	Analysis of the PASEV prototype [22] and the Spanish historical soundscapes project [26], along with Jõao Rosário's thesis [25].
Spatial search mechanism	This search mechanism limits the displayed data and should be possible through selecting or typing an input.	Analysis of the PASEV prototype [22] and the Spanish historical soundscapes project [26], along with Jõao Rosário's thesis [25].
Multiple cities	When implementing software, the project it belongs to might desire different cities to be available each with their set of data such as the locations, the events, etc.	The Spanish historical soundscapes project [26]

Table 4. Domain optional non-functional requirements

Requirement	Description	Source
Adaptability	Some solutions may desire/require (as is the case for the PASEV project) the software to adapt to the type of device executing it (e.g. laptop, phone, etc.).	Jõao Rosário's thesis [25].
Cost	Cost reduction is usually a desired requirement to be considered. This requirement depends fully on the organization responsible for the system being developed.	Conference with project advisors.
Reliability and Availability	This requirement is desirable for almost all software in any domain. However, it often comes with a significant cost associated. The software in this domain are not considered to be essential and would likely not cause too many problems if it was inaccessible. The extent to which an organization is willing to go to ensure this requirement is completely optional.	Conference with project advisors.

5.3 Use Case Diagram

In this subsection, all functional requirements are presented through a use case diagram. These use cases were determined through extensive examination of the currently developed projects in the domain [21,22,26], an analysis of Rosário's thesis [25], consultation with both André Ferreira and Bruno Ponte (FCT-UNL students developing their thesis on extending the PASEV platform), and deliberation with the two project advisors. The software used to create this diagram was the StarUML software modeler [17]. If the reader is not confident in interpreting a use case diagram, please consult Section A.1.

The use case diagram is used to illustrate the functional requirements as an auxiliary to the tables presented in the sub-sections above. This is because, despite not depicting the non-functional requirements, it distinguishes between mandatory and optional requirements through the use of stereotypes, it helps illustrate the dependencies between use-cases and subsequently between functional requirements. In addition, it helps when developing the models in Section 6.

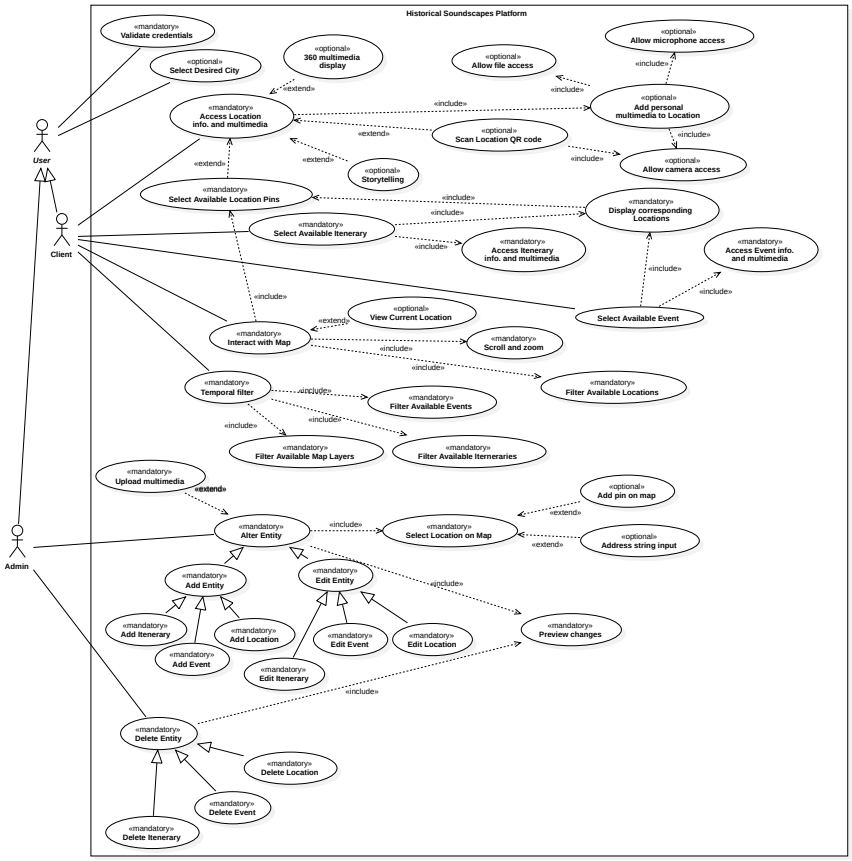

Fig. 3. Use case diagram of the domain model

6 Domain Modeling

The author's "understanding of the domain entities (objects) and their relationships"[11] is described in Section 6.1 through a class diagram. The "end-user's perspective of the capabilities of the"[11] software in the domain for both the front-office and back-office platforms are also presented through feature models in Section 6.2.

Domain modeling is the Section that raised most issues as is explained in Section 7. Any decisions made that may have lead to conflict are thoroughly described in that section. Hence, all that is required from the reader, in this part, is the ability to understand the notations of both class diagrams and feature models. If the reader requires more information on how to interpret these diagrams please consult sections A.2 and A.3 respectively.

If an update on the requirements is done, there must be an attempt to trace all the entities and their relations, as well as the features, which were dependent on the aforementioned requirements.

6.1 Conceptual Modeling

In this section, the conceptual model for systems in this domain is presented through a class diagram. This diagram is used, rather than an entity relationship diagram (ERD), because it has a more direct application when it comes to programming the software, and its interpretation is not much more complicated than ERDs for readers unfamiliar with either diagram [12]. The software used to create this diagram was the StarUML software modeler [17]. The concepts which might cause some confusion for the reader are defined in Section 3. For a more in-depth understanding of the reasoning behind each component of the diagram please refer to Section 7.

In figure 4, the entity *Type* is an enumerator. Its values are not listed on the diagram as it would make the diagram too dense. Instead, they are listed in the Appendix, Section A.4 [1].

OCL constraint :

```
1 context Event inv checkEventLocation:
2     self.eventPlace -> notEmpty() implies self.eventItinerary
          ->isEmpty()
3     and
4     self.eventItinerary -> notEmpty() implies self.eventPlace
          ->isEmpty()
```

The OCL constraint above refers to the class diagram in figure 4 and states that an instance of an *Event* is either linked to an *Itinerary* (which in turn represents an ordered set of *Place* entities) **or** to a single *Place*. It cannot have a relationship with both.

[1] The event types listed are the main observable from the currently developed projects in the domain. It is not a static list and is subject to change whenever there is an update in a project of the domain.

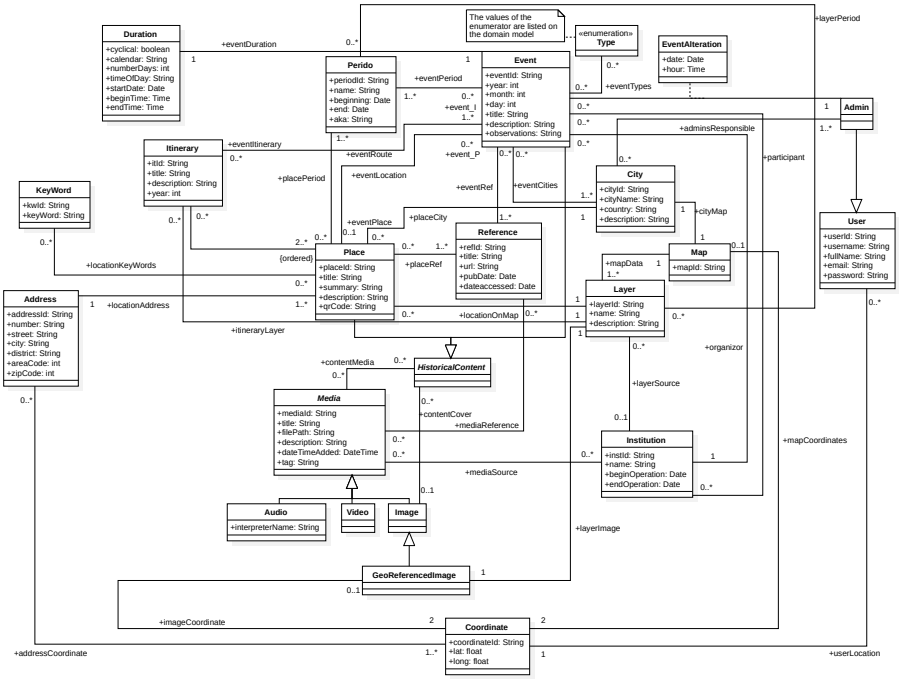

Fig. 4. Class diagram of the domain model

6.2 Feature Modeling

This section, presents a complementary component to the diagram presented on Section 6.1. The previous section models which entities are present in software in the domain, and how they should relate, whereas this section presents client visible components for software in the domain. The features extracted in sections 4.1 and 4.2 help determine some of the features presented in this section. However, as the domain analysis attempts to model all possible software in the domain, not limited to the two current software developed, not all commonalities from Section 4.1 will be considered as commonalities in the feature models, and some new features will be introduced to both the commonalities and the variabilities. These new features are the product of analyzing the requirements and use case diagram, along with contemplation of the domain, supported by consultation with the two project advisors.

The new commonalities introduced, despite not being present in both of the currently developed software, are considered commonalities as they are compulsory features in order to achieve the mandatory functional requirements described in Section 5.1. This means, despite the domain analysis being based on the PASEV project [22], and the Spanish historical soundscapes project [26], the implemented software may not fully fulfill the requirements. This in turn sug-

gests that the developers should likely reanalyze their platforms and assess the modifications required. It does not however make the domain analysis any less correct. It means it is effective to the extent of modeling concepts the developers had not yet considered. If this analysis had been available from the beginning when the developers were planning their platforms, they would most likely not miss important features or requirements. The description of conflicts that arose when developing this diagram and how they were approached is available in Section 7

Figure 5 and figure 6 are the feature models for the back-office platform and the front-office platform respectively. The interpretation of figure 5 is facilitated by the legend presented in the image, and the interpretation of figure 6 is facilitated by the legend in figure 7. The software used to create these models was the *FeatureIDE* eclipse plug-in [15].

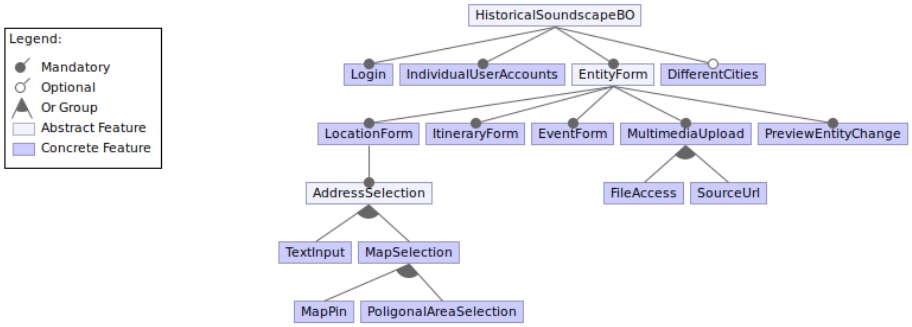

Fig. 5. Back-office feature model of the domain model

Figure 7 not only contains the legend of figure 6 but also contains the front-office platform feature constraints. The constraints explanations are as follows:

1. Adding personal multimedia to a location requires the system to have file access.
2. Adding personal multimedia to an event requires the system to have file access.
3. If the client user wishes to view its current location on the map, the system requires access to the user's geolocation.
4. If the software wishes to implement storytelling and display multimedia relevant to the user's location automatically, the system requires access to the user's geolocation.
5. For there to be individual user accounts in the system, the platform must also allow the client users to log in and vice versa.
6. For a user to be able to add personal multimedia to a location, the system must know which user the multimedia belongs to. Hence, it requires individual user accounts.
7. Following the same principle as above, adding personal multimedia to an event requires individual user accounts.

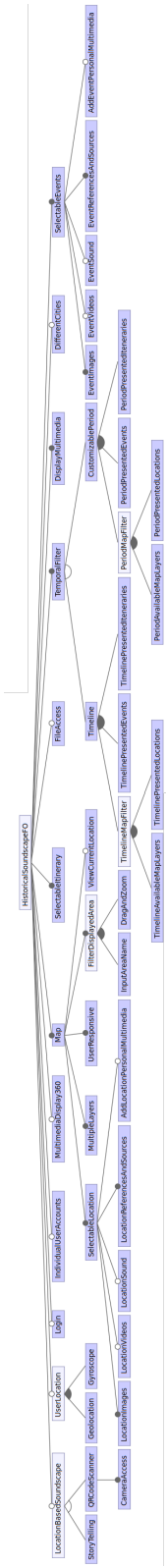

Fig. 6. Front-office feature model of the domain model

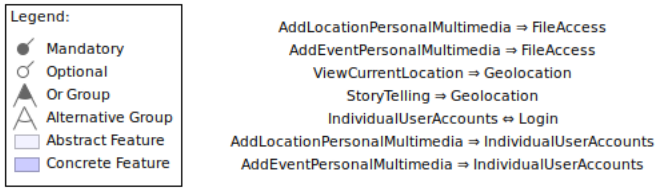

Fig. 7. Legend and logical constraints of the front-office feature model in figure 6

The reader may notice some features very similar to one another in the same feature model (e.g. TimelineMapFilter and PeriodMapFilter). This is because of how feature diagrams work. Each feature can only have one *father* feature. This leads to practically identical features, with minor differences, to repeat child features required on both cases (e.g. the Timeline feature and CustomizablePeriod feature have the same goals yet their user interface is different). This has no major negative impact on the domain model except overpopulating a diagram with repetitive data. Unfortunately, it is an unavoidable consequence of using a feature diagram.

In order to have more information about each feature, the reader is required to view the feature models on the Eclipse IDE software with a plug-in code extended by the author. A detailed description of each feature is available when viewing the projects listed in Section 6.2, with the plug-in extension, by hovering over each feature.

Viewing the feature models with a FeatureIDE extension

Part of the author's investigation was to extend the plug-in to allow adding multimedia to the diagrams. It was required so there can be examples of the desired features present in the models. This was achieved by allowing URLs to be linked to each feature.

In order to view the diagrams presented in this section with the corresponding examples, please consult the manual [4] available in the git repository of the author's investigation: https://github.com/marianaBonito/APDC-Investigation-HistoricalSoundscapesCaseOfEvora-DomainAnalysis.

The back-office feature model diagram project is present in the git repository: https://github.com/marianaBonito/HistoricalSoundscapesFeatureModel-BO.

The front-office feature model diagram project is present in the git repository: https://github.com/marianaBonito/HistoricalSoundscapesFeatureModel-FO.

7 Domain Model Issues

When developing the content for the previous sections in the domain model, several issues/conflicts where raised. In this section, these issues are detailed, along with their alternatives (if applicable), and the chosen course of action is

also described. This section is present to facilitate modifications to the domain model that may be required as more domain information is made available.

Issues with Section *5 Requirements* - As mentioned before, the lack of available software in the domain is the main limitation of this domain analysis. The principal struggles this caused was, firstly, being able to determine the functional and non-functional requirements. Secondly, being able to distinguish between mandatory and optional requirements.

The extractions of the requirements have their sources presented in the last column of the corresponding line in Section 5.

The differentiation between mandatory and optional requirements required a more thorough understanding of the domain. Despite the author not being a domain expert, with the help of all the available documentation, the papers read to aid the domain knowledge, and the assistance of both project advisors, the mandatory requirements for any software in the domain were identified. The remaining were classified as optional.

Issues with Section *6.1 Conceptual Modeling* - When developing the class diagram for the Section in question, there where two main difficulties faced:

1. Standardizing the entities and attributes required in the database. These had to be extrapolated from the use case diagram in Section 5.3 and the forms available from the PASEV project [24].
   The entity that was most difficult to model was the Event as the forms available were not consistent. Table 5 presents an example of three inputs for the same data column of the forms filled in by the PASEV experts. The column on the left contains the Portuguese version exactly as registered in the form. The right column contains the English translation. The first row is the title of what the PASEV experts were requested to insert relevant to the event they were populating.

   Table 5. Three different inputs for the same data column [24]

Duração do evento (cíclico/ocasional)	**Event duration (cyclical/occasional)**
Ocasional (Noite: das 21h às 23h)	Occasional (Night: from 21h to 23h)
Ocasional (Um dia e noite)	Occasional (A day and night)
Ocasional (4 récitas em 4 noites)	Occasional (4 recites in 4 nights)

   Table 5 only displays a small sample of a single data column in the event-form, however, it is clear how adapting to the data input might be difficult. There was a simple option of allowing each column of the event-form to be a String attribute of the entity Event. This would facilitate the transition between the current database structure to the general database structure. However, one cannot be sure that the Spanish historical soundscapes project has the same data available and in the same structure. In addition, when wishing to analyze or manipulate the data, this would be much more difficult. Hence the adopted method attempted at generalizing the information as much as possible, trying to extract some attributes, not necessarily Strings.

This lead to it being clear that the duration, as well as the references, the period, and other columns should be split into different attributes. In order to keep the class diagram simple and avoid repetition of some attributes, entities that would not have been present otherwise were generated (e.g. Duration entity, Reference entity, Type entity, etc.). As with every situation, there is always a trade-off between generalization and specialization. Nevertheless, despite not entertaining every single option, the selected approach is the most adequate for a domain model as it needs to accommodate more than just the PASEV project.

The Media entity also required some pondering with its attributes. As seen in figure 4 in Section 6.1 the Media entity has a filePath String attribute but not a blob attribute to save the actual media data. This approach was chosen as keeping the blob files would add excessive weight to a relational database. The suggested approach is to save the blob file in a separate server (e.g. a cloud server) and simply saving the path to that media in the database.

There are also other entities present in the diagram which might not initially be obvious (e.g. User entity). These were created due to the requirements obtained in Section 5. One of these entities is the City entity. This class was created as there is an optional requirement in Section 5.2 which states the software may need to allow multiple cities to be integrated into the system, each with its own set of data. Hence, the associations with Map, Events, and Places.

2. Stipulating the relationship between entities was hard to make abstract as the two software do not have the same structure. In order to select the required relations, the purpose of each entity had to be very clear. There were three types of relationships that had to be represented:
   (a) Certain entities, as explained on the previous point, were created to keep the diagram easy to read or to avoid attribute repetition (e.g. Media entity, Coordinate entity, Institution entity). In these cases, the relationship between the entities is clear. Either inheritance (e.g. Image - Media relation), one-to-one relationships when an entity is meant as an attribute of another (e.g. Duration - Event relation), one-to-many relationship when a set of a specific entity is an attribute of another entity (e.g. Type - Event relation), or many-to-many if a set of a particular entity can be an attribute to more than one instance of another entity (**Note:** not all sets need to be equal)(e.g. KeyWord - Place relation).
   (b) The second type of relationships that were addressed were the ones very clear from the domain knowledge. These include the ordered set between an Itinerary and a place, the collection of different types of media for the abstract entity HistoricalContent (as both a Place and an Event need to be able to have media associated with it), etc.
   (c) The final kind of relationships that where added where the ones that required more consideration. In this case, all of these relationships are listed below with their justification as it may be the ones the reader does not agree with initially.

- **Event - Itinerary and Event - Place relationships:** To understand these relationships and the corresponding OCL constraint it is important to understand that an Itinerary serves as an encapsulation of an ordered set of Places. This justifies the ability to add a description, a title, and a year it refers to (if the Event it regards is cyclical and the Itinerary was not the same every year). However, there is no sense in having an Itinerary with only one Place as it would be a set with one object (hence the relationship between Itinerary and Place forcing a minimum of two Places per Itinerary). This being said, Events usually occur in one or more places. If it only happened in one Place (e.g. a ball) the Event will be associated with a Place. However, if it took place on multiple locations (e.g. a procession) it requires the route taken to be evident. Thus the Event will have a relationship with an Itinerary.
  An Event can have multiple Itineraries as the route might have changed during the years the Event took place. In addition, an Itinerary can have multiple Events as the title, the description and the years are optional and different events may use the same route.
  An Event can only have one Place because, as previously mentioned, if it has more than one location, it requires an Itinerary. On the other hand, a Place can be linked to multiple Events as different Events can take place in the same location (usually not at the same time).
- **Itinerary - Layer relationship:** The comprehension of this relationship requires some knowledge of how maps and layers work. As mentioned in Section 5.1, one of the mandatory requirements is to be able to *display routes on the map* which represent Itineraries. To do so, an Itinerary is associated with a specific line which will then be placed over the map. Hence, each Itinerary has a relationship with a Layer. However, the Itineraries can be all on the same Layer, or there can be a different Layer for each Itinerary. This depends fully on the preferred implementation by the developer.
- **HistoricalContent - Event and HistoricalContent - Place relationships:** The HistoricalContent entity is an abstract entity created solely to avoid repeating the relationships between the Media and the Place, the Media and the Event, the cover image and the Place, and the cover image and the Event. These two relationships are required for both the Place and the Event because the PASEV project associates the media with a Place, whereas the Spanish historical soundscapes project associates the media to the Events. There was an option of only allowing either one or the other, and suggesting one of the software should change. However, allowing the media to be linked with both Events and Places makes sense as there can be media of a Place that does not belong to any Event associated with said location, just as there may be media that apply to an Event but not particularly to any of its Places. Hence, by having these relation-

ships only displayed with the HistoricalContent entity and then be inherited by the Event and Place entities, keeps the diagram clean while still representing all relationships described.

- **Address - Coordinate relationship:** The Address entity has a one-or-more relationship with the entity Coordinate as there is an optional requirement in Section 5.2 which requests allowing Place locations to be determined through a polygonal area in the map.
- **Layer - GeoReferencedImage relationship:** This relationship, again, requires knowledge on how map Layers work. To simplify, each layer is data and is associated with an image (as shown in Section 3). In order to know where on the map that image needs to be placed, it has coordinates associated with it. This image with the respective coordinates is designated as a GeoReferencedImage. Hence the one-to-one relationship between the GeoReferencedImage entity and the Layer entity.
- **Layer - Period relationship:** When creating software in this domain, the developer may wish to associate a Period to a Layer as there can be Layers which are maps of the cities disposition in the past.
- **Admin - Event relationships:** This relationship is present so that whenever an Event is created or altered, the members of the responsible organization know exactly when the event was created/altered and by whom.
  Each event can only be created by one user, however, it can be altered by any user which has access to the back-office. In addition, each Admin user can create or alter multiple events. Thus the many-to-many relationship. Nevertheless, there is a minimum of one Admin user per Event as the Event needs to have been created by an Admin user.
- **Admin - City relationships:** The obvious City relationships were mentioned previously. However, the Admin - City relationship may require a better justification. This relationship was created due to Admin users supposedly being specialists in the historical-cultural events and soundscape of an area the system focuses on. This being said, if the system covers many different Cities, each Admin may not be a specialist in all of them. Thus, the responsible organization may wish to limit the access of each Admin solely to the Cities the user is a specialist on. Hence the many-to-many cities. Nevertheless, there is always at least one Admin per City as otherwise the data of the City could not be updated and it would be pointless to have the City registered in the system.

Issues with Section *6.2 Feature Modeling* - The feature models were mainly based off of the two diagrams previously developed and analysis of the requirements. Hence, all decisions relevant to the feature models are a product

of the conflicts previously resolved. This is to say, once the issues where resolved in previous parts of the domain analysis, these models only had to illustrate the choices already made. The only two important decisions that were made when developing the feature models were:

1. **The choice to only represent functional requirements:** Representing the non-functional requirements would be acceptable in these models. However, this decision was done due to not wanting the models unnecessarily comple.g. In addition, the representation of non-functional requirements is not the usual application of feature models.
2. **The selected level of abstraction to use for each feature:** Clearly, not all features have the same level of abstraction on the feature model. Each feature was expanded to the highest level possible while still allowing a large range of implementation options. Not all features were expanded to the same level because some are not as critical for the domain and permit various approaches.

8 Conclusion

This document presented an initial domain model. This domain model focuses on the requirements and their understanding along with solutions for said requirements through the use of use-case diagrams, class diagrams, and feature models. It defines terms necessary for the proper understanding of the domain and justifies the main decisions made which may impact future work. The evaluation of this domain model and future work acknowledgment is part of another paper by the author.

This domain model is not final as it requires a constant update whenever new information about the domain is made available. It has clear links between information and their sources to allow these updates to be more efficient.

The main challenge faced was the lack of available software to base the product-line model on. Nevertheless, a thorough analysis of said software allowed for a good understanding of what is desired by systems in the domain and hopefully was a good base for modeling the whole domain.

References

1. ArcMap - What is a layer? (2019). Retrieved from https://desktop.arcgis.com/en/arcmap/10.3/map/working-with-layers/what-is-a-layer-.htm
2. ArcMap -Essential layer vocabulary (2019). Retrieved from https://desktop.arcgis.com/en/arcmap/10.3/map/working-with-layers/what-is-a-layer-.htm
3. Bonito, M. (2020). Domain Analysis and Geographic Context of Historical Soundscapes: The case of Évora (Tech.). Retrieved from https://github.com/marianaBonito/APDC-Investigation-HistoricalSoundscapesCaseOfEvora-DomainAnalysis/blob/master/Final_Report.pdf
4. Bonito, M. (2020). FeatureIDE URL Plugin Extension - Manual
5. Caliper Mapping and Transportation Glossary - What is a layer? (2020). Retrieved from https://www.caliper.com/glossary/what-is-a-map-layer.htm
6. DeChampeaux, D., Faure, P., & Lea, D. (1994). Chapter 12: The Analysis Process and Chapter 13: Domain Analysis. In Object-oriented system development. Reading, MA: Addison-Wesley.
7. Dhamelia, M., & Dalvi, G. (2019). Cultural Domain Analysis for Soundscape Assessment. (A. Chakrabarti, Ed.). Springer Nature Singapore.
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. Hjørland, Birger. 2017. "Domain Analysis." Knowledge Organization 44(6): 436-464. 151 references.
11. 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.
12. Kaushik, N. (2019, January 31). Difference Between ERD and Class Diagram. Retrieved from http://www.differencebetween.net/technology/software-technology/difference-between-erd-and-class-diagram/
13. Laganière, R. (2001). 4.1 Domain Analysis. Retrieved from http://www.site.uottawa.ca/ laganier/seg2500/domain
14. Lee, J., Hicks, D., Henriksen, D., Mishra, P., & the Deep-Play Research Group. (2015). Historical Soundscapes for Creative Synthesis.
15. Leich, T., & Saake, G. (2005).
16. 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
17. MKLabs Co.,Ltd. (2020).
18. Nishadha. (2019, July 2). Use Case Diagram Relationships Explained with Examples. Retrieved from https://creately.com/blog/diagrams/use-case-diagram-relationships/
19. Nishadha. (2019, May 9). Use Case Diagram Tutorial (Guide with Examples). Retrieved from https://creately.com/blog/diagrams/use-case-diagram-tutorial/
20. Oxford University Press. (2019). period. Lexico.com. Retrieved from https://www.lexico.com/definition/period
21. PASEV • Patrimonialization of Évora's Soundscape • 1540-1910. (2019). Retrieved from https://pasev.hcommons.org/

22. Plataforma PASEV. (2019). Retrieved from https://pasev.di.fct.unl.pt/#brand
23. Rabiser, R., Schmid, K., Eichelberger, H., Vierhauser, M., Guinea, S., & Grünbacher, P. (2019). A domain analysis of resource and requirements monitoring: Towards a comprehensive model of the software monitoring domain (Vol. 111, pp. 86–109). Elsevier B.V. Retrieved from https://www.sciencedirect.com/science/article/abs/pii/S0950584919300606?via=ihub
24. Rosário, J. (2019). [Historical events form]. Unpublished raw data.
25. 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.
26. Ruiz Jiménez, J., & Lizarán Rus, I. J. (2015). Paisajes sonoros históricos (c.1200-c.1800). Retrieved from http://www.historicalsoundscapes.com/
27. Shapere, Dudley. 1977. "Scientific Theories and their Domains." In The Structure of Scientific Theories, ed. Frederick Suppe. 2nd ed. Urbana: University of Illinois Press, 518-65.
28. Sterne, J. (2013). Soundscape, Landscape, Escape. De Gruyter. Retrieved from https://www.degruyter.com/downloadpdf/books/9783839421796/9783839421796-010/9783839421796-010.pdf
29. Visual Paradigm. (2020). UML Class Diagram Tutorial. Retrieved from https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/
30. Vylder, T. D. (2011, June 29). PDF. Belgium.
31. 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

A Appendix

A.1 Interpret Use Case Diagrams

In this section, only the concepts relevant to interpreting the diagram in figure 3 Section 5.3 are explained. For more detailed information please refer to https://creately.com/blog/diagrams/use-case-diagram-tutorial/.

Main Terms

Use case: The use cases represent the functions provided by the system. In the diagram, they are shown as an oval with the function descriptive name in the center

Actor: Actors are entities that do not belong to the system but perform a role in it. They are represented by stickman on the diagram. In this case, the actors are only Users.

System: The system defines the scope of use cases. In the diagram in figure 3, it is the large rectangle surrounding all use cases with the system name at the top "Historical Soundscape Platform".

Important rules:

1. All use cases must be in the system rectangle
2. All actors must be outside the system rectangle

Relationships

Actor Relationships

There are only two types of relationships actors can have:

1. Actor - Use case relationship: This relationship represented by a single black line means an entity playing the role of that particular actor can do the function associated with the use case.
2. Actor - Actor relationship: This relationship (present in the diagram) is a *Generalization* relationship. It means "an actor can inherit the role of another actor"[18]. It is represented with an arrow with a wide empty head pointing from the specification to the generalization. In the diagram's example, the User is the generalization and there are two specializations: the Admin and the Client. The Admin and the Client can do everything the User can do but not vice versa. In addition, the User actor is in italics. This means the User actor is abstract and cannot have any instances that are not either Admin or Client.

Use case Relationships

1. Generalization: This relationship is similar to the generalization relationship between actors. It is also represented using an arrow with a wide empty head pointing from the specification to the generalization. The functionalities of the generalization are inherited by the specialization but not vice versa.
2. Include: This relationship is represented using a dotted arrow with the word *include* beside it. The arrow points from the base use case to the included use case. This relationship isn't optional. The base use case cannot occur without the included use case. The included use case is usually used to reuse common functionalities. However, the diagram also uses them to emphasize certain functions that may not be obvious but are worth emphasizing.
3. Extend: This relationship is represented using a dotted arrow with the word *extend* beside it. The arrow points from the extending use case to the base. It means the extending use case cannot happen without the base but the base can happen without the extending use case. Hence, it is optional and activated with a trigger. The extending use case serves to add functionality to the base use case.

A.2 Interpret Class Diagrams

In this section, only the concepts relevant to interpreting the diagram in figure 4 Section 6.1 are explained. For more detailed information please refer to https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/.

Class definition

A class can also be referred to as an entity. A class represents a type of object present in the system. There can be as many instances of a class as desired. A class is made of attributes. Attributes are the components of the class and can have many different types. The attribute types present in this case are String (array of characters), Date, int (integer), Boolean (true or false), Time, DateTime (Date and Time), and float (decimal point with a 32-bit precision).

In class diagrams, classes are represented using a rectangle with three sections. In the top Section and in bold is the name of the class. If this name is in italic it means there cannot be an instance of that class that is not one of the specialization classes (see generalization relationships for more detail). The middle Section contains the class attributes followed by ":" and the type of the attribute. The last Section is usually where the operations for that class are listed but is irrelevant for this case.

Relationships

There are many different types of relationships but only the ones present in the diagram in figure 4 are listed below:

1. Association: These relationships are represented with a solid line between two classes. They will have an association label in the middle and will have a cardinality on either end. They might also have role names on either end which represent how each class recognizes the other but these are only used in the Event - Itinerary association and the Event - Place association as they are needed for the OCL constraints. Cardinalities represent how many instances of a given class are associated with the other. The cardinalities used are as follows:
   - 1 - Exactly one instance
   - 0..1 - Zero or one instance
   - x..* - x or more instances, x is an integer value greater than or equal to 0

   There is also the special association between the Itinerary and Place entities which indicates the Itinerary has an ordered set of Places.
   Take the association between the Map and the Layer entities as an example of how to read the association cardinalities. The Map has one or more Layers associated with it. However, each Layer is only associated with one Map.
   Association classes are a special type of class. An example is the EventAlteration class linked by a dotted line to the association between the Event and the Admin entities. An instance of this class is created every time a new association is formed between an instance of an Event and an instance of an Admin.
2. Inheritance: These relationships are represented using an arrow with a wide empty head pointing from the specification to the generalization. The specification inherits the attributes, relationships, and operations of the generalization but not vice versa.

A.3 Interpret Feature Models

In this section, only the concepts relevant to interpreting the diagrams in figures 5 and 6 are explained. For more detailed information please refer to http://msdl.cs.mcgill.ca/people/hv/teaching/MSBDesign/201011/projects/Thomas.DeVylder/report/FeatureModelling%20-%20Thomas%20De%20Vylder%20%28final%29.pdf.

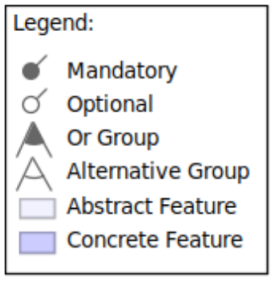

Fig. 8. Legend of feature models in the domain model

The legend in figure 8 is present to aid the understanding of the explanations below.

Feature definition

Simply put, a feature is a user-visible component of the system. It is represented with the dark and light rectangles and their name is in the center of said rectangles. The diagrams are read from the top down and as the level increases (further down) the level of abstraction decreases.

The lighter features are abstract. This means these features are a generalization of other features and cannot be instanced (is a concept rather than an actual user-visible component). On the other hand, the darker features are concrete and when developing software the developers can instance that particular feature and it will be visible to the user.

Important: The feature at the top represents the software being developed, hence, it is always included. As the diagram is read from the top down, if a parent feature is not included in the software being developed, its children are not even considered as its branch is completely discarded.

Relationships

There are four types of relationships between features and all are used in the current domain model. Their representation is evident in the legend in figure 8. The relationships are:

1. Mandatory: As the name suggests, mandatory features are non-optional. This means, if its parent feature is included, the mandatory feature is also included.
2. Optional: Again, the name of the relationship is self-explanatory. An optional feature is one that the developer can choose whether or not to implement.
3. Or Group: This relationship incorporates all children of a specific feature. It means that if the parent feature is included, one or more of its children can be included.
4. Alternative Group: This relationship also incorporates all children of a specific feature. It means that if the parent feature is included, one and only one of its children can be included.

When using a feature model to select which features to include, the configuration is deemed valid when all selected and mandatory features are resolved.

A.4 Type entity enumerator values

In this section, the event types are listed. These event types are in English and are a suggestion for the systems in the domain. It does not mean the systems cannot have other types registered. They were selected as they fit most of the event types available in both PASEV and the Spanish historical soundscapes projects [24,26]

- Civic Demonstrations
- Religious Manifestations
- Military Demonstrations

– Popular Manifestations
– Games and Festivities
– Community Celebrations
– Announcements
– Other Religious Events
– Other Military Events
– Commercial Activities
– Natural Disasters
– Entertainment Events