

- ORIGINAL ARTICLE -

Towards Building Reuse-Based Digital Libraries for National Universities in Patagonia

Hacia la Construcción de Librerías Digitales Basadas en Reuso para Universidades Nacionales en la Patagonia

Alejandra Cechich¹, Agustina Buccella^{1,2}, Daniela Manrique¹, and Lucas Perez¹¹*GIISCO Research Group, Departamento de Ingeniería de Sistemas, Facultad de Informática, Universidad Nacional del Comahue, Neuquen, Argentina*

{alejandra.cechich,agustina.buccella,daniela.manrique,lucas.perez}@fi.uncoma.edu.ar

²*Consejo Nacional de Investigaciones Científicas y Técnicas - CONICET*

Abstract

This article presents a case study exploring the use of software product lines and reference models as mechanisms of a reuse-based design process to build digital libraries. As a key component in a modern digital library, the reference architecture is responsible for helping define quality of the resulting repository. It is true that many efforts have been addressed towards providing interoperability; however, repositories are expected to provide high levels of reuse too, which goes beyond that of simple object sharing. This work presents the main steps we followed towards building a reusable digital library capable of accommodating such needs by (i) providing mechanisms to reuse resources, and (ii) enabling explicit sharing of commonalities in a distributed environment.

Keywords: Software Product Lines, Digital Libraries, Reference Architectures, Delos Reference Model

Resumen

Este artículo presenta un caso de estudio que explora el uso de líneas de productos de software y modelos de referencia como mecanismos de un proceso de diseño basado en reuso para construir bibliotecas digitales. Como componente clave en una biblioteca digital moderna, la arquitectura de referencia es responsable de ayudar a definir la calidad del repositorio resultante. En la literatura se han realizado muchos esfuerzos para proporcionar interoperabilidad;

sin embargo, se espera que los repositorios proporcionen también altos niveles de reutilización, que van más allá del simple intercambio de objetos. Este trabajo presenta los pasos principales para construir una biblioteca digital reusable capaz de acomodar tales necesidades mediante dos actividades específicas (i) proporcionar mecanismos para reutilizar recursos, y (ii) permitir el intercambio explícito de aspectos comunes en un entorno distribuido.

Palabras claves: Líneas de Producto de Software, Librerías Digitales, Arquitecturas de Referencia, Modelo de Referencia Delos

1 Introduction

Developing architectures for digital libraries thinking of reuse is not a new concept. For more than a decade, researchers and practitioners have shown different approaches to modelling, from general to domain-specific libraries, such as the Alexandria Digital Earth Proto-Type (ADEPT) architecture [1], whose framework defines uniform client-level metadata query services that are compatible with heterogeneous underlying collections; the Digital Assets Repository (DAR) [2], whose third version shows a modular design including components and a content model for digital objects based on current standards [3]; or the reference architecture proposed by Candela, Manghi, and Pagano [4], whose architectural design pattern builds upon a type-based repository system capable of dealing with a federation of knowledge. However, even if such proposals are effective in some aspects, the process of building reusable repositories is not clearly addressed, letting aspects, such as reuse of policies and processes, out of the process.

By making use of specific techniques of software reuse, such as *software product lines* [5, 6, 7], it is possible to supply software functionality as optional modules, that can be added to the product at required locations. Applying this principle can overcome many current limitations concerning digital library reuse from

Citation: A. Cechich, A. Buccella, D. Manrique and L. Perez. "Towards Building Reuse-Based Digital Libraries for National Universities in Patagonia". Journal of Computer Science & Technology, vol. 18, no. 2, pp. 81 - 96, 2018.

DOI: 10.24215/16666038.18.e10

Received: December 4, 2017. **Revised:** June 7, 2018. **Accepted:** June 19, 2018.

Copyright: This article is distributed under the terms of the Creative Commons License CC-BY-NC.

early stages in a distributed context. For instance, policies may be encapsulated as reusable components that map to reusable procedures, which in turn map to reusable services. Then, like a puzzle, a particular location might compose its concrete repository by reusing existing components and extending the architecture according to its own preferences.

In this paper, we present the development of a software product line for digital libraries based on the definition of a reference architecture built upon different aspects that actors should address. As any other process for defining reference architectures, here we have selected a reference model as a starting point. A reference model is an abstract framework that provides basic concepts used to understand the relationships among items in an environment. On one hand, we looked at the DELOS Digital Library Reference Model [8], which was created with the aim of setting the foundations and identifying concepts within the universe of digital libraries. DELOS allowed us to start from six core concepts: *content* (data and information made available to users); *users* (the various actors entitled to interact with a digital library); *functionality* (the services offered to different users); *quality* (the parameters for evaluating the content and behaviour of a digital library); *policy* (the sets of conditions, rules, terms and regulations governing interaction between the digital library and users); and *architecture* (a mapping of the functionality and content offered by a digital library to hardware and software components). On the other hand, we have also taken into account our previous experiences building a software product line for a different domain (marine ecology), but with complex data management as well [9].

This paper presents the results of a research and development project held in Patagonia during 2010-2017, as part of a National initiative to move government and academic libraries into digital repositories. Two specific research questions framed this investigation:

1. What reuse needs do librarians believe would contribute to build distributed and integrated repositories among all universities in Patagonia?
2. How those needs could be achieved through architecting the underlying software by thinking of reuse?

This paper is organized as follows. In Section 2, we present the literature review. Then, we describe the main steps towards organizing work among the different parties, considering reuse from early stages and based on the reference model. Section 4 introduces our process through a case study developed in the context of Argentinean Universities. Final remarks are introduced in Section 5. Conclusion and future work are discussed afterwards.

2 Literature Review

Several Digital Library frameworks [10, 11], reference models [8] and repository software tools [12, 13, 14, 15, 16, 17] have addressed specific problems in Digital Library System architectural design and implementation. Of course, there are several efforts towards content reuse [18, 19, 20]; however, to situate our proposal of building a reusable architecture for a digital library, this section reviews existing proposals for distributed digital library architectures.

Distributed component architectures arrange components and contents spread across multiple locations. Fedora Commons [15] and Greenstone [13] are typical examples of digital library tools with a distributed architectural design. Fedora's architecture was designed to handle any type of digital content and its key strength is its inherent support for long term preservation. Fedora's distributed model also makes it possible for complex digital objects to make reference to content stored on remote storage systems. One of the features, which makes Fedora appealing to us, includes versioning, policy model, and object extensibility. However, lacks a dedicated user interface, and more importantly, it was not conceived for intensive reuse. Greenstone was designed for building and distributing digital library collections. Its architecture is decentralized, making the system scalable, flexible and extensible. The flexibility enables Greenstone to support distributed collections capable of being served from different machines, but at the same time maintaining a consistent presentation view to the end user.

Improvements to this architecture turned its design into a network of modules that communicate in terms of XML messages [14]. All modules characterize the functionality they implement in response to a describe messages, and can transform messages using XSLT to support different levels of configurability. This improvement aims at adding new collections and services adaptively, facilitating extensibility. Therefore, reuse is addressed from a concrete design view missing opportunities of systematic reuse from early stages. Greenstone is a service-based and dynamically configurable approach that can be found in other designs and systems, such as the Extended Open Archives Initiative protocol (XOAI) [21] and the OpenDLib system [22].

Finally, DSpace's architecture [17] is divided into three layers: application, business logic, and storage. This organization is, in some way, similar to ours in the sense we have split the model into layers to deal with the different aspects of an application. However, our layers are modeled following a product-line approach, so reusability is reinforced at every level. In addition, we used standards and models to contrast how the dynamics of building a digital library may change moving towards a completely reusable refer-

ence architecture.

As far as we know, an holistic approach covering domain as well as application analysis - like software product lines propose - is something new for building digital libraries.

3 Organizing Work Thinking of Reuse

Working packages were aligned to recommendations from the DELOS reference model [8], as Figure 1 shows, and optimized to keep two premises: delivery on time and reusing at every time.

Let us further describe working packages to clarify these points.

WP1: Management & Supervision. This package was in charge of defining schedules, and coordinate work among the different sites. It also evaluated quality of deliverables and supervised work. Deliverables included typical management reports.

WP2: Meeting & Survey. This package helped define elemental tasks to coordinate work and collect information about technological infrastructure and resources (content) from the different sites. Deliverables included local as well as integrated information. As Figure 1 shows, WP2 maps to the Content Domain of the reference model.

WP3: Diagnosis. This package helped us to elaborate a strategy to incrementally building by focusing on required services at local and global levels. Deliverables included a situation report and the strategy to build the following models collaborative and incrementally. As Figure 1 shows, WP3 was mapped to the Content Domain and the Functionality Domain of the reference model.

WP4: Policy & Process Models. This package focused on elaborating two different but related perspectives, which included strategic planning, assessment and improvement of policies and processes. They were further elaborated into a digital library policy model, a digital library management model, a digital library operation model, and a digital library support model. A process for defining, validating and using metadata was also included. As Figure 1 shows, tasks and deliverables of WP4 map to all domains of the reference model.

WP5: Infrastructure & Service Models. This package complemented WP4 by going deeper into two complex facets. On one hand, infrastructure addressed not only technological issues but also organizational ones such as staffing, furniture, room, etc.; and on the other hand, services included local as well as global requirements. Deliverables included a technological infrastructure model, an organizational

infrastructure model, a local service model, and a global service model. As Figure 1 shows, WP5 maps to Functionality, Quality and Architecture domains of the reference model.

WP6: Assessment Models. This package addressed quality properties and measures needed to evaluate quality of other models and the resulting architecture. Although some quality aspects were considered to define the models, WP6 stressed the point by producing a local and a global service assessment model. Indicators and metrics were suggested too. As Figure 1 shows, WP6 maps to the Quality Dimension of the reference model.

WP7: Software Product Line. This package helped us to finally draw the picture. Components from policy, procedures, infrastructure and services were composed all together to build a domain view of the product line. Then, they were specialized into the application domain as an implementation for the whole repository. Validation and deployment were addressed through pilot cases starting from common ones. As Figure 1 shows, WP7 maps to the Architecture Domain of the reference model.

4 Illustrating the Case of Systematic Reuse

Development Context and Participants

The work described in this paper was contextualised in a project, identified as PICT-O 2010-0139¹, supported by the *Agencia Nacional de Promoción Científica y Tecnológica* of the Argentine Republic. In this project, named RdiPatag², we described the necessary activities for developing reusable processes and procedures involved in the construction of a digital library. The participants of the project were members of four different universities located in Southern Argentina: *National University of La Pampa* (UNLPA)³, *National University of Comahue* (UNCOMA)⁴, *National University San Juan Bosco* (UN-SJB)⁵, and *National University of Patagonia Austral* (UNPA)⁶.

The team was a multi-disciplinary staff belonging to two areas of interest: libraries and informatics. The staff included in the first area, in general *librarians*, was involved in activities related to defining policies of storing and recovering digital documents, defining metadata, etc.; and was responsible for determin-

¹http://www.agencia.gov.ar/IMG/pdf/Res.330-11_PICTO_CIN_II_.pdf

²<http://rdipatag.wordpress.com/>

³<http://www.unlpam.edu.ar/>

⁴<http://www.uncoma.edu.ar/>

⁵<http://www.unp.edu.ar/>

⁶<http://www.unpa.edu.ar/>

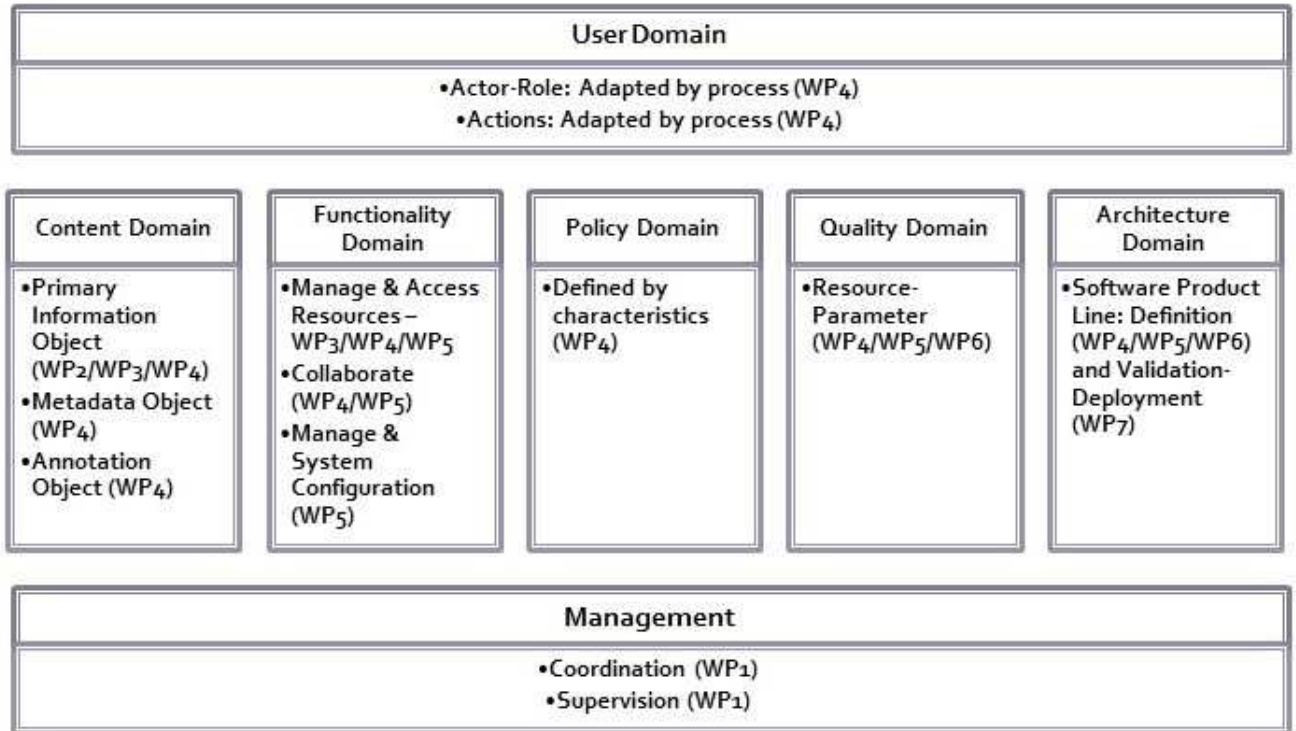


Figure 1: Mapping working packages to DELOS

ing the main requirements of the digital library. The second group, in general *software engineers*, was involved in activities related to defining services for implementing these requirements.

By considering the aspects previously mentioned, we defined the technical structure of the digital library. It consisted of four digital repositories, one for each university, connected through a federated layer. This structure was defined as part of the diagnosis activity of WP3.

Developing a Software Product Line for Digital Libraries

In this work, we applied the methodology defined in [9, 23] as an adaptation of several methodologies widely referenced in academy and industry [5, 6, 7]. Figure 2 shows the *domain engineering* and *application engineering* phases of our case. According to the methodology, the activities of the domain engineering are the following.

- Domain analysis

Information source analysis (ISA): This activity analyzes sources of information that can support the domain analysis in order to obtain a first set of requirements.

Subdomain analysis and conceptualization (SAC): The information recovered in the previous process is used to analyze and organize

the features or services that the subdomain should offer together with the general features derived from the upper domains. Also, in this process the subdomain must be conceptualized by different software artifacts (such as class models and process models) when it is possible. *Reusable component analysis (RCA)*: This process identifies the set of reusable components that could be used to implement the features defined in the last process. It returns a preliminary reference architecture.

- Organizational analysis

Reuse and boundary analysis (RBA): This activity defines the organizational boundary, commonality, and variability features. Thus, by considering the features specified in the subdomain analysis and conceptualization process and the information from domain experts, the scope of the product line must be defined.

Platform analysis and design (PAD): This activity builds the reference architecture based on the features defined in the previous activities and processes. The preliminary structure of reusable components defined in the reusable component analysis process is reorganized and refined. Here, each component with its common and variable parts (when necessary) is fully designed.

Then, in the application engineering phase, in

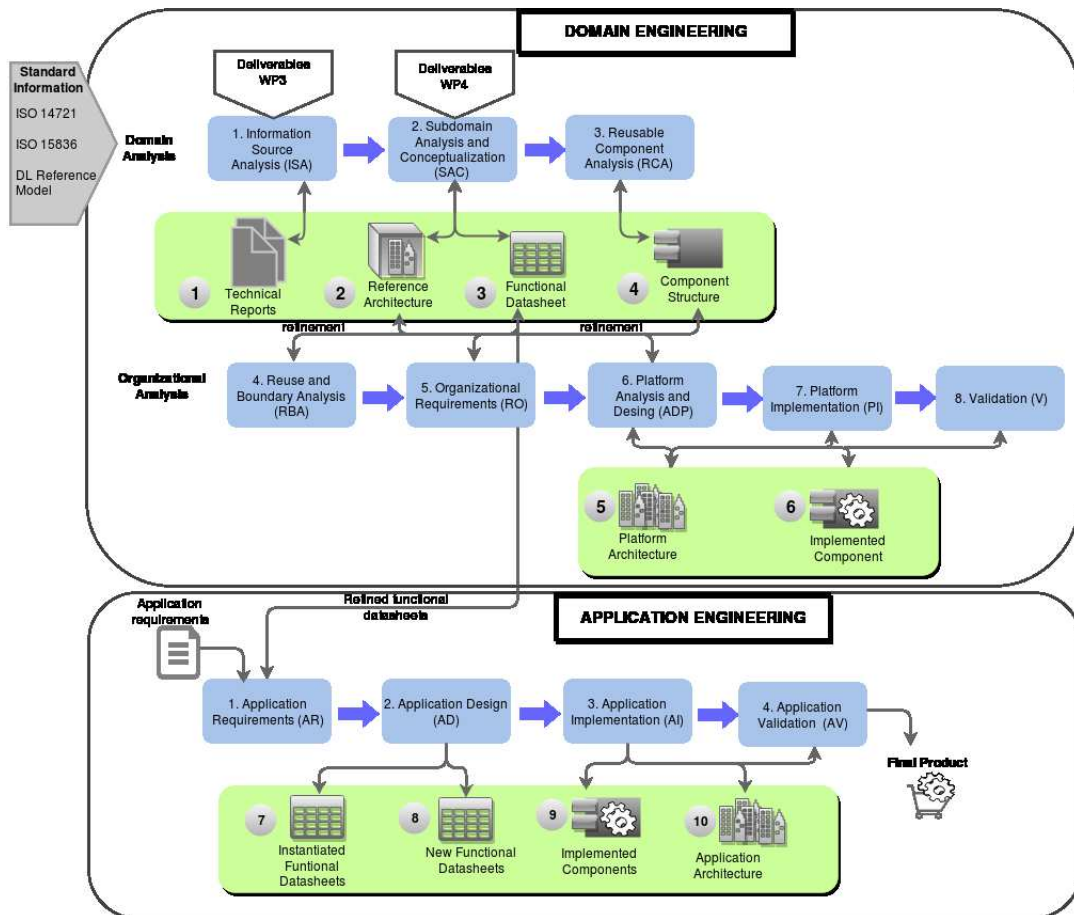


Figure 2: Activities of the domain and application engineering phases

which we perform the specific activities for developing new products from the line, we must consider the following four activities:

- **Application requirements engineering (AR):** This activity must retrieve the specific requirements for a particular organization or application by considering the reusable domain requirements.
- **Application design (AD) and implementation (AI):** By taking into account the reference architecture and the specific requirements of an organization, the activity must define and implement the application's architecture. It selects and configures the reusable components of the reference architecture and adds specific adaptations. In this activity, the variabilities defined for the reusable components must be bounded in order to fix the specific functionality of the resultant product.
- **Application testing (AT):** This activity must validate and verify an application against its specification.

In Figure 2, we can observe on one hand, the eight activities of the domain engineering phase together

with the software artifacts (numbered from one to six); and on the other hand, the four activities of the application engineering together with four artifacts (numbered from seven to ten). Another important aspect of the figure is the big arrow on the left side. It denotes the influence of the standards on the domain analysis activities.

4.1 Our Case: Domain Engineering Phase

1. Information source analysis. This process must gather information and analyze three types of sources within the library domain: standards, existing applications and librarians as domain experts.

Firstly, information was obtained from ISO standards and the DELOS Reference Model. The ISO 14721 [24] and DELOS introduce techniques and processes to define reference models for storing, managing and publishing digital information. On one hand, the ISO 14721 defines, in a more general and abstract way, a reference model for an open archival information system (OAIS) describing common services and responsibilities for this type of systems. On the other hand, DELOS provides a more specific scenario for defining digital libraries by introducing a framework composed of three types of systems: Digital Library, Digital Library System, and Digital Library Manage-

ment System. The manifesto describes the main aspects of these systems focusing on functionality, architecture, and actor roles (end-user, designer, administrator, etc.), among others. Finally, the ISO 15836 [25] defines the Dublin Core metadata standard⁷, in which a set of vocabulary terms are introduced to describe different types of resources, such as physical or web resources.

Secondly, we analyzed whether the universities had used some software tools for managing digital information. We observed that universities had no applications or processes to manipulate this information. They only had some web pages (available on Internet) to allow end users to download some type of documents without well-defined rules. Also, here we analyzed the deliverables of WP3 (represented in Figure 2 by a big arrow on the top of this activity) in which an analysis about the real situation of the universities was described. These deliverables were designed as technical reports describing two main aspects: available resources, such as technical infrastructure and personnel, and policies and procedures about manipulating digital information. Technical reports represent the first software artifact created in the domain engineering phase (Figure 2, 1).

2. Domain analysis and conceptualization. In this process we must analyze and organize the information recovered in the previous activity according to the features and services that the library domain should offer.

In our project, we defined a preliminary set of policies and procedures, as part of WP4 deliverables, in which we describe specific requirements and restrictions of the domain. In Table 1 we can see some policies defining manipulation and management of digital information. These policies introduce a framework to regulate the operation of the digital repositories. They provide a clear view of roles and responsibilities of the parties involved, as well as comply with the guidelines of the National Digital Repositories⁸. As we can see in Table 1, policies are classified into five main categories: policy framework, content and object collection management, resource deposit, use and access, and copyright and intellectual property.

Table 1 enumerates each policy included in one category together with a goal description. For example, policies defined in the second category, *content and object collection management*, describe valid content of the documents, the way they are organized and procedures for their management. Each document's content includes the research and academy production generated by professors, researchers and special-

Table 1: Policies defined as part of WP4

Categories	Policies	Description
Policy framework	Intra-university policies	Strategies and activities to be performed for guaranteeing the correct management and use of digital documents within a university
	Inter-university policies	Strategies and activities to be performed for guaranteeing the correct management and use of digital documents among universities
Content and object collection management (COCM)	Collection structures	Aspects about mandatory information included in each collection
	Document type	Allowed types of documents together with the metadata associated to each one
	Collection management	Permissions, revisions, validations, backups, etc.
Resource deposit (RD)	Direct deposit	Requirements about the direct deposit of documents
	Mediated deposit	Requirements about the mediated deposit of documents
	Deposit type	File formats depending on the document types
Use and access (UA)	User types	User types and well-defined roles
	Access type	Access types according to the roles
Copyright and intellectual property (CIP)	License type	Licenses used to publish documents
	Document category	Classification of type of documents according to the review process submitted

ist of each university. In particular, the policy document type describes the documents to be accepted, such as articles, books, book chapters, etc., and metadata requested for each case. This last information was extracted from the ISO 15836 standard. At the same time, we defined a set of procedures, each one describing the set of services needed to implement a set of policies (a procedure can implement more than only one policy).

All the previous information was classified and analyzed in order to create a *reference architecture* (second software artifact of Figure 2). In this case, a layer-based approach was chosen, in order to include particularities of the library domain. Thus, we defined a *user interaction layer* for grouping services related to the user interaction, a *processing layer* for grouping transactional services, and a *model layer* in-

⁷<http://dublincore.org/>

⁸http://repositorios.minicyt.gob.ar/pdfs/Directrices_SNRD_2012.pdf

cluding services for database access. In Figure 3, we show the main global components belonging to each layer. Also we can see a *federation layer* with specific components to manage the federation aspects. This layer implements federated components, including those responsible for solving possible incompatibilities among different repositories. These incompatibilities emerge due to products, in spite of they were derived from the same platform, can add new functionalities that are specific of each product. In this way, the components of the federated layer include syntactic and semantic mechanisms in order to find possible conflicts that must be identified and solved to build an integrated environment.

At the same time, we modeled the information retrieved from the technical reports and the library domain particularities into *functional datasheets* (third software artifact of Figure 2). These artifacts were defined in previous works [9, 23] to model domain functionalities by designing interactions among variant and common services. In this work, we adapted the functional datasheets in order to store and model all information about procedures. The template used for classifying these procedures is shown in Table 2. It contains, for each procedure, an identification, such as a number or code; a textual name describing the main function; a procedure type classifying the procedure as generic or specific, considering that generic procedures describe services that are needed by a set of specific procedures; the policies implemented; the list of services involved for fulfilling the procedure; a graphical notation consisting of a set of design artifacts (generally represented as UML diagrams) representing service interactions graphically; and a set of XML files specifying the design artifacts. For these two last items, we must define the set of dependencies that allow us to represent the interactions. These dependencies involve the common interactions among common⁹ and variant services.

Id	Identification of the procedure
Name	Textual Name of the procedure
Procedure Type	Whether the procedure is generic or specific
Policies Implemented	Set of policies the procedure implements
List of Services	List of services used to fulfill the procedure
Graphical Representation	Graphical notation showing service interactions
XML Files	XML files representing the services and their interactions

Table 2: Template for domain procedure definition

In order to produce machine-readable dependencies (from the Orthogonal Variability Model (OVM) proposed in [6]), we defined a set of XML tags. The

⁹Common services are services which will be part of every product derived from the SPL

dependencies represented are (Table 3):

Use: specifying a dependence between common services, which are not necessarily associated with a variation point.

Mandatory variation point: determining the selection of a variant service when the variation point is included.

Optional variation point: specifying that zero or more variant services, associated to the variation point, can be selected.

Alternative variation point: defining that only one variant service, of the set of associated variants of the variation point, must be selected (XOR relation).

Variant variation point: defining that at least one variant service, of the set of associated variants of the variation point, must be selected (OR relation).

Requires: specifying a relation between two variant services independent from the variation points the variants are associated with, in which the selection of one variant service requires the selection of the other.

Excludes: which is the opposite of the *requires dependency* specifying the exclusion of a variant when another one is selected.








Dependency	XML Tag	Graphical Notation
Use	<code><Use></code>	
Mandatory variation point	<code><MandatoryVP></code>	
Optional variation point	<code><OptionalVP></code>	
Alternative variation point	<code><AlternativeVP></code>	
Variant variation point	<code><VariantVP></code>	
Requires	<code>dependency:Requires = "serviceName"</code>	
Excludes	<code>dependency:Excludes = "serviceName"</code>	

Table 3: Set of dependencies used to model procedures

Differently from OVM, which suggests employing XML tags to mark text fragments, we defined three types of XML documents for representing datasheets. The first one, named *service interactions* is generated to represent the graphical service interactions defined in a datasheet. The second type of XML documents is the *service information* containing the service id, the textual description, and the name of the architectural component in which it is included. Then, for each service involved in a *service interaction* XML file, a link to the *service information* XML file must be included. Finally, the third type of XML documents is the *variability constraint* which describes the variability constraints imposed to the services. Thus, for each

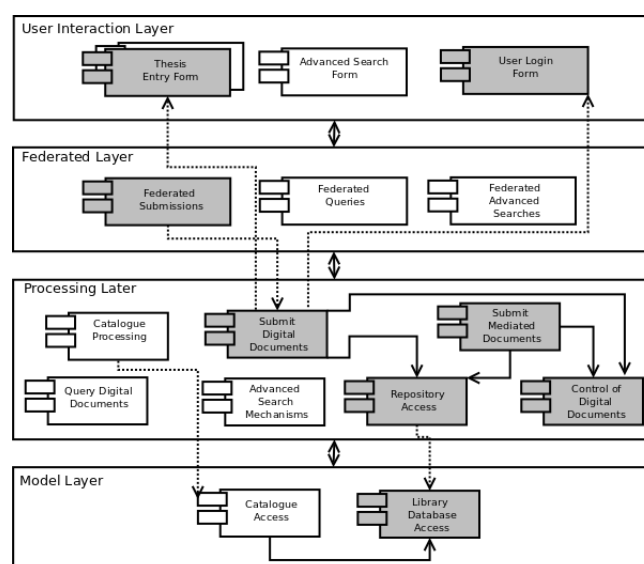


Figure 3: Reference Architecture

required functionality of the domain, one template is completed by generating the functional datasheets with a set of XML files.

In order to illustrate the set of resources described previously, Table 4 shows the *Submit Thesis Datasheet* in which we can see the information related to the submit thesis specific procedure to submit a digital thesis by an author.

Id	STD1
Name	Submit Thesis
Procedure Type	Specific
Policies Implemented	RD.DT3 - RD.DD2 - UA.AT1 - COCM-DT1
List of Services	user authentication, selecting communities, selecting collections, etc.
Graphical Representation	Figure 4
XML Files	To be defined during the organizational analysis

Table 4: Submit Thesis Datasheet

As we can see in the Table, the procedure implements several policies related to the document type (COCM.DT), deposit type (RD.DT), use and access (UA.AT), and direct deposit (RD.DD). At the same time, it uses a set of services that can be inherited from generic procedures previously defined. In order to show this different granularity among procedures, we include, as part of the graphical notation, workflow diagrams that represent each procedure. For example, the workflow for the submit thesis procedure is shown in Figure 4. Here we can see that generic procedures (procedures 2, 3, and 6) are represented by simple boxes involving filled out, control and licenses. The specialized procedures are defined as overwritten ones (represented by highlighted boxes) showing the

specific activity that is involved. Thus, the Submit Thesis specific procedure overwrites procedures 1, 4, 5 and 7 in order to allow content producers to authenticate (defined by the access type policy); fill out metadata included in the thesis (defined by the document type policy); define the embargo over the document (defined by the use and access policy); and upload the file in source and pdf format (defined by the deposit type policy). The output of the procedure is a document, physically stored in a repository, which is set with pre-published state.

3. Reusable component analysis. This process identifies the set of reusable services that could be used to implement the procedures defined in the last process. Here, a preliminary structure (forth software artifact of Figure 2) composed of possible reusable components must be included. For example, the Submit Thesis specific procedure can be specified as a Submit Thesis service implemented as part of the Submit Digital Documents component. This preliminary structure will be then modified during the organizational analysis.

4. Reuse and boundary analysis. This activity defines the organizational boundary, commonality, and variability features. Thus, by considering the services specified in the domain analysis and the information from domain experts, we defined the scope of the product line.

Firstly, we performed a detailed analysis and design of each procedure defined in the functional datasheets of the domain analysis. The most important activity was to define the commonality and variability included in each of them. For example, for the Submit Thesis procedure, two of the overwritten pro-

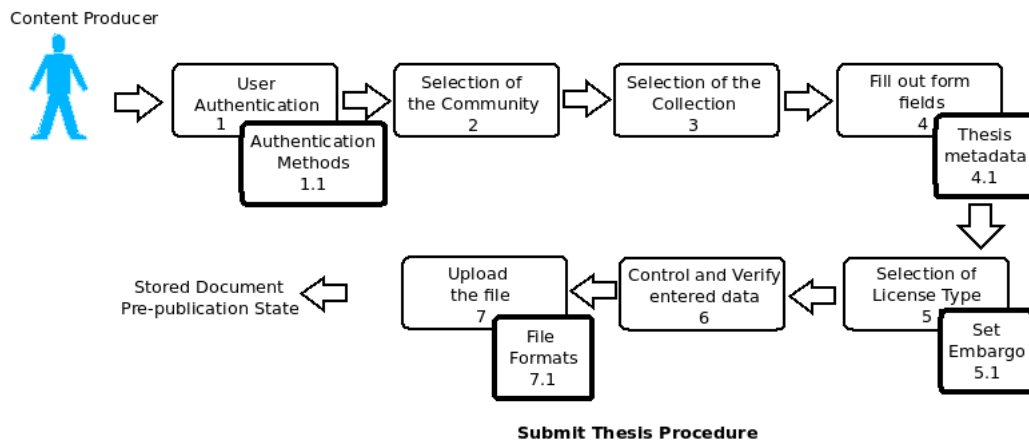


Figure 4: Submit Thesis specific procedure

cedures (1 and 7 of Figure 4) implement these extensions by means of variability definition. Thus, both user authentication and file format have variant services making the implementation of these policies be adapted by each university.

In order to design specific dependencies among common and variant services, we built two new diagrams. These diagrams are part of the graphical representation item of the submit thesis datasheet (Table 4). Firstly Figure 5 shows two variability diagrams associated to a sequence diagram. The first one is associated to the user authentication procedure, being the login authentication service defined as mandatory and the network access authentication service as optional. The other variability represents the way thesis are uploaded, being always requested in postscript format and optionally in source format.

Then, the second diagram (Figure 6) represents the dependencies within the layered reference architecture, previously defined. On the left of Figure 6 we can see the service model diagram, according to the variabilities and dependencies also represented in Figure 5. This model was built by using a design tool, named *Datasheet Modeler*, developed in a previous work [26] for supporting service models and their variabilities. Specifically, the tool was created for allowing designers to specify dependencies, variabilities, graphical representations and XML transformations. Once a designer performs the service model diagram in the *Datasheet Modeler* tool, it automatically derives XML files representing, in a computer-readable format, the service information and interactions. The XML files generated are part of the XML files item of the datasheets (Table 4). On the right of Figure 6, we can see the XML files generated by the tool. For this service model, the tool derived fourteen service information XML files corresponding to the fourteen services, and one service interaction XML file for representing layers and the way services interact to each other.

5. Organizational requirements. In this activity, we used the information of the commonality and variability identified in the last activity, the information provided by the domain analysis and conceptualization, and reusable component analysis processes. The main goal here is to define the range of products and services that the line is able to implement.

In our project, we defined a product/service matrix indicating which subset of services will be part of the product line and which subset of services will be product specific. In this work, we followed a minimalist approach, that is, only the features used in all products are part of the SPL. Thus, our SPL is then seen as a platform [7]. A part of this product/service matrix can be seen in Table 5 in which we show some of the services of the Submit Thesis procedure. For example, services as *uploading files* and *thesis form* are part of the platform because they are required by all products; rather, services such as *source file* and *user authentication by network* will be part only of some products (Products 2 and 4).

6. Platform analysis and design. This activity builds the reference architecture based on the services and components defined in the previous activities and processes.

Firstly, we reorganized and redefined the preliminary structure of reusable components defined in the reusable component analysis (fourth software artifact of Figure 2) in order to design the final platform architecture (fifth software artifact of Figure 2).

To do so, we took the XML files previously generated (Figure 6) as inputs of the *Component Derivator* tool [27]. This is a semi-automatic tool that analyzes possible scenarios of service interaction models and applies a set of predefined rules to generate the final reference architecture. The tool analyzes specifically the architectural components and dependencies of ser-

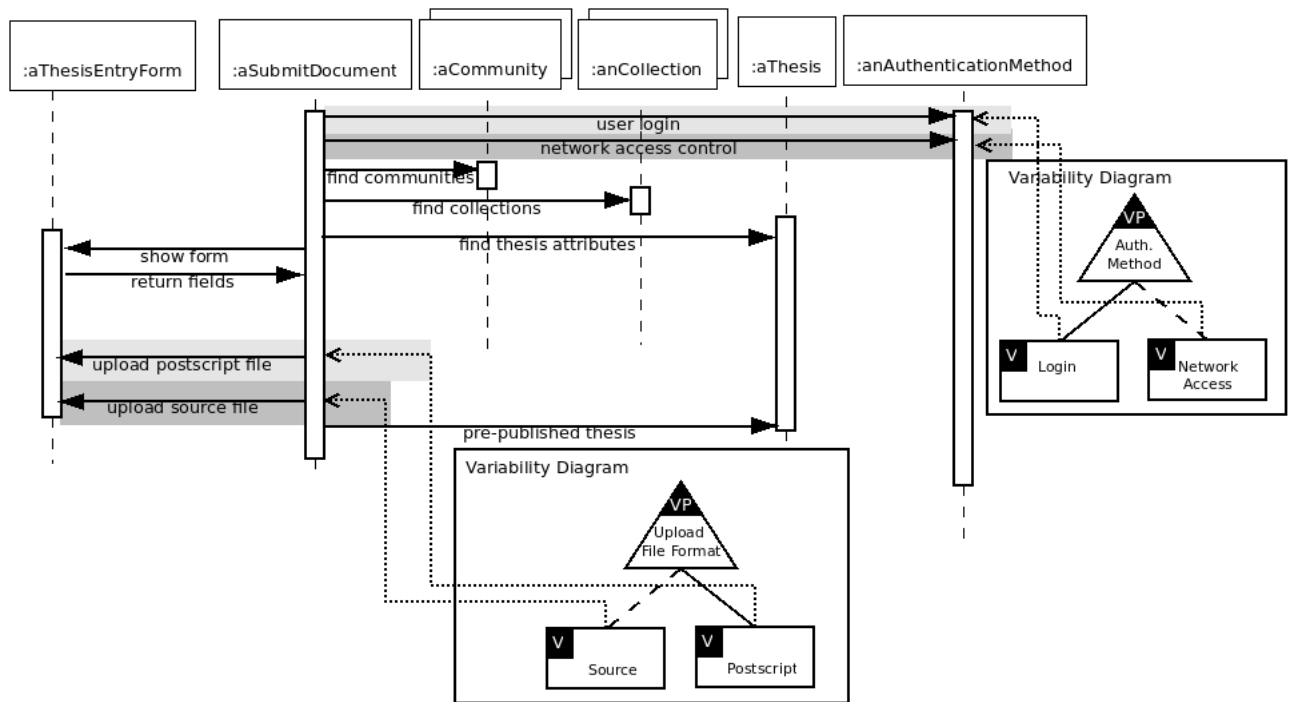


Figure 5: Variability model associated to the sequence diagram of the Submit Thesis procedure

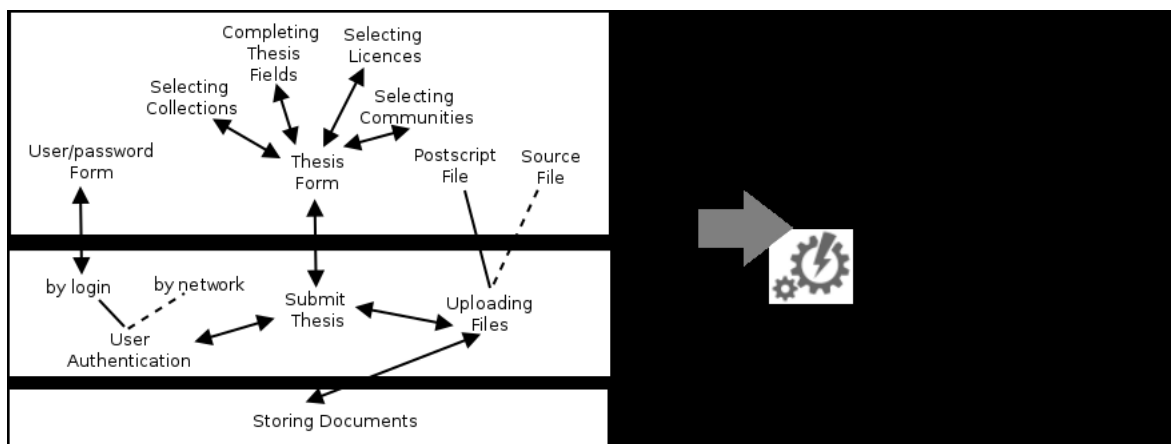


Figure 6: Service model diagram associated to the reference architecture of the Submit Thesis procedure

Services	Product1 (UNComa)	Product2 (UNPA)	Product3 (UNSB)	Product4 (UNLa)
completing thesis fields	X	X	X	X
user/password form	X	X	X	X
selecting licences	X	X	X	X
selecting communities	X	X	X	X
thesis form	X	X	X	X
postscript file	X	X	X	X
source file	X			X
user authentication by login	X	X	X	X
user authentication by network		X		X
submit thesis	X	X	X	X
uploading files	X	X	X	X
storing documents	X	X	X	X

Table 5: Product/service matrix for services included in the Submit Thesis Procedure

vices in order to define a component structure that can be implemented. Once the tool specifies this structure, the designer must determine the final structure to be implemented.

By following the submit thesis procedure, in Figure 7 we can see the software components generated within the reference architecture. For example, for the case of the user authentication service, the tool proposed a software component that implements both variabilities within the same layer (see Figure 6). The user/password form was separated as another component because it is part of the User Interaction layer. At the same time, user authentication, submit thesis and uploading files services are into different components due to they are also used by other procedures (or included in other datasheets) such as submitting journal documents, proceedings, etc. Something similar happens with the thesis entry form and selecting communities/collection/licenses components because they will be reused by other procedures.

7. Platform implementation. In this activity, components that are common for all products, that is, components of the line are implemented. In particular, in our project, as part of WP2 and WP3 deliverables, we performed an analysis of software tools for implementing digital repositories. In that study, we analyzed at least six different digital repositories and at the same time, we compared those to the digital repositories actually implemented in the Argentine Republic. From them, we considered that DSpace provides the better environment for reuse and interoperation issues. At the same time, it provides an architecture well-documented and flexible for being adapted to our policies and procedures previously defined.

First of all, previous to implement components, we performed a re-engineering of DSpace in order to determine the way our reference architecture fitted into the DSpace architecture. In other words, it was necessary to know which procedures were completely supported by DSpace and which of them must be imple-

mented by extending parts of DSpace's components. It was a complex task, because each software component had to be identified within the DSpace architecture in order to determine whether the procedures were completely or partially supported. In order to show complexity and to see the magnitude of the work, we analyze here the user authentication component of our architecture (shown in Figure 7) versus the authentication/authorization component of the DSpace architecture. In DSpace, the authentication is implemented as a stack in which several methods can be added. Comparatively, in our procedures, the authentication can be made by user/password (mandatory) and by network access (optional) and we do not allow any other authentication options. Remember that the network access service was inside our user authentication component (Figure 7).

In this way, we had to restrict the authentication/authorization component of DSpace in order to provide a common platform for the SPL. Thus, when the SPL is instantiated (see Section 4.2) only the allowed methods can be selected. This adaptation of the DSpace architecture was performed by changing the configuration property *plugin.sequence.org.dspace.authenticate.AuthenticationMethod* that defines the authentication stack. As this file is a comma-separated list of class names, each of them implementing a different authentication method, we modified it in order to give support for *Authentication by Password* (class : *org.dspace.authenticate.PasswordAuthentication*) as default, and *IP Address based Authentication* (class : *org.dspace.authenticate.IPAAuthentication*) as optional. In Figure 8 we can see the documentation of this adaptation for the authentication methods. Another aspect that we can see in the figure is that we had to provide a correspondence between layers of both architectures. Fortunately, the three layers of both architectures have the same semantic meaning grouping the same set of functionalities; only a different organization of components are implemented by each of them.

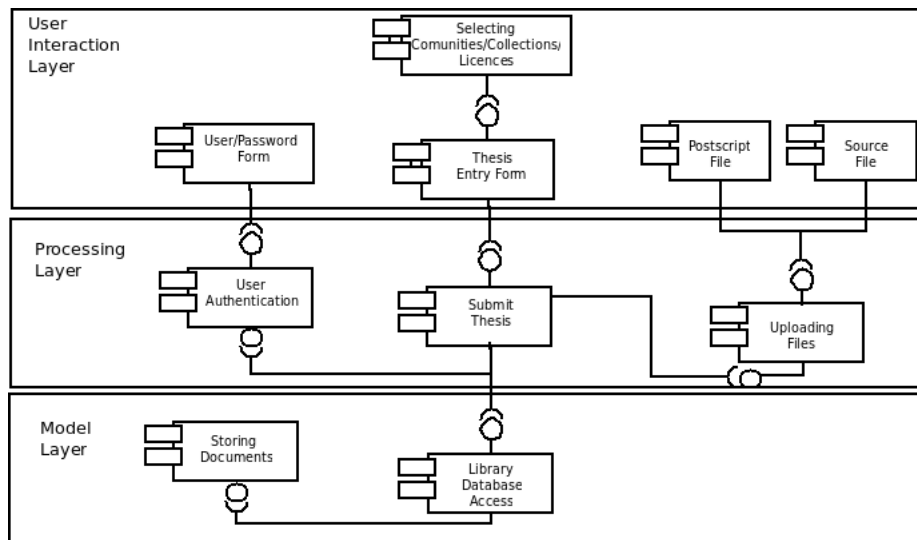


Figure 7: Reference architecture and components for the Submit Thesis procedure

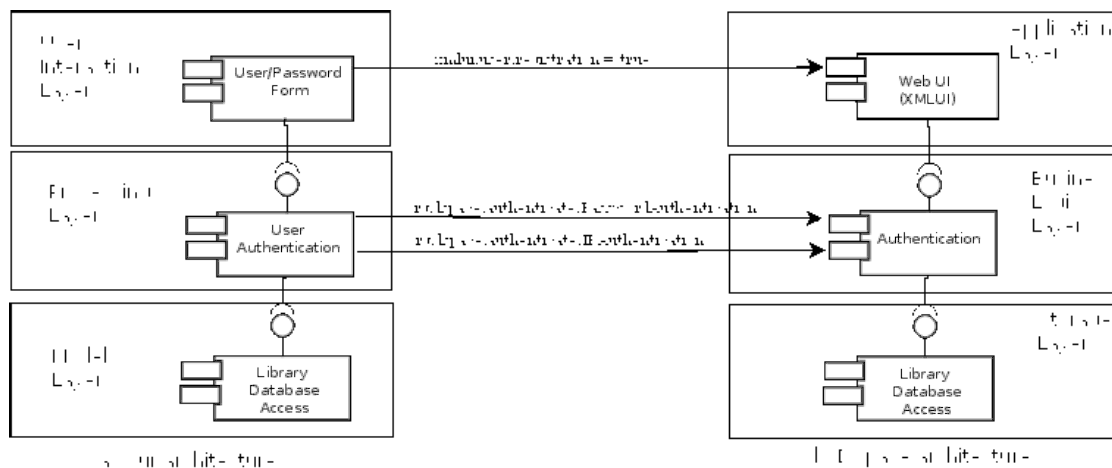


Figure 8: Documentation of correspondences between our architecture and DSpace for the authentication component

The output of this activity is the *Library SPL* which conforms the platform for the four universities involved.

4.2 Our Case: Application Engineering Phase

In order to show an instantiation of the product line, here we describe the activities performed by the UNCOMA University for creating a new product based on the SPL platform. To do so, we followed the activities defined in the application engineering phase of Figure 2.

1. Application requirements. In the first activity, we captured the specific requirements of the university and we obtained a set of requirements – some of them included in the services implemented in the line, and some others were new ones. For our case, a new requirement emerged in order to support a special case of licenses.

2 and 3. Application design and implementation. In this activities, we instantiated the reference architecture into an application architecture by performing two main activities: binding the variability and implementing specific components.

For the UNCOMA repository, the variability defined in the *Submit Thesis* procedure was bound by using the login and password authentication service and by the two options for uploading files, in postscript and source file formats. Figure 9 shows the service model diagram of the final documentation for this procedure. In addition, we had to create new specific components according to the new requirements. Particularly, we had to develop one specific component in order to support the service for allowing users to choose different types of licenses when a thesis is ready to be published. In Figure 11 we can see the user authentication page that should be used for user login when submitting and querying thesis of different collections (faculties). The repository is available at <http://rdi.uncoma.edu.ar>

4. Application validation. Here, the specific software product was validated. To do so, we analyzed two main aspects of any SPL development: *reusable component development* and *product development*¹⁰. The former was analyzed from the point of view of the reuse capability given by the opinion of domain experts and addressing the procedures specified in the platform. For this, a validation of some procedures was carried out in their common and variable aspects with participants from all the sites involved. Also within this aspect, the effort made to adapt DSpace to the reuse needs of the platform was

¹⁰http://www.sei.cmu.edu/productlines/frame_report-meas_tracking.htm

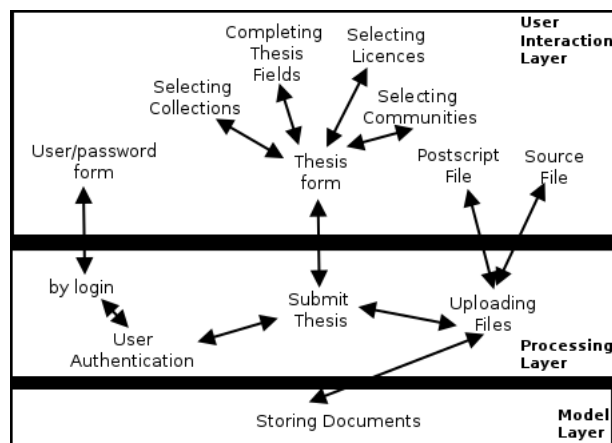


Figure 9: Service model diagram of the Submit Digital Thesis procedure binding for the UNCOMA product

analyzed. Here, aspects such as existence of documentation of DSpace, knowledge of the developers, flexibility of the tool, etc. were taken into account. As a result of both analyzes, we reach the conclusion that any implementation of our proposal will require a careful selection of participants (previous experience in component-based software development, knowledge in DSpace, etc., significantly alters adaptation times of DSpace); and it will require a careful selection of procedures to be implemented (preferably selected by their incremental complexity). The latter, product development, was based mainly on the time required to derive the specific product (in our case *submit thesis*), taking into account again the developers' capabilities and the time invested (without considering prior learning for the use of the platform, since it had been considered in the previous aspect). As preliminary results, flexibility and speed in the elaboration of the product showed that the approach can be highly beneficial by reducing development efforts.

5 Discussion

Reuse-oriented models can reduce the effort of software development when comparing with other methods. However, moving into reuse is not a panacea. It is not always practical because the collection of components may not be available, developing with reuse may not be an established practice, or selecting services for reuse requires domain knowledge is included in some way. Our proposal addressed those issues by modeling through a product line following an integral perspective of the digital library domain. From this perspective, we elaborate some key points:

- An integral perspective for developing reuse-based digital libraries should include mechanisms for reusing different assets – from policies to architectural components (including of course, object collections). To do so, a developer may look



Figure 10: Web page for authentication in the digital repository of the UNCOMA product

at reference models (as our case does) or proposing a set of policies, procedures and services that constitute the domain rules.

- Reuse models should be thought as linked abstractions that connect the different assets. Therefore, not only software will be reused but also decisions, activities, roles, and regulations. It means that the adopting organization should be ready to think of itself as a set of building blocks to be composed. We realize that this is one of the hardest parts of our proposal, which we addressed by sharing diverse experiences, and building commitment through participation. It implied considerable leadership and management.
- Once reuse artifacts are built, selecting the appropriate combination requires knowing the services a library system can provide. Interaction among stakeholders is the basis for extracting and modeling this knowledge in the way the development team considers more efficient. In our case, service model diagrams along with datasheets were used for communication and documentation, reducing the gap between librarians' and software developers' backgrounds.
- Selecting the implementation platform will impact on reuse. Reusing third-party software components requires additional efforts in understanding the components, their interfaces, and the way they can be adapted (and eventually extended). We mitigated this risk by using a well-known platform (DSpace); however complexity of reengineering/adapting any platform implies a considerable amount of work that should not be neglected.

6 Conclusion and Future Work

A digital library is a library in which collections are stored locally, or accessed remotely via computer net-

works. In the last case, and coming back to our research questions, we found that librarians as well as software developers agreed on the needs of integrating not only documents but also policies and procedures. Then, our answer to the second question (How those needs could be achieved through architecting the underlying software by thinking of reuse?) led us to use reference models and reuse-based development.

This paper introduced the main steps we followed towards building this systematic approach. Preliminary validation has shown promissory results; however, we are aware that further experimentation is needed to wide-spreading. Our actual efforts are focused on providing quantitative measures to both process and product quality.

Acknowledgements

This work was partially supported by the UNCOMA project 04/F009, and the PICTO CIN Project 2010-0139

Competing interests

The authors have declared that no competing interests exist.

References

- [1] G. Jane and J. Frew, "The ADEPT digital library architecture," in *JCDL '02 Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pp. 342–350, 2002.
- [2] I. Saleh, N. Adly, and M. Nagi, "DAR: A Digital Assets Repository for Library Collections," *Research and Advanced Technology for Digital Libraries, LNCS*, vol. 3652, pp. 118–127, 2005.
- [3] Y. Mikhail, N. Adly, and M. Nagi, "DAR: institutional repository integration in action," in

- TPDL'11 Proceedings of the 15th international conference on Theory and practice of digital libraries: research and advanced technology for digital libraries*, pp. 348–359, 2011.
- [4] L. Candela, P. Manghi, and P. Pagano, “An Architecture for Type-based Repository Systems,” in *Second Workshop on Foundations of Digital Libraries, in conjunction with 11th European Conference on Research and Advanced Technologies on Digital Libraries (ECDL 2007)*, 2007.
- [5] J. Bosch, *Design and use of software architectures: adopting and evolving a product-line approach*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000.
- [6] K. Pohl, G. Böckle, and F. J. v. d. Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.
- [7] F. van der Linden, K. Schmid, and E. Rommes, *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.
- [8] L. Candela, G. Athanasopoulos, D. Castelli, K. E. Raheb, P. Innocenti, Y. Ioannidis, A. Katifori, A. Nika, G. Vullo, and S. Ross, “The DELOS Digital Library Reference Model - In a Nutshell,” tech. rep., DL.org Consortium, 2011.
- [9] A. Buccella, A. Cechich, M. Arias, M. Pol’la, M. del Socorro Doldan, and E. Morsan, “Towards systematic software reuse of GIS: Insights from a case study,” *Computers & Geosciences*, vol. 54, pp. 9–20, 2013.
- [10] R. Kahn and Wilensky, “A framework for distributed digital object services,” *International Journal on Digital Libraries*, vol. 6, pp. 115–123, 2006.
- [11] M. Goncalves, E. Fox, L. Watson, and N. Kipp, “Streams, structures, spaces, scenarios, societies (5s),” *ACM Transactions on Information Systems*, vol. 22, pp. 270–312, 2004.
- [12] R. Tansley, M. Smith, and J. H. Walker, “The DSpace Open Source Digital Asset Management System: Challenges and Opportunities,” *Research and Advanced Technology for Digital Libraries, LNCS*, vol. 3652, pp. 242–253, 2005.
- [13] D. Bainbridge, G. Buchanan, J. McPherson, S. Jones, A. Mahoui, and I. Witten, “Greenstone: A platform for distributed digital library applications,” in *ECDL 2001, Research and Advanced Technology for Digital Libraries, 5th European Conference*, pp. 137–148, 2001.
- [14] D. Bainbridge, K. Don, G. Buchanan, I. Witten, S. Jones, M. Jone, and M. Barr, “Dynamic digital library construction and configuration,” in *ECDL 2004, Research and Advanced Technology for Digital Libraries 8th European Conference*, pp. 1–13, 2004.
- [15] D. Bainbridge and I. Witten, “A FEDORA librarian interface,” in *8th ACM/IEEE-CS joint conference on Digital libraries, Pittsburgh PA, PA, USA, June 16-20*, pp. 407–416, 2008.
- [16] A. Pepe, T. Baron, M. Gracco, J. Le Meur, N. Robinson, T. Simko, M. Vesely, and J.-y. L. Meur, “CERN Document Server Software: the integrated digital library,” in *ELPUB 2005 conference, Heverlee (Belgium)*, pp. 8–10, 2005.
- [17] DSpace, “The DSpace Developer Team. DSpace 3.x Documentation.” 2013. <https://wiki.duraspace.org/display/DSDOC3x>.
- [18] J. H. Canos, M. I. Marante, and M. Llavador, “SLiDL: A Slide Digital Library Supporting Content Reuse in Presentations,” in *Research and Advanced Technology for Digital Libraries*, pp. 453–456, 2010.
- [19] J. Diederich and W.-T. Balke, “The Semantic GrowBag Algorithm: Automatically Deriving Categorization Systems,” in *European Conference on Digital Libraries*, pp. 1–13, 2007.
- [20] M. Gahegan, R. Agrawal, T. Banchuen, and D. DiBiase, “Building rich, semantic descriptions of learning activities to facilitate reuse in digital libraries,” *International Journal on Digital Libraries*, vol. 7, pp. 81–97, 2007.
- [21] H. Suleman and E. A. Fox, “Designing protocols in support of digital library componentization,” in *European Conference on Digital Libraries*, pp. 568–582, 2002.
- [22] D. Castelli and P. Pagan, “A system for building expandable digital libraries,” in *The third ACM and IEEE joint conference on Digital Libraries*, pp. 335–345, 2003.
- [23] A. Buccella, A. Cechich, M. Pol’la, M. Arias, S. Doldan, and E. Morsan, “Marine ecology service reuse through taxonomy-oriented SPL development,” *Computers & Geosciences*, vol. 73, no. 0, pp. 108 – 121, 2014.
- [24] ISO/IEC, “ISO 14721: Space data and information transfer systems - Open archival information system - Reference Model,” 2003. International Standard.
- [25] ISO/IEC, “ISO 15836: Information and documentation - The Dublin Core metadata element set,” 2009. International Standard.

- [26] M. Mancuso, A. Buccella, A. Cechich, M. Arias, and M. Pol'la, "Datasheet modeler: Una herramienta de soporte para el desarrollo de funcionalidades en líneas de productos de software," in *Proceedings of the CACIC'15: XXI Argentine Congress of Computer Science*, (Junin, Argentina), 2015.
- [27] M. Arias, A. Buccella, and A. Cechich, "Towards semi-automatic component derivation from an spl variability model," in *Proceedings of the CONAISI: III Argentine Congress of Informatics Engineering*, (Argentina), 2015.