Development of Digital Libraries with Software Product Line Engineering

Delfina Ramos-Vidal, Alejandro Cortiñas, Miguel R. Luaces, Oscar Pedreira* and Angeles Saavedra-Places

Universidade da Coruña, Centro de Investigación CITIC, Laboratorio de Bases de Datos, Facultade de Informática, Elviña, 15071 A Coruña, Spain
E-mail: delfina.ramos@udc.es; alejandro.cortinas@udc.es; luaces@udc.es; oscar.pedreira@udc.es; asplaces@udc.es
**Corresponding Author*

Received 31 March 2021; Accepted 22 July 2021;
Publication 22 October 2021

Abstract

Digital Libraries have become popular nowadays since important libraries all over the world started distributing their collections online, properly classified, and, in many cases, with access to the digital version of the resource. These programs have been beneficial to the general population as well as research groups in fields such as language and literature. Nonetheless, since their creation is a time-consuming and costly process, small organizations are forced to rely on obsolete or poorly designed software. However, most of the features, including the data model, are shared by this type of system, with minor variations depending on the type of resources to be handled. This article presents a Software Product Line (SPL) for the semi-automatic generation of Digital Libraries (DL). Our SPL allows developers to specify which DL features are required, which will define the data model variation and the generated source code. The specification is then transformed into a fully functional DL application with the specified features that is ready for deployment. We present the feature model, the SPL implementation, and a case study on three sample projects that enabled us to evaluate the resulting software, with a focus on development effort savings.

Journal of Web Engineering, Vol. 20_7, 2017–2058.
doi: 10.13052/jwe1540-9589.2072

Keywords: Software product lines engineering, digital libraries, generation engine.

1 Introduction

Over the last decades, there have been many efforts to create *digital libraries* (DL) to manage and upload digital resources (and their bibliographic information) online, either obtained from digitization processes of literature and cultural heritage or native digital resources. Non-profit organizations that own valuable books, journals, and pamphlets have seen DLs as a means to preserve and disseminate their cultural heritage. Research groups in Digital Humanities have adopted DLs as a basis for sharing resources and promoting research in a way that would be unfeasible without their navigation and search capabilities. Also, *digital libraries* with a web-based interface have opened a huge and valuable collection of cultural resources to the general public.

In many ways, developing a DL is a difficult task. Experts in both Software Engineering and Humanities must find common ground to understand each other and come to a mutually agreeable design and implementation decision. The Database Lab, the authors' research group, has been involved in several successful multi-disciplinary projects regarding *digital libraries*, in collaboration with experts in different Humanities areas (Language and literature, Arts, Pedagogy, etc) [2, 14]. As a result of years of experience developing DLs, we've concluded that creating a DL from scratch is far too expensive, both in terms of time and money, for many non-profit organizations and research groups. However, there is a set of characteristics that DLs always have in common [18]. As a result, a shared core of assets can be defined for their development. Furthermore, even though each DL has its own data model and characteristics, they can be analyzed and defined using a formal language. Therefore, we consider the domain of DLs is appropriate for the application of *software product lines engineering* (SPLE).

In SPLE, the goal is not to design and implement a single software system but a platform (*software product line*, SPL) to automatically generate a family of similar software systems that share a set of core assets and which have some features in common while they can vary in others. The features that can be present in each product of the family are modelled in a feature model. Given a selection of the features to be included in a particular product, the SPL processes the core assets to generate the specified product. SPLE allows for systematic reuse of software artefacts and implies a significant reduction in the development effort.

In this article, we present a *software product line* for the development of *digital libraries*, following the steps of the SPLE methodology presented in [5]. A preliminary version of this article was published as a conference paper [16]. This work extends that preliminary publication, which only covered the analysis phase: we present the design and implementation of the SPL (architecture, core assets, and other design decisions), and we present the results of an evaluation of the SPL after generating and analyzing three DLs based on real scenarios. The results show that following the SPLE approach allows to produce different DLs with a significantly reduced cost. Our analysis of the DL domain shows that DLs present a degree of variability that makes them a suitable application domain for digital libraries. Also, our tests on the use of the SPL we have developed show the SPL can be used to generate DLs that can be considered adequate for different cases, with very little custom development effort on the products generated automatically.

The rest of the article is structured as follows: In Section 2 we review background and related work. In Section 3 we present the SPL, including an analysis of the architecture and main features of a DL and the feature model. Section 4 details the implementation of the tool that allows the semi-automatic generation of the web-based DLs, transforming a feature specification into the generated code. Section 5 presents an evaluation on three sample applications. Finally, Section 6 presents the conclusions and lines for future work.

2 Background and Related Work

2.1 Digital Libraries

According to the Alexandrian principle, *a librarian must increase the stock of his library*. Following this principle, even the largest libraries are doubling in size every 16 to 20 years [19], an increase that cannot be supported indefinitely. The need for new architectures capable of storing all the knowledge produced by society is more imperative than ever.

A *digital library* is not simply a "digitized library". *Digital libraries* are defined as focused collections of digital objects, including text, video, and audio, along with methods for access and retrieval, and for selection, organization, and maintenance [19]. This technology focuses on finding new ways of dealing with knowledge: preserving, collecting, organizing, propagating, and accessing it. Besides, *digital libraries* give significant weight to the user (who accesses and retrieves information) and the librarian (in

charge of selection, organization, and maintenance). The distinction between user and librarian is not as clear when it comes to *digital libraries*, but it is important anyway to distinguish between the different roles. For the task of developing a *digital library*, the point of view of libraries and information science should be taken into account, considering the librarian's functions instead of focusing on the technology. Therefore, having access to real experts in the domain is key in this process.

Digital libraries tend to present an opaque appearance: they are usually web pages without any indication of their content or what is behind. On the contrary, physical libraries occupy a physical space that visually reflects the work involved in the collection and organization of all those resources. However, not all websites providing digital objects should be considered a *digital library*. For physical libraries, adding a new book to the shelf is an easy task; likewise, it should be possible for new material to become a first-class member of a *digital library* without any need for manual updating of the structures used for access and retrieval.

During the last decades, numerous *digital libraries* have been produced, whose technology has evolved from the 60s to today. In the rest of this section, we present an analysis that focuses on a selection of *digital libraries* that includes both large ones that mark the current state of the art and smaller ones developed by our research group.

The **Library of Congress**[1] is the largest *digital library* in the world, with millions of resources. It stores not only books or printed material, but also photos, drawings, newspapers, music, maps, and many other kinds of resources. Its web page is updated frequently with new collections and news. It provides a simple search engine to search for simple keywords, but the results can be filtered and sorted by many properties. The user interface is very friendly and modern, and the resources can be downloaded freely in different formats. This *digital library* is used by the congress of the US as its main documentation provider.

The **Project Gutenberg**[2] is a huge DL that provides over 60,000 free eBooks for online reading or downloading, mostly of old works for which US copyright has expired. The DL is maintained by volunteers who digitize the content. Its interface is not very friendly, and its search engine capabilities are limited (it uses external search engines to search within the content of the works), but it provides categorized browsing from more general categories, or

[1] Library of Congress: https://www.loc.gov/

[2] Project Gutenberg: http://www.gutenberg.org/

bookshelves, as the web calls them, to specific domains. The eBooks are also provided in different formats to facilitate the reader's download and posterior reading.

Cervantes Virtual Library[3] is one of the most important *digital libraries* in Spain. Its catalog is classified in different areas, from literature and language to the American library or Galician literature library. They support digital resources of many types: PDF, web, images, and eBooks. They also provide metadata in MARC21 format [9], the physical libraries standard. The search engine allows searching keywords in the content or the metadata, and it allows filtering depending on the kind of document.

Archivo Digital Valle Inclán[4], a smaller author library, was awarded the Hispanic Digital Humanities Award in 2018 for the best tool, infrastructure, or resource of the year. The members of Valle-Inclán Research Group of the University of Santiago de Compostela (GIVIUS) digitized more than 4,500 documents and 80,000 images, grouping books, illustrations, photographs, letters, and interviews among others. Altogether it is a great tool for both researchers and people interested in Valle Inclán's work since it includes a powerful search engine that allows searching by syntagma, pseudonyms, editorials, agents, and many other categories related to Valle.

The project from the University of A Coruña, the **Galician Virtual Library (BVG)**[5], is a smaller regional library with an old fashioned interface that stores a large collection of Galician publications, with digital resources in different formats: text, audio, video, and images. It allows search by content using a strategy based on Bounded Natural Language. The resources stored in the database can have comments made by the users. It also handles editions from the same author, even when written under different pseudonyms.

These *digital libraries* were analyzed to search for similarity and variability in their features. We thus identified that all the considered *digital libraries* manage information related to literary works, such as editions, stand-alone works, collections, or parts of bigger pieces. One main characteristic of *digital libraries* is the fact that they provide access to digital resources, they are not just a catalog of physical entities. Instead, they use different file formats to manage content. However, while most libraries store data about several authors, Cervantes Virtual and Valle-Inclán focus on concrete authors and study their life and work, managing not only their publications but also

[3] Virtual Library Miguel de Cervantes: http://www.cervantesvirtual.com/

[4] Digital Archive Valle Inclán: https://www.archivodigitalvalleinclan.es/publica/principal.htm

[5] Galician Virtual Library: http://bvg.udc.es

papers and studies related to the author's work. Additionally, Valle-Inclán operates with documents that do not fit the structure of a usual library, such as illustrations, letters, and interviews. This set of common elements and variations regarding the characteristics of *digital libraries* reflects that this domain is a good candidate for the application of product lines. All five *digital libraries* presented cover very different topics but it can be seen that they have many functionalities in common after all. This similarity will be the base on which the *Software Product Line* will be built.

For a research group in Humanities, embracing the task of developing a complete DL such as the ones mentioned above is a big challenge due to the cost of these developments. The DLs are designed over a quite complex model that relates all the elements of the particular domain of research, such as authorities, editions, works, organizations, etc. Besides that, handling media files is always hard to achieve adequately. In some cases, having this into account, document management solutions such as Alfresco[6] or Opentext[7] are used instead of developing a proper software. However, these solutions lack the actual functional requirements of a DL, such as advanced search engines focused on the data relevant in the domain of the DL, or using an adequate user interface that provides useful views of this data (i.e., not showing a document as a simple work, but instead show it with all the related elements), but instead provide just a document catalogue with a set of very standard metadata fields.

In summary, most DLs share many features but can be quite different in others. This makes it difficult to develop a general-purpose software for DLs that fulfils the specificities needed in each case. As a result, most DLs are developed as custom projects that require a high investment. Also, in some cases, document management systems are used for this purpose although they do not offer all the functionalities expected in a DL. This article presents an alternative approach for the development of digital libraries based on SPL, so the core assets of any DL can be reused but automatically adapted to the specific features needed in each project.

2.2 Software Product Lines

Software Product Lines Engineering (SPLE) is a field of Software Engineering that focuses on reusing the same software artifacts among different software systems from the same family, trying to reduce development costs

[6] Alfresco: https://www.alfresco.com
[7] Opentext: https://www.opentext.com

and time to market for such systems. That is, when a software development team has to develop a set of products that share most characteristics, both functional and non-functional, instead of carrying out the development of each one of them from scratch, it is more efficient to develop a *Software Product Line (SPL)* and approach the development of all of them together. This way, the base code of all the products is the same, and each of the actual products to sell or finally publish includes the share of code needed, depending on the actual functionalities it must provide. For example, if a company specializes in e-commerce web applications, some clients may need to have "tags" on the products, while others do not. Then, "having tags on the products of the shop" can be a variant feature that can be selected or not for a specific product. The source code implementing this functionality will only be included in the final product's source code if the feature is selected.

Designing and developing an SPL is a costly task but it is still more efficient than developing each product on a one-by-one basis. It is documented that from the third product onwards the extra cost of the SPL is compensated [3]. One additional difficulty is that the decisions made will affect not only a particular product but the whole set of products generated by the SPL. There are different methodologies to follow when designing an SPL. The authors of this work defined one in previous work [5], suitable when the development team has access to both the source code of existing products of the family and experts in the said domain. This methodology divides the definition process into four stages, as seen in Figure 1: *requirements analysis*, *architecture design*, *architecture evaluation*, and *derivation of a specific product*. The *requirements analysis* stage is divided into two steps, *domain analysis*, which consists of analyzing both the requirements and the related work of the domain, and *product planning*, which consists of studying existing applications of the domain or family, including products developed by the team who is designing the SPL as well as external products. *Architecture design* focuses on studying relevant product architectures for the domain and prioritizing components, as well as for deciding the technology stack for the products.

The output of both these stages is the input of the next one, *architecture evaluation*, whose purpose is to validate the models previously defined and to link the different assets defined in them to support the future evolution of the SPL. The last stage is the *derivation of a specific product*, which takes part in a different context, the *application engineering*, this is, the team in charge of configuring a specific product is the one in charge of this product, not the whole SPL.

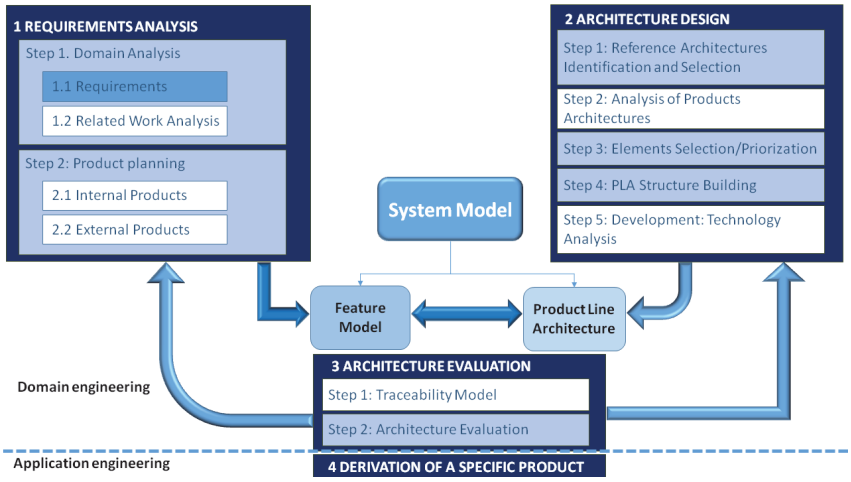


Figure 1 Methodology for constructing an SPL [5].

The first stage of this methodology, *requirements analysis*, was addressed in [16]. In this work, we continue following the next stages of the methodology, defining the complete set of features for an SPL for *digital libraries*. An architecture design for this domain is proposed and implemented which will be evaluated by generating three different products to verify the resulting product line.

3 A Software Product Line for Digital Libraries

The development of digital libraries has been focused on web-based applications. An evolving set of standards and protocols, such as the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) [8], provides repository interoperability, as defined by several authorities. Some of these standards are international standards set by bodies like the International Organization for Standardization (ISO) and the Internet Engineering Task Force (IETF), others are national standards set by bodies like the National Information Standards Organization(NISO) in the United States or the British Standards Authorities (BSA) in the United Kingdom [6], some others are industry standards set by industry bodies and many others are corporate standards produced by a single company and accepted by widespread usage. Among them Open Archival Information System (OAIS) standard becomes relevant since it is considered the optimum standard to create and maintain

a digital repository over a long period of time. ISO 14721:2012 defines the reference model for an OAIS, an archive with an organizational scheme composed of people and systems that has accepted the responsibility to preserve information and make it available for a designated community [7, 17].

In this Section, we present an SPL for web-based DLs.

3.1 Requirements Analysis

This Section illustrates the first stage for the definition of a *software product line (SPL)* in the domain of digital libraries following the methodology defined by the authors [5]. First of all, we describe the domain. Then, we detail the two steps of this stage (see Figure 1). Finally, we show and explain the feature model defined as the result of the analysis.

3.1.1 Domain analysis

Digital libraries share a common set of features that are usually present in most digital libraries. First of all, digital objects are stored to allow remote access to resources. The distinction of user roles is key, the admins take the role of librarians, in charge of updating and maintaining the *digital library*, while public users are the readers who access and search the information. Additionally, most digital libraries implement the OAI-PMH protocol, used by both libraries and museums to disseminate metadata.

We classified the domain requirements in four different groups: Library objects (R1), which includes all the requirements related to which kind of elements the library should handle (e.g., editions, collections, kinds of authorities, etc.); Library data exportation (R2), with the requirements regarding the ways the data may be exported from the library (e.g., supported formats, exportable elements, etc.); User management (R3), with the requirements regarding the user roles the application should handle, and the features associated to each of these roles (e.g., reporters and collaborators, digital file access by anonymous users, the possibility of registering in the application, etc.); Library access (R4), which involves a huge set of variable requirements that involve the kind of navigation digital libraries must support (e.g., the most important elements are editions, works or authors, the way the users can use search engines, etc.). Besides these, there is another requirement: applications should be in a particular language (R5), which usually corresponds with the language of the content available in the *digital library*.

We modeled the required features for systems of this domain with the aforementioned requirements in mind. The result is a set of 165 features (see

Table 1) representing the most relevant functionalities of these applications. Deciding the grade of importance is done in section 3.1.2.

3.1.2 Product planning

The purpose of this step is to verify the requirements extracted from the previous step with actual products in production. Besides that, analyzing these products we can establish which features are common to all the family of variant products what kind of relationship exists between a feature and its sub-features, and also determine the constraints between the features that are not related to the feature tree itself.

Besides that, another goal fulfilled within this step is the prioritization of the features, classifying the features regarding their appearance in existing products of the family line. This is important not only as a validation of the features but also to facilitate the development of the source code of the SPL: the more relevant features should be implemented first, whereas the development of features that are more specific to certain products can be delayed. This serves to reduce the initial cost of the product line, since products can be generated earlier, and to improve the quality of the rest of the components, since their implementation can take advantage of the users of these initial products, improving the quality of the final interfaces and functionalities based on their feedback.

The list of products presented for comparison includes products that our laboratory has developed in the past, which allows complete access to software assets and data models. The three products selected are:

- **Association of Writers in Galician Language (AELG)**[8]: an official entity that brings together more than 415 writers of Galician literature in all its manifestations. In this case, in addition to common features, they require that an admin validates the registration of new users and use a static page as the homepage.
- **Socio-Pedagogical Galician Association: Galician Literature History (ASPG-HLG)**[9]: focused on pedagogical objectives and linguistic standardization, they offer a historical collection of works, organized by century. Their main page is static and they store information regarding organizations that were relevant for Galician literature.

[8] AELG: https://www.aelg.gal/

[9] ASPG-HLG: http://literaturagalega.as-pg.gal/

Table 1 Features definition (Full version of the table can be seen in Appendix 4)

REQ	FEATURES	DESCRIPTION
R1	LibraryObjects	
R1.1	LO_Element	Englobes all the different types of publishments, from local journals and diaries to the most expensive and sinuous future. Can be Edition, Work, Part or Collection
R1.1.1	LO_E_SchedulingPublication	Allows scheduling a publication to publish in a given time
R1.1.2	LO_E_ElementTypes	Types of publication elements supported
R1.1.3	LO_E_LocationManagement	Management of the cardinal location in which an edition was published
R1.1.6	LO_E_DigitalResource	Digital resource types supported by the library. Can be links to external resources, EPUB, video, audio, PDF or digitized pages
R1.2	LO_Authority	Authorities related to the publication of an element
R1.2.1	LO_A_Alias	Pseudonym associated with an authority
R1.3	LO_Organization	Organization related to literature
R1.3.1	LO_O_Director	Authority that worked as director of an organization
R2	LibraryDataExportation	Allows the exportation of data from the digital library for download
R2.1	LDE_FileType	File types that can be exported. Can be PDF, TXT, JSON, Excel, EPUB, HTML or CSV
R2.2	LDE_Capabilities	Capability to export several items at the same time. Can export one single item or select several items
R2.3	LDE_ExportableElements	Types of elements that allow exportation. Can be authorities, collections, edition, works, parts, organization or the bibliographic references
R3	UserManagement	Allows the management of different users of the application
R3.1	UM_UserProfiles	Stores additional information for the profile of the user such as full name, occupation, etc.
R3.2	UM_UserRoles	Roles a user can take inside the digital library
R3.2.1	UM_UR_Admin	Admin user has full access to the library. Can modify all the data, review content uploaded by collaborators and manage static pages
R3.2.2	UM_UR_Collaborators	Collaborator user, can access admin site and update data if allowed, or modify data uploaded by oneself
R3.2.3	UM_UR_Reporters	Reporter user, has access to the public site, can manage lists of favourite authors and works
R3.2.4	UM_UR_Anonymous	Anonymous users can manage if they are allowed to access the digital resources or not
R3.3.2	UM_AR_SocialRegistration	Allows logging into the digital library using a social networks account
R4	LibraryAccess	Manage the ways to access the library and its content
R4.1	PublicAccess	Access to the public site of the digital library
R4.1.1	PA_Search	Allows searching the data available. Offers different search results, as lists or as a map. Enables choosing simple search by different fields or advanced search filtering several fields at the same time.

(*Continued*)

Table 1 Continued

REQ	FEATURES	DESCRIPTION
R4.1.2	PA_Navigation	Manage the navigation flow across the library. Choose the home page, which can be a static page or a list of elements
R4.1.3	PA_Maps	Manage the visualization of data represented in a map
R4.2	ManagementAccess	Access to the management site of the digital library
R4.2.1	MA_Search	Manage the search capabilities of the management site. Allows advanced by several fields
R4.2.2	MA_Lists	Control the filtering of the different pages, allowing simple search
R4.2.3	MA_LibraryUsageStatistics	Collect usage statistics for digital libraries like the number of elements published
R4.2.4	MA_MapSearch	Manage the map search of the different editions
R5	Language	Manage the language in which the library is developed to match the language of the content

- **Galician Virtual Library (BVG)**[10]: pioneering initiative developed by a group of interdisciplinary work of specialists in Galician-Portuguese Philology and Computer Science.

Additionally, Miguel de Cervantes Virtual Library was considered as a case of study due to its relevance in the domain of Spanish digital libraries, being one of the most complete products released in the field, and as an external product since it is important not to validate only with our products.

Apart from validating the coherence of the features identified, during this step a priority value was set for each feature, from 0 (meaning none of the studied products implements this feature) to 4 (meaning the feature is shared by all the products), giving a clear overview of which ones are common to every product, therefore becoming mandatory, and which appears only in some of them, becoming optional features. Table 5 shows the priorities of the features and whether they were implemented or not.

3.1.3 Feature model

The resulting feature model is shown in Figure 2. Each requirement group, described in detail in Section 3.1.1, derived in a high-level feature: `LibraryObjects`, `LibraryDataExportation`, `UserManagement`, `LibraryAccess` and `Language`.

`LibraryObjects` feature (contains 35 sub-features) is directly related to the kind of data the *digital library* supports. `LO_E_ElementTypes` sub-features include a feature to enable works, editions, parts of works, and

[10]BVG: https://bvg.udc.es/

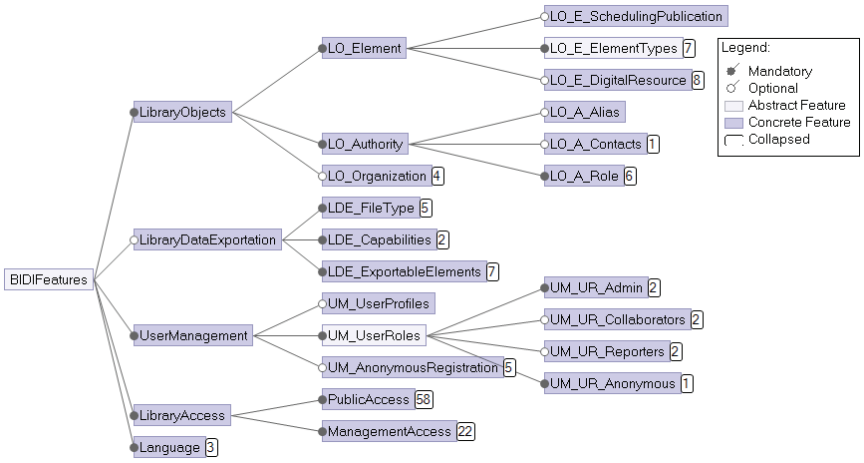


Figure 2 Simplified feature model (many features are collapsed).

collections (the first two are mandatory since we have not found a DL without this kind of elements). There is also a feature to enable the capability of managing the locations where editions are published, including setting the actual geographical coordinates in a map or managing ancient names of places for historical purposes. `LO_E_SchedulingPublication` enables scheduling elements for publication, instead of showing them the moment they are created, allowing easy management for release dates. `LO_E_DigitalResource` includes sub-features to decide which kind of digital resources are allowed. `LO_Authority` includes sub-features to define the possible roles for the authorities managed (i.e., author, editor, illustrator, etc.), the possibility for authors to have aliases or pseudonyms, and to store contact information for the authors. `LO_Organization`, which is optional, enables managing the organizations, and it has sub-features to decide if the director of an organization is stored as an authority.

`LibraryDataExportation` feature (contains 17 sub-features) groups all the features related to exporting data from the listings of the digital library. The sub-features define the possible formats of the exported information (PDF, EXCEL, TXT, etc.), how the feature is applied (to single elements or multi-element exportation), and the elements that allow exportation.

`UserManagement` feature (contains 19 sub-features) enables the user authentication module, which is mandatory since every *digital library* requires at least one user with administration permission. There are features to enable other three roles: Reporter, a standard registered user role;

Collaborator, people that collaboratively digitized content and keeps the library updated; and anonymous users. Other relevant features are related to each role, such as allowing access to digital resources for anonymous users or requiring an administrator to review the modifications done by the collaborators before publishing it. Further providing users with the option of logging into the *digital library* using their social networks account is another feature, as well as allowing anonymous users registration as reporters.

`LibraryAccess` feature (82 sub-features) is composed of every characteristic of the *digital library* related to its user interface. It has two main sub-features, `PublicAccess` and `ManagementAccess`, which encompass the features for the anonymous users and reporters, and collaborators and administrators, respectively. The two sub-trees are equivalent, describing the lists and search engines that the *digital library* provides. The public interface differentiates between two possible search engines: a simple one, which can be accessed through the menu bar across the whole *digital library*, and an advanced one that has its specific link, and which has options to enable searching by different properties. Furthermore, the listings of the *digital library* can have different types of pagination style (depending on the number of elements expected), and they can also be standard listings or grid views with thumbnails of the elements. Features related to the management interface are more simple since there is only an advanced search engine with specific options such as filtering elements not reviewed, or filtering elements that are drafts, for example. Besides that, `PublicAccess` has some sub-features that define which is the home page of the *digital library*, which can be a list of a particular element kind (editions, authorities, or works), the list of latest publications, or a static page that can be managed by the administrator (if the feature related to this capability is selected).

`Language` feature (3 sub-features) serves for selecting the language in which the labels for the user interface of the generated product should be. For the libraries analyzed, three languages need support: English, Spanish and Galician.

Besides the 162 features, there are 13 cross-tree constraints affecting 22 features. Constraints come naturally and are logical e.g., the homepage of the public access cannot be the lists of editions if listings are disabled (`PA_N_HP_Editions` implies `PA_N_L_Editions`, or "DL public homepage is the list of editions" implies "DL provides the lists of editions for public access"). Taking into account the constraints, if we compute all the possible combinations of features, the SPL would allow us to implement 7,806 completely different products.

3.2 Architecture Design

In this stage, we design the architecture of the products generated by the SPL. The decisions regarding this architecture, both from the functional and technological points of view, must be supported by existing product architectures and literature, as well as the current technological context. This stage is divided into five minor steps, which we describe in this Section.

The first step is *reference architectures identification and selection*, consisting of searching for proven or standard architectures, both in academia and industry. Some time ago, there were attempts to define a reference architecture for web-based digital libraries, but it seems the work has not been continued, and nowadays there is no real standard on how to build a *digital library*. However, back in 2003, some principles for *digital library* design were established [10] that are still relevant.

- The architecture must be service-driven and provide the tools to deliver the service.
- It must have an open architecture that supports interoperability among systems.
- The architecture must be scalable, robust, and reliable in a high transaction rate setting.
- The architecture must ensure persistent access to the collection of the *digital library* to provide digital preservation.
- The architecture must manage privacy and support both anonymous and customized access.
- The architecture should implement flexible and practical approaches to standards.
- Modularity that allows the combination of new technology with legacy pieces, all of which must interoperate.

The definition of an adequate architecture for digital libraries has been targeted by our research group for some time already. Already in 1999, we proposed an architecture for virtual libraries [1]. Later on, we designed and developed both a virtual library for emblem books [11] and the *Galician Virtual Library* [12]. Moreover, we have also worked on the functional point of view, with proposals that focus on feeding the DL, one of the most important features of a DL [13, 14].

Besides searching for reference architectures, we also analyzed the architectures of the products in Section 3.1.2 as part of the step *analysis of products architectures*. However, since there is not a standard architecture for DLs, and both the related literature and the product architectures are outdated, the

resulting product line architecture was mostly designed from scratch, focusing on the functional point of view to determine which components should be included in our product line architecture (step *element selection/priorization*). This process was guided also by the priority order resulting from the product planning step, which can be seen in Table 5.

The resulting product line architecture is shown in Figure 3. This architecture for the family of digital libraries is based on the client-server model, and therefore it is composed of two differentiated elements, the server or back-end, and the client or front-end. Communication between the server and the client-side is done using the REST paradigm, and each side has a set of components devoted to handling this communication: `Repositories` and `WebControllers`. Besides that, the server-side has a very typical architecture for web applications nowadays, with a model layer divided into services and data access objects, as well as entities to represent the objects of the database. The client-side is designed to have components for every kind of feature supported, some of them based on entities (`EntityComponents`), and some which are smaller components used in different parts of the application (`CommonComponents`). There are also transversal components both on the server-side and on the client-side, which are related to authentication and localization.

This architecture favors the separation and independence between both elements, facilitating exchange or modification among them without affecting other parts of the system. Besides, it allows the existence of different clients for the same server and the replication of components to cope with high demand.

The last step is *development: technology analysis*, which focuses on exploring and choosing the right set of technologies for the development of the products taking into account all the requirements and context. For the defined architecture, three different types of technologies and tools need to be used depending on the part of the architecture where they are applied: relational database, back-end, and front-end. For the first of them, we have only considered open source technologies, and, among them, we chose PostgreSQL[11] due to its usability and compatibility. Regarding the back-end or server-side, we decided to take advantage of our expertise in Java technologies, which are also used in most of the applications developed by companies. Therefore, we chose to use Spring[12] as the most known alternative for Java

[11] PostgreSQL: https://www.postgresql.org/

[12] Spring: https://spring.io/

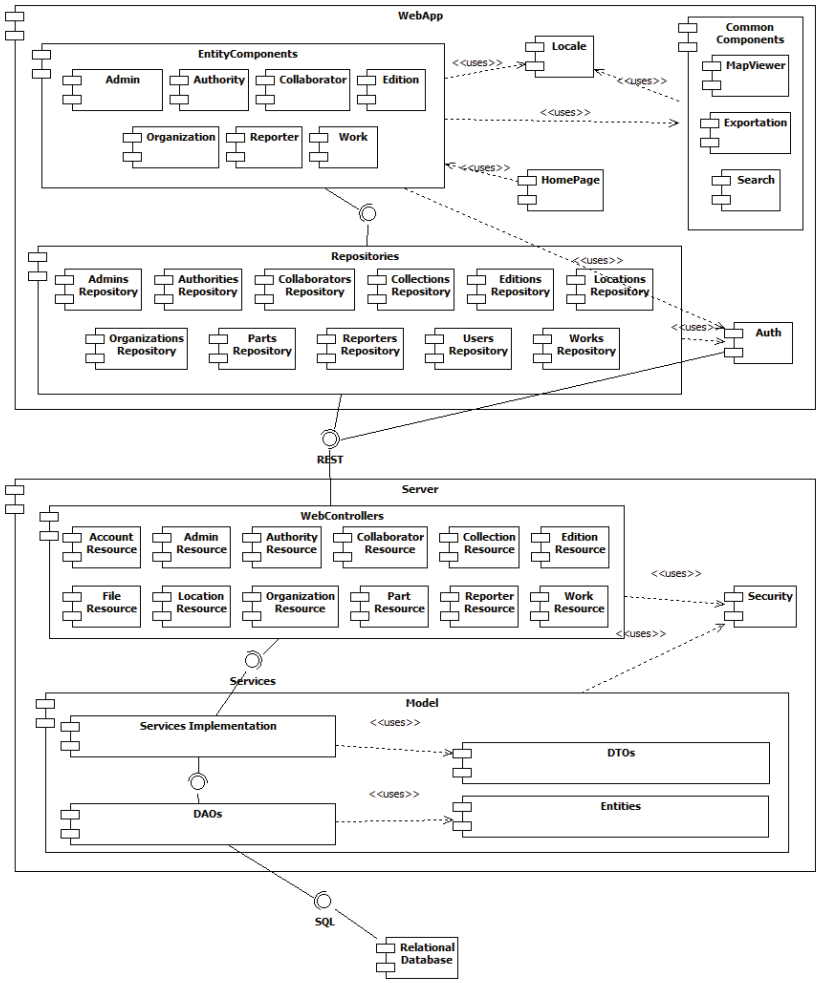


Figure 3 Product line architecture.

data server and its set of libraries (such as Spring MVC, Spring Security, etc.). For the front-end or client-side, current web applications use the *single page application* pattern to simulate the feeling of a desktop application. Among the set of well-known frameworks supporting this pattern, we consider Vue.js[13] as an adequate choice for its learning curve, since the main alternatives are harder to learn for new developers, and its popularity.

[13] Vue.js: https://vuejs.org/

3.3 Architecture Evaluation

Once both the feature model and the Product Line Architecture (PLA) were defined, the next stage in the methodology requires matching or creating a traceability model among features and functional components of the platform. In case there are inconsistencies, the two first stages should be done again in an iterative way to refine their output and solve said inconsistencies.

The traceability among features and the components for the product line can be seen in Table 2. Since the architecture was built mostly from scratch, as we mention in the previous section, there are no discrepancies between the PLA and the feature list and we did not need to adjust the architecture.

4 Implementation of the SPL

The derivation engine employed to build the SPL was spl-js-engine[14], created by the authors, and successfully applied to previous projects such as a SPL for web-based geographic information systems [4, 5]. The derivation engine was designed following the principles of negative variability, that is, the base code of the components includes all the code that may be needed but has annotations in the form of comments that enables associating parts of the code with the product's features. This way, when the user selects which features to include in a specific product, the derivation engine takes those components and tweaks or removes unnecessary code for that specific product.

The advantages of this implementation are that it is less invasive regarding the code because the annotations are written as comments. Besides, it receives the feature specification in a standard format, JSON[15], so that any specification interface could generate it very easily.

The architecture, as seen in Figure 4 is the usual one of a *Software Product Line*. The specification interface is a custom-made web interface, seen in Figure 5. It is an application that has contemplated the feature tree that allows selecting them and the dependencies between the characteristics. Depending on what the user selects, it generates the Product specification in JSON format that will be the input for the derivation engine.

The components were implemented in Java and Vue.JS, with comment annotations so that it was not intrusive. The specification interface generates the JSON and invokes the generation engine, whose output is the generated result, and returns the full product compressed in a ZIP file ready to deploy.

[14] spl-js-engine: https://github.com/AlexCortinas/spl-js-engine
[15] JSON: https://www.json.org/json-en.html

Table 2 Traceability features – components

FEATURES	SERVICES
LibraryObjects	Relational Database
LO_Element	Editions, Works, Collections, Parts
LO_E_SchedulingPublication	
LO_E_ElementTypes	
LO_E_LocationManagement	Locations Repository
LO_E_DigitalResource	File Resource
LO_Authority	Authority Resource
LO_Organization	Organization Resource
LO_O_Director	
LibraryDataExportation	Exportation
LDE_FileType	
LDE_Capabilities	
LDE_ExportableElements	
UserManagement	Users Repository, Auth
UM_UserProfiles	
UM_UserRoles	
UM_UR_Admin	Admins Repository, Auth
UM_UR_Collaborators	Collaborators Repository, Auth
UM_UR_Reporters	Reporters Repository, Auth
UM_UR_Anonymous	Auth
UM_AnonymousRegistration	
UM_AR_SocialRegistration	
LibraryAccess	HomePage
PublicAccess	
PA_Search	Search
PA_S_Advanced	
PA_S_Simple	
PA_Navigation	Editions, Works, Authorities
PA_N_Lists	
PA_N_HomePage	HomePage
PA_Maps	Map Viewer
ManagementAccess	Auth
MA_Search	Search
MA_Lists	Editions, Works, Parts, Collections, Collaborators, Reporters
MA_L_Filter	
MA_L_AlphabeticRangePagination	
MA_LibraryUsageStatistics	
MA_MapSearch	Map Viewer
Language	Locale

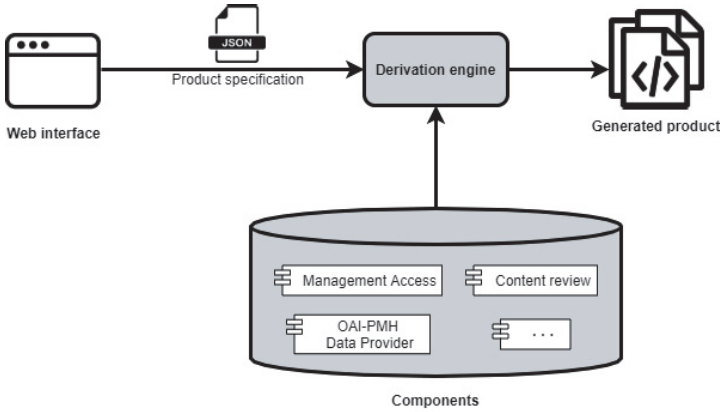


Figure 4 Software product line architecture.

4.1 Data Model Variability

One of the complexities of implementing this SPL was that the variability does not only affect the components and which lines of code are included or not. The variability can even modify the basic structure of the data model and change the whole implementation of the database, the entities, and the classes. Given the feature model that was presented beforehand (see Section 3.1.3), some features determine the presence of an entity e.g. whether we need to store location information or not.

To visualize the variability a small piece of the data model is presented in Figure 6, although the complete model can be seen as complementary resources in Appendix 11.

A new notation had to be created to define the variability of the SPL due to the lack of a standard pre-defined that allowed the representation of all the possibilities. The presence or absence of a class or attribute is denoted with `[if` *`FeatureName`*`]` right before the name of the element. This means that if the feature associated with the said element is not selected, it won't be created, to begin with. In the shown excerpt, `Location` entity is included in the product only if the features *`LocationManagement`* is selected.

The relations among the different entities can also vary depending on the specification. The variability of a relation is denoted with a dotted line labeled with the name of the feature determining its presence as `[if` *`FeatureName`*`]`. This variability can go to the point of changing the whole structure of the database to make two entities have a direct connection or, conversely, create a new entity that is related to both. In the shown

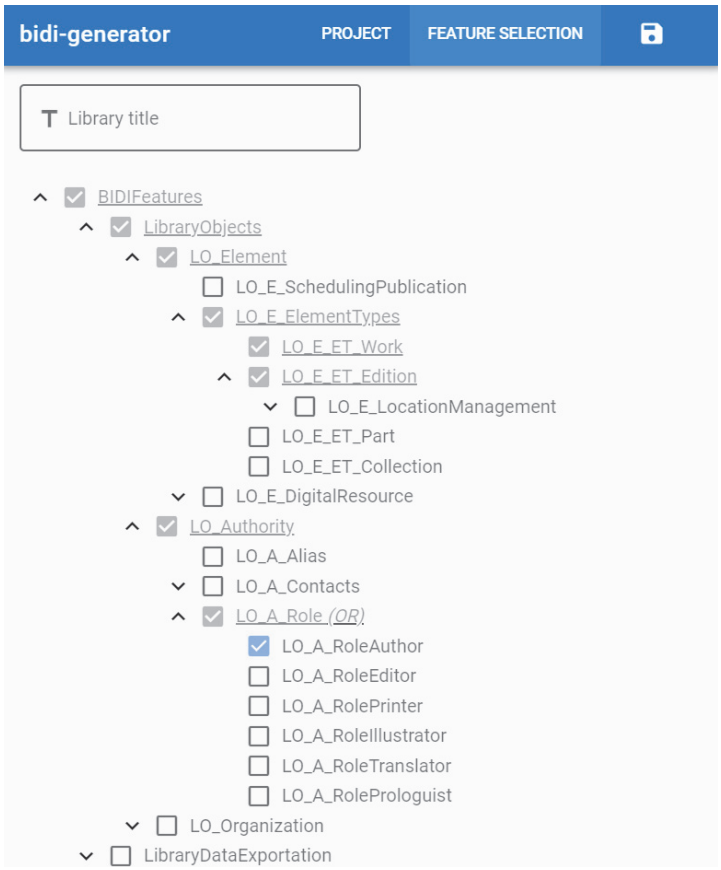


Figure 5 Generation interface.

excerpt, depending on whether the feature *`Alias`* is selected or not, there is an entity `Alias` with two relationships (`Alias-Authority` and `Alias-AuthorityElement`), or an alternative relationship between `Authority` and `AuthorityElement`.

4.2 Annotations

Annotations are the key point of the derivation engine used, which offers the possibility to establish different delimiters for each file type, therefore enabling the integration of the annotation as code comments so that they do not impact code compilation, neither on the IDEs used to develop this

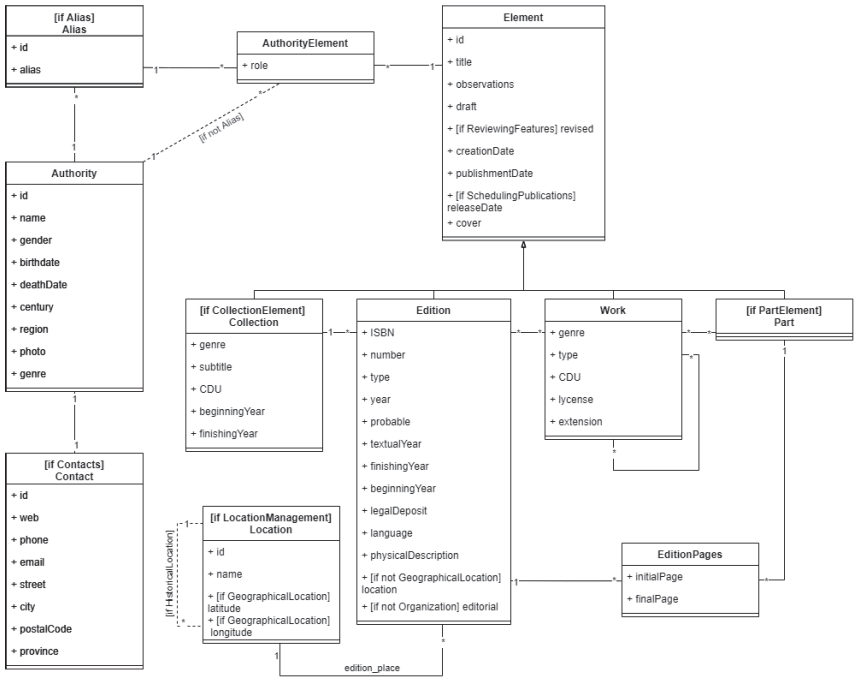


Figure 6 Piece of the data model.

code. Nevertheless, there are very specific exceptions to this. For example, the framework Vue.js supports mixing HTML, JS, and CSS languages in one file type, keeping the comments for each side with the original delimiters, instead of unifying them as other frameworks do in the same situation, such as AngularJS[16]. Therefore, annotations in files of this kind do break the code in IDEs, and they should not be used in case the developers prefer to keep the annotated code working (in the specific case of Vue.js, there is an alternative way of using the framework without mixing languages). Another exception is for example JSON files, which do not support comments and therefore compilation errors may arise when compiling the code.

A small code snippet can be seen in Listing 1, only to enable the visualization of how these annotations work, inserted in the main code and following the delimiter configuration stated above. The annotation process entails wrapping up a slice of code that must be added just in case the feature associated with that piece was selected in the *FeatureTree*.

[16] AngularJS: https://angularjs.org/

```
public void delete(Long id) throws NotFoundException{
    Authority bdAuthority = authorityDAO.findById(id);
    if (bdAuthority == null)
        throw new NotFoundException("authority with id " + id + " not found!");
    /*% if (feature.LO_A_Contacts) { %*/
    if (bdAuthority.getContact()!=null) {
        /*% if (feature.LO_A_SocialNetwork) { %*/
        List<SocialNetwork> sns =
        socialNetworkDAO.findByContact(bdAuthority.getContact().getId());
        if (sns!=null)
            sns.forEach(social -> socialNetworkDAO.delete(social.getId()));
        /*% } %*/
        contactDAO.delete(bdAuthority.getContact());
    }
    /*% } if (feature.LO_A_Alias) { %*/
    List<Alias> aliases = aliasDAO.findByAuthority(bdAuthority.getId());
    if (aliases!=null)
        aliases.forEach(alias -> aliasDAO.delete(alias));
    /*% } %*/
    authorityDAO.delete(id);
}
```

Listing 1 Code snippet from the REST service with annotations

5 Evaluation of the SPL

The main benefits provided by SPLE are the systematic reuse of software artifacts and a reduction on the development effort compared with an ad-hoc implementation of each product from scratch. In this section, we present a study on the use of the SPL that allowed the evaluation of the tool and its implementation, focusing on the size and characteristics of the generated products, which have a direct impact on the development effort. Three sample projects with different specifications were generated to perform the evaluation. Each sample is characterized by its number of entities and lines of code since these elements define the size of the system and are subjects of variation with the SPL. For the three of them, the resulting *digital libraries* were analyzed in terms of the total number of source code files, and the number of generated lines of code.

In the rest of this section, we describe the sample projects and their models. Later on, the results obtained for the generated products are compared and discussed the finding and limitations of the case study.

5.1 Sample Project 1: AELG

The first application covers the features needed to recreate the services offered by the current AELG website. Said features can be seen in Table 5.

The specification defines an application that will allow storing editions, literary works, and parts of works or editions, although no collections nor locations are required. Since this model does not include a separate entity for both Location and Organization, they are stored as attributed inside the Edition object. This model dispenses with the Alias class too since the literary works are always signed with the same name when we talk about one author, so the relation between Authority and Element is direct. However, since it is a contemporary association where authors can provide their content to update their profile, they store contact information and social networks.

On the user management side, anonymous users can register into the site but they need to be validated by an admin before being granted full access. Figure 7 shows the resulting class diagram for the application generated with this specification, note that the significant changes are highlighted in red.

5.2 Sample Project 2: BVG

The second application covers the features needed to recreate the Galician Virtual Library (BVG) and includes the features marked in Table 5. This specification defines a platform that will allow storing editions, literary works, and parts of works or editions, as well as the previous sample. However, this time the author of each element is stored by its alias, allowing the storage of several aliases for each authority, so a new entity is created at generation time. This model only contemplates admin users with basic profiles, discarding both reporters and collaborators. For the digital resources, BVG offers external resources in addition to videos, audios, PDFs, and images.

Due to the lack of information on the management side of these platforms caused by the absence of more detailed documentation, several features were never selected. To reproduce a real specification, some features were selected by the authors even if they were not marked in Table 5, taking into account the structure of the application to include those features that fitted best. That is the case of the scheduling publication features, which causes the addition of the "releaseDate" attribute in class Element. The complete model for this application can be seen in Figure 8, with the relevant variations highlighted in red.

5.3 Sample Project 3: ASPG-HLG

The last sample project was created to replicate the services offered by the Galician Socio-Pedagogical Association, whose complete list of features can be seen in Table 5. For the last case, the type of elements managed is the same

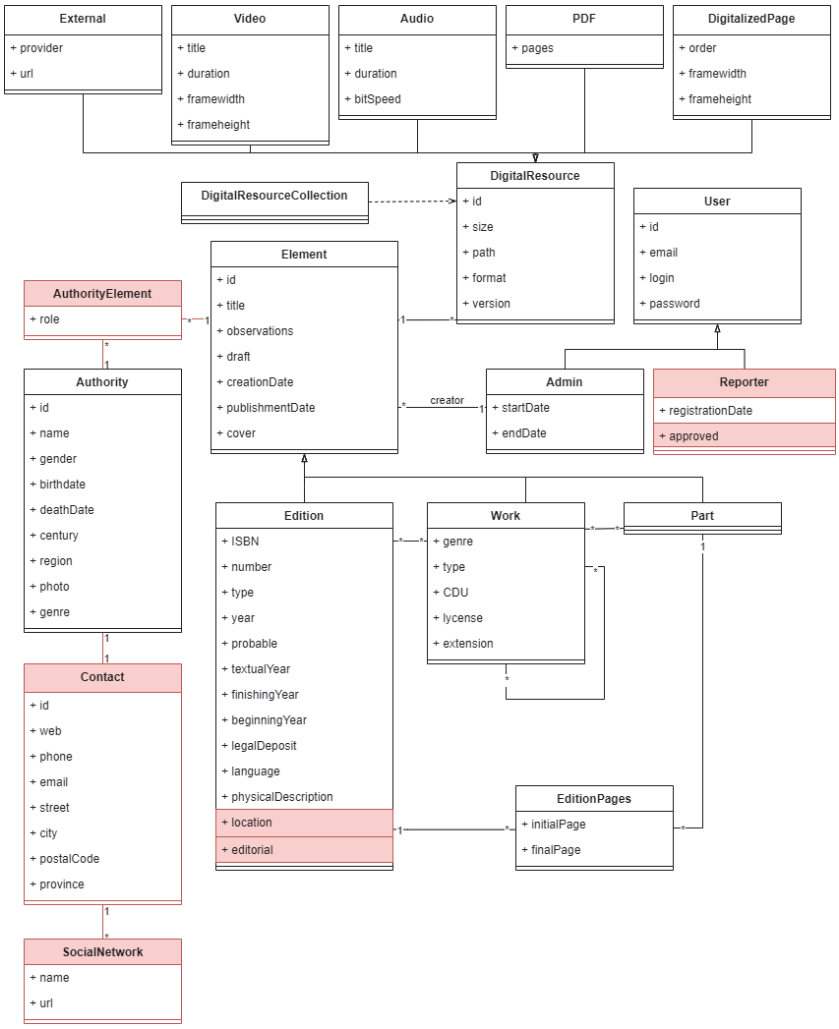


Figure 7 Class diagram of the AELG example project.

as the two previous cases: edition, literary work, and part of work or edition. This set seems to be consistent. The digital resources stored related to said elements, in this case, are external resources, videos, audios, and images. As in the first sample, this case does not store aliases for each author, so the relation between authority and element is direct as well. One new variation is that AS-PG stores information regarding organizations, in particular stores data about institutions and associations, so an additional entity is modeled.

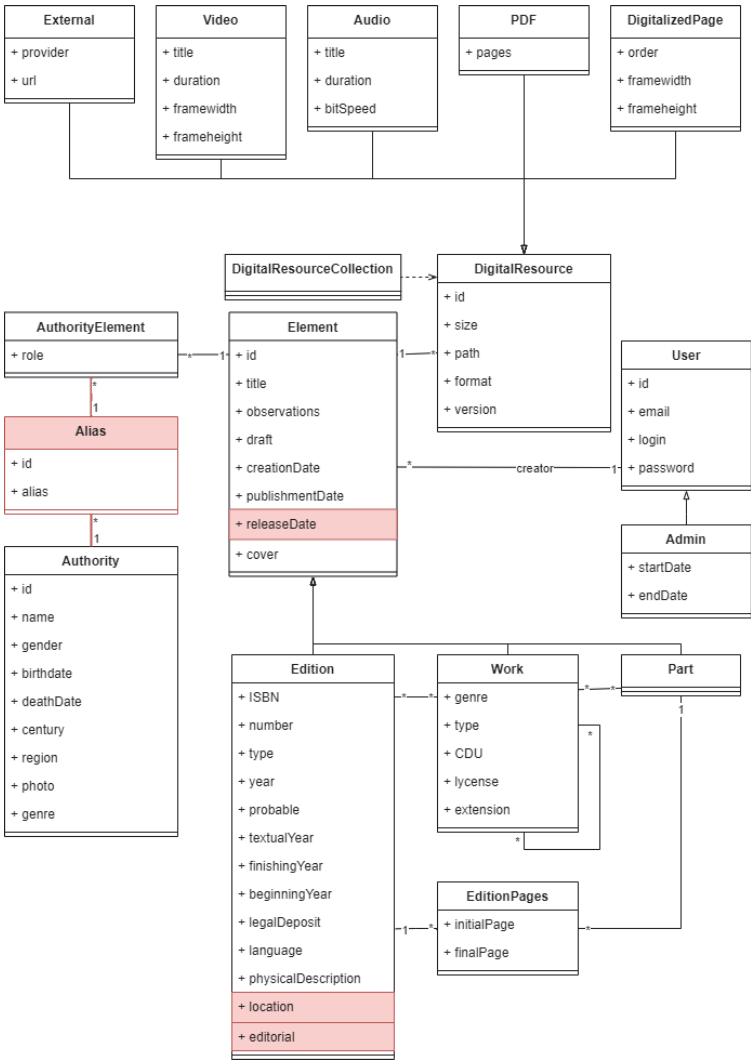


Figure 8 Class diagram of the BVG example project.

Regarding user management, this model differentiates between collaborators and admins, although it does not allow for anonymous users to register on the site as reporters. Even if it does not allow reporters, the rest of the users must provide personal information to access the platform, so the User and Collaborator entities include several related attributes. To fit the presence of collaborators, the feature of reviewing elements was modeled

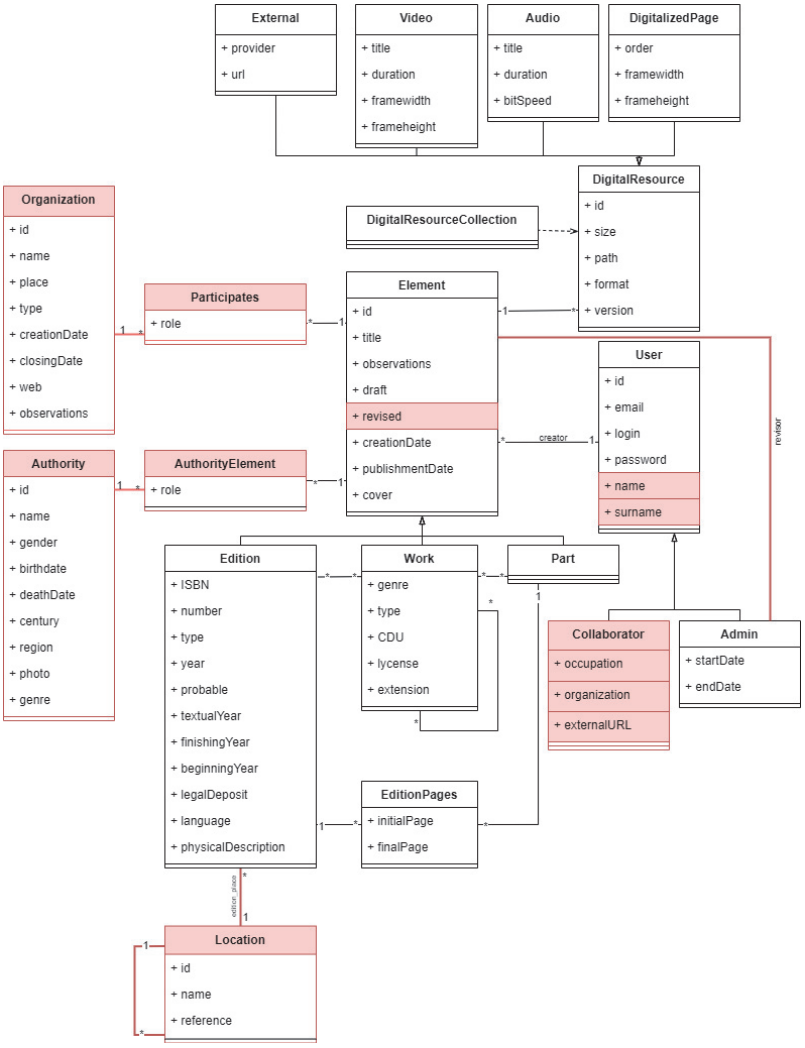


Figure 9 Class diagram of the ASPG example project.

as well in this sample, which adds the attributes “revised” and “revisor” to the Element class. Lastly, an entity to store locations was modeled to store historical locations, since the Galician History Literature library manages data about past literary eras and places. The complete class diagram depicting the model for this project can be seen in Figure 9, with the relevant variations highlighted in red.

5.4 Results

In this section, the results of the analysis obtained from the three sample projects are presented. The specification of each project was modeled using the generation tool developed for this SPL, which depicts the full feature tree and provides a user-friendly interface to easily generate the products. Three applications were generated later, with the specifications detailed beforehand. As explained in the previous section, the software is generated from a set of code templates and a base application, that comprises 234 files (Java, XML, JSON, Vue.JS, JavaScript, HTML, and YAML) and 13,583 lines of code. The number of source code files and the number of lines of code (LOC) were measured for each of the three projects. In the case of the lines of code, there is a distinction between the number of LOC in the back-end and the front-end.

Table 3 shows the characteristics of the three sample projects regarding the size of the resulting generated code, in terms of number of files generated, number of classes generated, number of entities generated, and number of lines of code (LOC).

As can be seen in the table, the difference between the three sample projects allows us to analyze the results of the generation process with different specifications. In the first and third cases, the number of LOC is similar in the back-end, since they both implement a total of 19 entities. However, the number of LOC in the front-end is higher in the third case because some of the entities implemented, such as Organization and Location, require their own thesauri pages to allow information management by both collaborators and admins. Analyzing project 2, the number of generated LOC is small compared to the number of lines of the base application. Additionally, it can be noted that the number of generated LOC per entity is almost the same in the first two projects, 484 in project 1 and 494 in project 2, while it increases to 530 in project 3.

The advantages of using the SPL approach over single system development become perceptive from the third software system development [15]. The effort or cost of producing several products stabilizes after the third product, while the traditional implementation keeps a linear increasing trend, each

Table 3 Characteristics of the generated sample projects

				Lines of code (LOC)		
Project	Files	Classes	Entities	Total	Back-end	Front-end
Project 1	186	130	19	9,190	6,436	2,754
Project 2	161	110	17	8,396	5,641	2,755
Project 3	196	127	19	10,080	6,608	3,472

new product supposes starting from scratch with the same effort. However, it must be noted that the initial effort of developing an SPL is always bigger. This experiment has proved that after producing three projects, we generated 27,666 LOC, while we only had to manually implement the 13,583 LOC that compose the initial application.

Quality and maintainability are two other aspects of the created code that can be investigated. Since it is created automatically, the code generated by our implementation follows a simple architecture and a set of design patterns, with no possible deviation from those prescriptions.

The evaluation described in this section presents some validation threats. First, we chose three test projects that show the SPL approach is feasible for the development of DLs, both in terms of development efficiency and on supporting a wide variability. However, the results obtained could be different when generating other DLs. Second, to properly evaluate the performance of the SPL against a custom development, it would be necessary to give several teams with a similar level of experience the specifications of the three products, making sure that the generated ones are treated thoroughly to achieve a refinement level similar to that of those custom made. This would be the last step of the methodology presented before in Figure 1. Anyhow, this experiment does not cover said step because of the cost it has, but we do try to generate three products and compare the size that each one of them would have. Even if it is not as precise as doing the real experiment, it provides useful information if we assess that the effort for each KLOC is more or less uniform in the SPL and custom-made developments.

6 Conclusions

In this paper, we proposed an SPL for *Digital Libraries*, which is a collection of software-intensive systems that share a common, managed set of features that meet the specific needs of the *Digital Libraries* domain and are built from a predefined set of core assets. The SPL was implemented in a tool that generates the system's source code from a user-supplied specification and returns the product ready for deployment.

The study presented uses three sample projects to evaluate software products generated using specifications from three previously analyzed pre-existing products. Because these sample projects are of varying sizes, we were able to analyze the lines of code generated in various scenarios while also taking into account the number of lines of code forming the base application. One limitation of the case study is the standardization of the products

generated, with all three of them following the same design and presenting the same appearance as a result of the automatic generation methodology as if they were manufactured products. However, when compared to the size of the system requirements, it allowed us to infer that the number of lines of code we can produce is significant. Even taking into account the limitations of the experiment, the current implementation allows us to generate 7,806 unique products without having to modify the base. As we stated in the previous section, a more thorough evaluation that includes the development of real-world applications is still in the works.

Next steps include updating and publishing a set of digital libraries using the software product line, starting with BVG (Section 5.2). During this process we will refine the SPL, probably implementing some extra feature, and refining the user interfaces to make them even more suited to the actual needs of the final users.

Acknowledgements

Partially funded by:
Xunta de Galicia GRC: ED431C 2021/53;
MICIU/FEDER-UE, MAGIST: PID2019-105221RBC41;
MICIU/FEDER-UE, EXTRACompact: PID2020-114635RB-I00;
MIC IU/FEDER-UE BIZDEVOPSGLOBAL: RTI-2018-098309-B-C32;
Xunta de Galicia/FEDER-UE, ConectaPeme, GEMA: IN852A 2018/14;
MICIU/FEDER-UE, Red Aracne-Nodus, RED2018-102755-T;
CITIC research centre funded by XUNTA and EU through the European Regional Development Fund- Galicia 2014–2020 Program, grant ED431G 2019/01.

Appendix

Table 4 Complete list of features definition

REQ	FEATURES	DESCRIPTION
R1	LibraryObjects	
R1.1	LO_Element	Englobes all the different types of publishments, from local journals and diaries to the most expensive and sinuous future. Can be Edition, Work, Part or Collection

(*Continued*)

Table 4 Continued

REQ	FEATURES	DESCRIPTION
R1.1.1	LO_E_SchedulingPublication	Allows scheduling a publication to publish in a given time
R1.1.2	LO_E_ElementTypes	Types of publication elements supported
R1.1.3	LO_E_LocationManagement	Management of the cardinal location in which an edition was published
R1.1.4	LO_E_HistoricalLocation	Reference to historical names of places with a different name in present times
R1.1.5	LO_E_GeographicalLocation	Coordinates for a geographical location
R1.1.6	LO_E_DigitalResource	Digital resource types supported by the library. Can be links to external resources, EPUB, video, audio, PDF or digitized pages
R1.2	LO_Authority	Authorities related to the publication of an element
R1.2.1	LO_A_Alias	Pseudonym associated with an authority
R1.2.2	LO_A_Contacts	Contact information to reach contemporary authors
R1.2.3	LO_A_SocialNetwork	Social network information for contemporary authors
R1.2.4	LO_A_Role	Role an authority took in the creation process, can be author, editor, printer, illustrator, translator or prologue writer.
R1.3	LO_Organization	Organization related to literature
R1.3.1	LO_O_Director	Authority that worked as director of an organization
R1.3.2	LO_O_Role	Role of organization in the life of a literary work. Can be library, institution, association or editorial
R2	LibraryDataExportation	Allows the exportation of data from the digital library for download
R2.1	LDE_FileType	File types that can be exported. Can be PDF, TXT, JSON, Excel, EPUB, HTML or CSV
R2.2	LDE_Capabilities	Capability to export several items at the same time. Can export one single item or select several items
R2.3	LDE_ExportableElements	Types of elements that allow exportation. Can be authorities, collections, edition, works, parts, organization or the bibliographic references
R3	UserManagement	Allows the management of different users of the application
R3.1	UM_UserProfiles	Stores additional information for the profile of the user such as full name, occupation, etc.
R3.2	UM_UserRoles	Roles a user can take inside the digital library
R3.2.1	UM_UR_Admin	Admin user has full access to the library. Can modify all the data, review content uploaded by collaborators and manage static pages
R3.2.2	UM_UR_Collaborators	Collaborator user, can access admin site and update data if allowed, or modify data uploaded by oneself
R3.2.3	UM_UR_Reporters	Reporter user, has access to the public site, can manage lists of favourite authors and works
R3.2.4	UM_UR_Anonymous	Anonymous users can manage if they are allowed to access the digital resources or not
R3.3	UM_AnonymousRegistration	Allow registration for anonymous users without the intervention of an admin

(*Continued*)

Table 4 Continued

REQ	FEATURES	DESCRIPTION
R3.3.1	UM_AR_AdminUserValidation	Demands that an admin validates the registration before giving full access
R3.3.2	UM_AR_SocialRegistration	Allows logging into the digital library using a social networks account
R4	LibraryAccess	Manage the ways to access the library and its content
R4.1	PublicAccess	Access to the public site of the digital library
R4.1.1	PA_Search	Allows searching the data available. Offers different search results, as lists or as a map. Enables choosing simple search by different fields or advanced search filtering several fields at the same time.
R4.1.2	PA_Navigation	Manage the navigation flow across the library. Choose the home page, which can be a static page or a list of elements
R4.1.3	PA_N_Lists	Allows selecting infinite scroll or pagination and, in said case, the type of pagination desired
R4.1.3	PA_Maps	Manage the visualization of data represented in a map
R4.2	ManagementAccess	Access to the management site of the digital library
R4.2.1	MA_Search	Manage the search capabilities of the management site. Allows advanced by several fields
R4.2.2	MA_Lists	Control the filtering of the different pages, allowing simple search
R4.2.3	MA_LibraryUsageStatistics	Collect usage statistics for digital libraries like the number of elements published
R4.2.4	MA_MapSearch	Manage the map search of the different editions
R5	Language	Manage the language in which the library is developed to match the language of the content
R5.1	L_Spanish	Implement the library in Spanish
R5.2	L_English	Implement the library in English
R5.3	L_Galician	Implement the library in Galician

Table 5 Features with priority order

REQ	FEATURES	AELG	ASPG	BVG	Cervantes	PRIORITY
R1	LibraryObjects	x	x	x	x	4
R1.1	LO_Element	x	x	x	x	4
R1.1.1	LO_E_SchedulingPublication					0
R1.1.2	LO_E_ElementTypes	x	x	x	x	4
R1.1.2.1	LO_E_ET_Work	x	x	x	x	4
R1.1.2.2	LO_E_ET_Edition	x	x	x	x	3
R1.1.2.3	LO_E_ET_Part	x	x	x		3
R1.1.2.4	LO_E_ET_Collection				x	1
R1.1.3	LO_E_LocationManagement					0
R1.1.4	LO_E_HistoricalLocation					0
R1.1.5	LO_E_GeographicalLocation					0
R1.1.6	LO_E_DigitalResource	x	x	x	x	4
R1.1.6.1	LO_E_DR_FileTypesSupported	x	x	x	x	4
R1.1.6.2	LO_E_DR_ExternalLinks	x	x	x		3
R1.1.6.3	LO_E_DR_EPUB					0
R1.1.6.4	LO_E_DR_Video	x	x	x	x	4

(*Continued*)

Table 5 Continued

REQ	FEATURES	AELG	ASPG	BVG	Cervantes	PRIORITY
R1.1.6.5	LO_E_DR_Audio	x	x	x	x	4
R1.1.6.6	LO_E_DR_PDF	x		x	x	3
R1.1.6.7	LO_E_DR_DigitalizedPages	x	x	x	x	4
R1.1.6.8	LO_E_DR_Collections	x	x	x	x	4
R1.2	LO_Authority	x	x	x	x	4
R1.2.1	LO_A_Alias			x	x	2
R1.2.2	LO_A_Contacts	x				1
R1.2.3	LO_A_SocialNetwork	x				1
R1.2.4	LO_A_Role	x	x		x	3
R1.2.5	LO_A_RoleAuthor	x	x		x	3
R1.2.6	LO_A_RoleEditor					0
R1.2.7	LO_A_RolePrinter					0
R1.2.8	LO_A_RoleIllustrator					0
R1.2.9	LO_A_RoleTranslator					0
R1.2.10	LO_A_RoleProloguist					0
R1.3	LO_Organization		x		x	2
R1.3.1	LO_O_Director					0
R1.3.2	LO_O_Role		x		x	2
R1.3.3	LO_O_RoleLibrary					0
R1.3.4	LO_O_RoleInstitution		x		x	2
R1.3.5	LO_O_RoleAsociation		x			1
R1.3.6	LO_O_RoleEditorial					0
R2	LibraryDataExportation					0
R2.1	LDE_FileType					0
R2.1.1	LDE_FT_PDF					0
R2.1.2	LDE_FT_TXT					0
R2.1.3	LDE_FT_JSON					0
R2.1.4	LDE_FT_EXCEL					0
R2.1.5	LDE_FT_EPUB					0
R2.1.6	LDE_FT_HTML					0
R2.1.7	LDE_FT_CSV					0
R2.2	LDE_Capabilities					0
R2.2.1	LDE_C_SingleItem					0
R2.2.2	LDE_C_SelectionOfItems					0
R2.3	LDE_ExportableElements					0
R2.3.1	LDE_EE_Authority					0
R2.3.2	LDE_EE_Collection					0
R2.3.3	LDE_EE_Edition					0
R2.3.4	LDE_EE_Work					0
R2.3.5	LDE_EE_Part					0
R2.3.6	LDE_EE_Organization					0
R2.3.7	LDE_EE_BibliographicReference					0
R3	UserManagement	x	x			2
R3.1	UM_UserProfiles		x			1
R3.2	UM_UserRoles	x	x			2
R3.2.1	UM_UR_Admin	x	x			1
R3.2.1.1	UM_UR_AdminReviewsElements					0
R3.2.1.2	UM_UR_AdminManagesStaticPages	x				0
R3.2.2	UM_UR_Collaborators		x			1
R3.2.2.1	UM_UR_CollaboratorsOnly ManageOwnContent					0

(*Continued*)

Table 5 Continued

REQ	FEATURES	AELG	ASPG	BVG	Cervantes	PRIORITY
R3.2.2.2	UM_UR_CollaboratorsCanManageThesauri		x			1
R3.2.3	UM_UR_Reporters	x				0
R3.2.3.1	UM_UR_ReportersCanFavoriteElements					0
R3.2.3.2	UM_UR_ReportersCanFavoriteAuthors					0
R3.2.4	UM_UR_Anonymous	x	x	x	x	4
R3.2.4.1	UM_UR_AnonymousCanAccessElementFiles	x	x	x	x	4
R3.3	UM_AnonymousRegistration	x				0
R3.3.1	UM_AR_AdminUserNeedsToValidate	x				1
R3.3.2	UM_AR_SocialRegistration					0
R3.3.2.1	UM_AR_SR_Facebook					0
R3.3.2.2	UM_AR_SR_Twitter					0
R3.3.2.3	UM_AR_SR_Google					0
R4	LibraryAccess	x	x		x	3
R4.1	PublicAccess	x	x		x	3
R4.1.1	PA_Search	x			x	2
R4.1.1.1	PA_SearchResult	x				1
R4.1.1.1.1	PA_SR_Edition	x				1
R4.1.1.1.1.1	PA_SR_E_Format				x	1
R4.1.1.1.1.2	PA_SR_E_F_AsList	x			x	2
R4.1.1.1.1.3	PA_SR_E_F_AsMap					0
R4.1.1.1.2	PA_SR_Works	x				1
R4.1.1.1.3	PA_SR_Authority					0
R4.1.1.2	PA_S_Advanced	x			x	2
R4.1.1.2.1	PA_S_A_ByTitle				x	1
R4.1.1.2.2	PA_S_A_ByAuthority				x	1
R4.1.1.2.2.1	PA_S_A_BA_Role				x	1
R4.1.1.2.2.1.1	PA_S_A_BA_R_ByAuthor				x	1
R4.1.1.2.2.1.2	PA_S_A_BA_R_ByEditor					0
R4.1.1.2.2.1.3	PA_S_A_BA_R_ByCustom					0
R4.1.1.2.3	PA_S_A_ByLocation					0
R4.1.1.2.4	PA_S_A_ByLanguage					0
R4.1.1.2.5	PA_S_A_ByDate					0
R4.1.1.2.6	PA_S_A_ByDateRange					0
R4.1.1.2.7	PA_S_A_ByIdentifier					0
R4.1.1.2.8	PA_S_A_ByISBN					0
R4.1.1.2.9	PA_S_A_ByContent	x			x	2
R4.1.1.3	PA_S_Simple	x			x	2
R4.1.1.3.1	PA_S_S_ByTitle	x			x	2
R4.1.1.3.2	PA_S_S_ByAuthor				x	1
R4.1.1.3.3	PA_S_S_ByYear					0
R4.1.1.3.4	PA_S_S_ByLocation					0
R4.1.2	PA_Navigation	x	x		x	3
R4.1.2.1	PA_N_HomePage	x	x			2
R4.1.2.2	PA_N_HP_Editions					0
R4.1.2.3	PA_N_HP_Works					0
R4.1.2.4	PA_N_HP_Authorities					0
R4.1.2.5	PA_N_HP_PresentationPage	x	x			2
R4.1.2.6	PA_N_HP_LatestPublicationsPage				x	1
R4.1.3	PA_N_Lists	x	x			2
R4.1.3.1	PA_N_L_Editions	x				1

(*Continued*)

Table 5 Continued

REQ	FEATURES	AELG	ASPG	BVG	Cervantes	PRIORITY
R4.1.3.1.1	PA_N_L_E_AsList	x				1
R4.1.3.1.2	PA_N_L_E_AsImages					0
R4.1.3.2	PA_N_L_Works		x			1
R4.1.3.2.1	PA_N_L_W_AsList					0
R4.1.3.2.2	PA_N_L_W_AsImages		x			1
R4.1.3.3	PA_N_L_Authorities	x	x			2
R4.1.3.3.1	PA_N_L_A_AsList					0
R4.1.3.3.2	PA_N_L_A_AsImages	x	x			2
R4.1.3.4	PA_N_L_Pagination	x	x			2
R4.1.3.4.1	PA_N_L_P_Editions	x				1
R4.1.3.4.1.1	PA_N_L_P_E_InfiniteScroll	x				1
R4.1.3.4.1.2	PA_N_L_P_E_StandardPagination					0
R4.1.3.4.1.3	PA_N_L_P_E_AlphabeticRangePagination					0
R4.1.3.4.2	PA_N_L_P_Works		x			1
R4.1.3.4.2.1	PA_N_L_P_W_InfiniteScroll					0
R4.1.3.4.2.2	PA_N_L_P_W_StandardPagination					0
R4.1.3.4.2.3	PA_N_L_P_W_AlphabeticRangePagination		x			1
R4.1.3.4.3	PA_N_L_P_Authorities	x	x			2
R4.1.3.4.3.1	PA_N_L_P_A_InfiniteScroll	x				1
R4.1.3.4.3.2	PA_N_L_P_A_StandardPagination					0
R4.1.3.4.3.3	PA_N_L_P_A_AlphabeticRangePagination		x			1
R4.1.3	PA_Maps					0
R4.2	ManagementAccess	x				1
R4.2.1	MA_Search					0
R4.2.1.1	MA_S_Advanced					0
R4.2.1.1.1	MA_S_A_ByElementTitle					0
R4.2.1.1.2	MA_S_A_ByElementAuthority					0
R4.2.1.1.2.1	MA_S_A_NoAuthority					0
R4.2.1.1.3	MA_S_A_ByDraft					0
R4.2.1.1.4	MA_S_A_ByRevised					0
R4.2.1.1.5	MA_S_A_ByElementId					0
R4.2.1.1.6	MA_S_A_ByElementISBN					0
R4.2.1.1.7	MA_S_A_ByElementYear					0
R4.2.1.1.7.1	MA_S_A_NoYear					0
R4.2.1.1.8	MA_S_A_ByGenre					0
R4.2.1.1.9	MA_S_A_ByEditionLocation					0
R4.2.1.1.9.1	MA_S_A_NoEditionLocation					0
R4.2.2	MA_Lists	x	x	x		3
R4.2.2.1	MA_L_Filter	x	x	x		3
R4.2.2.1.1	MA_L_F_Collaborators					0
R4.2.2.1.2	MA_L_F_Reporters					0
R4.2.2.1.3	MA_L_F_Elements	x	x	x		3
R4.2.2.2	MA_L_AlphabeticRangePagination	x	x	x		3
R4.2.3	MA_LibraryUsageStatistics					0
R4.2.4	MA_MapSearch					0
R5	Language					0
R5.1	L_Spanish	x			x	2
R5.2	L_English	x				1
R5.3	L_Galician	x	x	x		3

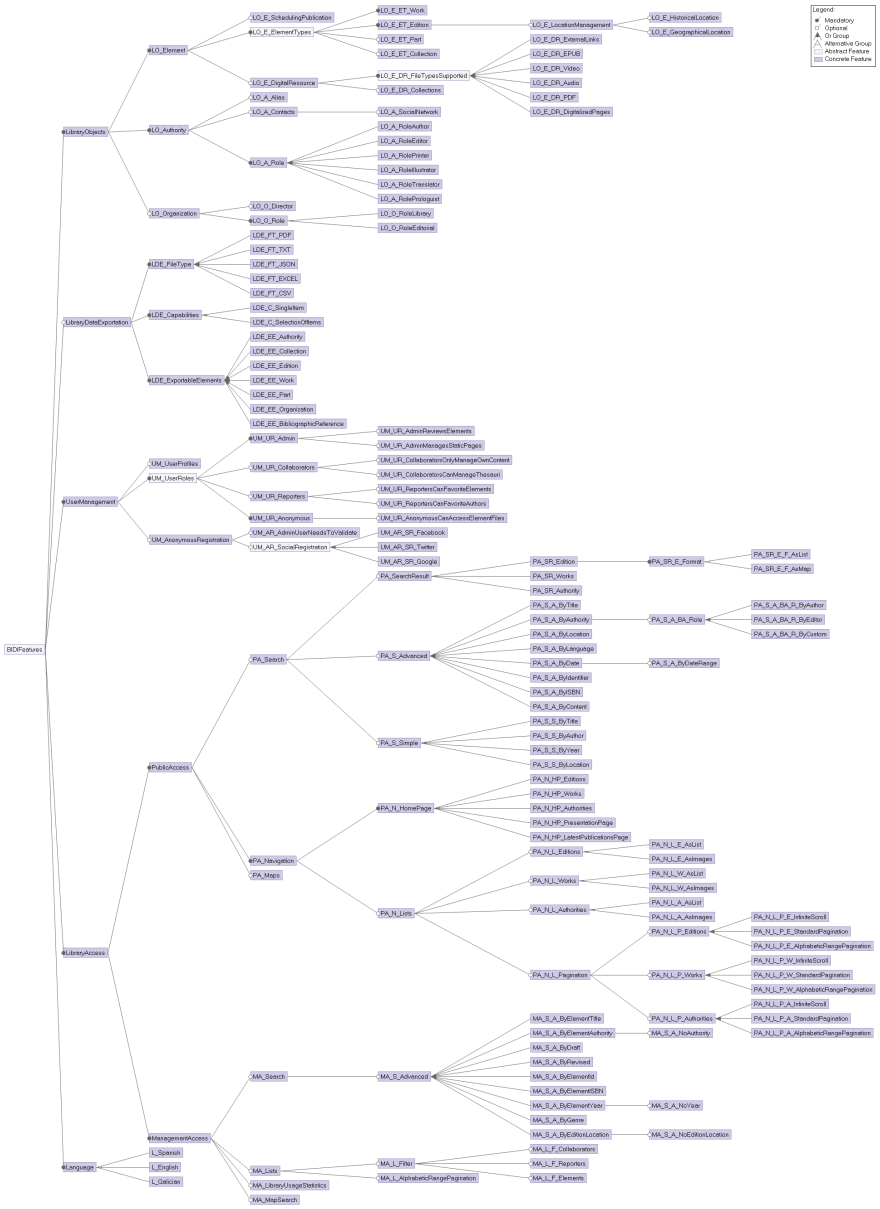


Figure 10 Complete feature model.

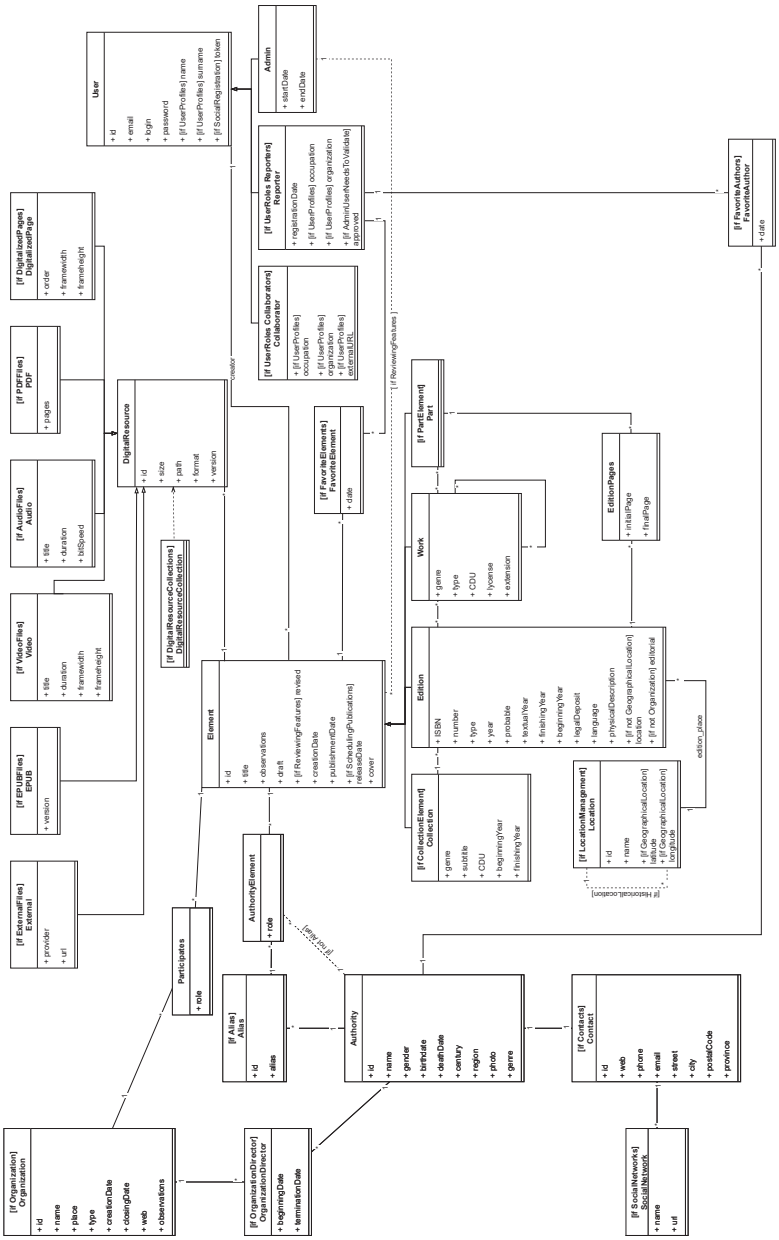


Figure 11 Complete data model.

References

[1] N. R. Brisaboa, M. J. Durán, C. Lalín, J. R. López, J. R. Paramá, M. R. Penabad, and A. S. Places. An architectural for virtual libraries. In *Proceedings of the 5th International Conference on Information Systems Analysis and Synthesis (ISAS'99)*, pages 512–517, Orlando, FL, 1999.

[2] N. R. Brisaboa, M. R. Penabad, A. S. Places, and F. J. Rodríguez. A system for the integrated access to digital libraries. In *Ingenieria del Software en la Decada del 2000*, pages 109–119. AECI-CYTED, Cartagena de Indias, 2003.

[3] Paul Clements and Linda Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2002.

[4] Alejandro Cortiñas, Miguel R Luaces, Oscar Pedreira, and Ángeles S Places. Scaffolding and in-browser generation of web-based gis applications in a spl tool. In *Procs. of the 21st International Systems and Software Product Line Conference-Volume B*, pages 46–49. ACM, 2017.

[5] Alejandro Cortiñas, Miguel R Luaces, Oscar Pedreira, Ángeles S Places, and Jennifer Pérez. Web-based geographic information systems sple: Domain analysis and experience report. In *Procs. of the 21st International Systems and Software Product Line Conference-Volume A*, pages 190–194. ACM, 2017.

[6] Jayant Deshpande. Digital libraries: An overview of standards, protocols and formats. In *International Journal of Library and Information Studies, ISSN: 2231-4911*, volume 8(1), Jan-Mar 2018.

[7] Space data and information transfer systems — Open archival information system (OAIS) — Reference model. Standard, International Organization for Standardization, September 2012.

[8] C. Lagoze, H. Van de Sompel, M. Nelson, and S. Warner. Open archives initiative protocol for metadata harvesting protocol. Technical report, January 2015.

[9] Library of Congress - Network Development and MARC Standards Office. Marc 21 format for bibliographic data. https://www.loc.gov/marc/bibliographic/, 2016. visited on 2020-09-21.

[10] Richa Pandey. Digital library architecture. In *DRTC Workshop on Digital Libraries: Theory and Practice, Bangalore, India, March, 2003; paper B*. DRTC, 2003.

[11] J. R. Paramá, A. S. Places, N. R. Brisaboa, and M. R. Penabad. The design of a virtual library of emblem books. *Software - Practice and Experience*, pages 473–494, 2006.

[12] A. S. Places, N. R. Brisaboa, A. Fariña, M. R. Luaces, J. R. Paramá, and M. R. Penabad. The galician virtual library. *Online Information Review*, pages 333–352, 2007.

[13] A. S. Places, N. R. Brisaboa, J. R. Paramá, O. Pedreira, and D. Seco. Managing the workflow of massive feeding of digital libraries. *Research in Computer Science*, pages 352–362, 2007.

[14] A. S. Places, A. Fariña, M. R. Luaces, O. Pedreira, and D. Seco. A workflow management system to feed digital libraries: proposal and case study. *Multimedia Tools and Applications*, pages 3843–3877, 2016.

[15] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer, Berlin, 2005.

[16] Delfina Ramos-Vidal, Alejandro Cortiñas, Miguel R. Luaces, Oscar Pedreira, and Ángeles Saavedra Places. A software product line for digital libraries. In Massimo Marchiori, Francisco José Domínguez Mayo, and Joaquim Filipe, editors, *Proceedings of the 16th International Conference on Web Information Systems and Technologies, WEBIST 2020, Budapest, Hungary, November 3-5, 2020*, pages 381–389. SCITEPRESS, 2020.

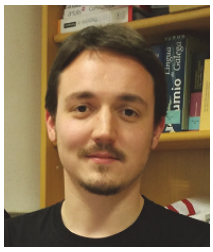
[17] Ccsds Secretariat and Navigation Office. Reference model for an open archival information system (oais). 2012.

[18] Jie Sun and Bao-Zhong Yuan. Development and characteristic of digital library as a library branch. *IERI Procedia*, 2:12–17, 2012. International Conference on Future Computer Supported Education, August 22- 23, 2012,Fraser Place Central - Seoul.

[19] Ian H Witten, David Bainbridge, and David M Nichols. *How to build a digital library*. Morgan Kaufmann, 2009.

Biographies

Delfina Ramos-Vidal is an Assistant Researcher at the Database Lab of the University of A Coruña, Spain. She graduated in Computer Science from the University of A Coruña in 2020 with an end of degree project entitled ”Tool for the semi-automatic generation of software for Digital Libraries”. She is currently enrolled in an M.Sc. in Data Science at the Polytechnic University of Madrid. Her research topics of interest include software product lines, data structures, and open data.



Alejandro Cortiñas is an Assistant Professor at the Database Lab of the Universidade da Coruña (Spain). He received a PhD in Computer Science from the same university in 2017 for his thesis, entitled “Software Product Line for web-based Geographic Information Systems”. His research topics of interest include software product lines, generative programming, geographic information systems, and spatial big data.

Miguel R. Luaces received his M.S. degree in Computer Science from the University of A Coruña (Spain) in 1998 and a European PhD in Computer Science from the University of A Coruña (Spain) in 2004. He undertook research in the area of spatial, temporal and Spatio-temporal databases at the FernUniversität Hagen (Germany) under the ChoroChronos project funded by the European Union. Today, he is an Associate Professor at the University of A Coruña, and he is currently a member of the Databases Laboratory of the University of A Coruña where he has been involved successfully in a number of research and development projects. His research interests include Geographic Information Systems, Spatial and Spatio-temporal Databases, Software Engineering, and Web-based Information Systems.

Oscar Pedreira has an M.Sc. and PhD degree in Computer Science from the University of A Coruña, Spain. He is an Associate Professor since 2008 at the same institution. He is a researcher of the Database Laboratory. His research interests include topics in databases (algorithms for similarity search, data structures and algorithms for graph databases, geographic information systems), and in software engineering (process improvement, testing, MDE, and SPL). He has co-authored many articles published in journals and conferences relevant to the research areas mentioned. He has continuously participated in research projects and technology and knowledge transfer projects with different companies.

Angeles Saavedra-Places is currently an Associate Professor at the Computer Science Department of the University of A Coruña. She received her PhD in Computer Science in 2003 from the same university. Her research interests are in the areas of Digital Humanities, Web Information Systems, Geographic Information Systems and Software Engineering. For further details about her CV see: http://lbd.udc.es/ShowResearcherInformation.do?id=5