ABSTRACT
The volume of generated and processed information in organizations leads to the need to seek solutions that enable greater organization and provision of such information in digital form. Document Management Systems (DMS) play an important role, enabling the automation of processes which contributes to cost reduction, human labor, processes’ cycle time and information flows on paper.

Document management systems fall within the field of content management systems, where can be found several offers available in the market with different characteristics and specialized in different types of content. This heterogeneity requires that the various content management systems implement standards to enable its interoperability.

This paper presents the solution adopted for the development and implementation in edoclink document management system, a service interface based on the CMIS (Content Management Interoperability Services).

Categories and Subject Descriptors

General Terms
Management, Documentation, Standardization

Keywords
Document Management Systems; Interoperability; edoclink; Content Management Interoperability Services

1. INTRODUCTION
The work presented in this paper was developed in company Link Consulting, in Lisbon, in units within the development, support and projects teams of edoclink system.

The problem of this project focuses on interoperability applied to edoclink document management system. Edoclink is an integrated solution that fits in a DMS area, covering management of the entire document life cycle, from capture and indexing, through management of access and version control, to its final destination or elimination.

The edoclink document management system has a proprietary web service interface, which is exposed to most of the features that the system offers, based on the system's business model. This services interface allows that third-party applications can interact and integrate with edoclink system enabling the automation of tasks that otherwise would have to be performed by users in different systems. The development processes of integration typically require a learning period, by the third teams involved, requiring prior knowledge of the model and system business concepts.

The project’s aim is to provide the edoclink system of a service platform designed for interoperability between content management systems, specifically the CMIS (Content Management Interoperability Services). The CMIS aims to facilitate access to and exchange of digital assets between different systems using Web Services and Web 2.0 interfaces on neutral Internet protocols and independent of the vendor of systems involved. Thus, the objective of this work is to make the analysis of the CMIS specification and design a mapping model between its data model and the edoclink system data model, ending with the creation and implementation of the technical solution to make available the new interface in the system.

2. RELATED WORK
2.1 Content Management
The concept of Enterprise Content Management (ECM) is defined by AIIM as the set of “the strategies, methods and tools used to capture, manage, store, preserve, and deliver content and documents related to organizational processes.” [1] The document management is a subdomain of the enterprise content management, focused on the management of the document’s specific type of content. The document management differs from the content management on the probative and informative value of the documents it incorporates, however, nothing prevents both from sharing common rules and principles. [2]

Figure 1. Relationship between Content Management, Document Management and Records Management

The ISO 15489-1:2001 standard defines Records Management as “the field of management responsible for the efficient and systematic control of the creation, receipt, maintenance, use and disposition of records, including the processes for capturing and maintaining evidence of and information about business activities and transactions in the form of records.” [3]

A Record is defined by the ISO 15489-1:2001 standard and the NP 4438-1:2005 standard like being “the information created, received, and maintained as evidence and information by an organization or person, in pursuance of legal obligations or in the transaction of business.” [3] [4]

According to the above definitions, the terms “document” and “record” may be considered equivalent, since a document
corresponds to information that is recorded on a specific support, not only as evidence, but also with information value for the one or organization.

It is estimated that about 80% of information and knowledge of an organization is stored in its documents in unstructured form, and only 20% in structured storage formats. [5] This shows us the importance of the Document Management Systems (DMS) in information and knowledge management on the organizations. A DMS can provide the right information to the right person at the place and time required.

2.2 Interoperability
Interoperability is defined as the ability of different applications and information systems to communicate with each other and work together, exchange data and use the information exchanged within organization or across organizational boundaries. [6] [7]

To be interoperable, information systems must be able to exchange and reuse data, following standard models, presenting them in a correct way and noticeable to users, while preserving its meaning.

Interoperability can be classified at the following levels [8]:

a) Syntactic interoperability
   Ability of systems to communicate and make data exchange, ensuring the integration at the technical level, infrastructure, media, transport, storage and data representation. [7]

b) Semantic interoperability
   Facilitates the encoding, transmission and use of information among the various actors, ensuring understanding of the information by own systems and users. [9] [10]

c) Organizational interoperability
   Enhances cooperation between organizations, leveraged by the compliance processes, channels, motivations and other elements that facilitate the achievement of common goals. [11]

2.3 Specifications and related implementations
2.3.1 WebDAV
Web Distributed Authoring and Versioning (WebDAV) was the first standard established in the area of content management, published in February 1999 and is supported by most content management systems. WebDAV is a set of extensions to Hypertext Transfer Protocol (HTTP) that enables the creation, editing and collaborative management of the content on a web server, making this a server with remote support capabilities of reading and writing data. [12]

2.3.2 Content Repository for Java
The Java Content Repository (JCR) is a standard API for accessing repositories. The standard began to be drawn by the Java community in 2002, publishing the first version in 2005 (JSR-170) and a second in 2009 (JSR-283). The aim of the JCR is to provide a common vocabulary for access to the content repository, allowing a reduction in the complexity associated with proprietary interfaces of the content management system vendors. The JCR allows an application to use the bidirectional API to access content stored and managed in different repositories, abstracting from their specific features.

2.3.3 MoReq2010 Export Service
The MoReq2010 is a specification of requirements for records management systems, published in May 2011 by the DLM Forum. The specification is intended to provide a modular set of simple requirements and easy to understand, so it can be adapted and applied in different types of organizations with different information models. [13] The specification composes sets of requirements associated with each module, grouped by services. A service means the ability to perform tasks that together constitute a coherent functionality, both the entity's point of view that provides the service as the entity that requests it. [2]

Figure 2 shows the set of services considered essential by MoReq2010 for the operation of a document management system.

![Figure 2. MoReq2010 services architecture](https://www.jcp.org/en/jsr/detail?id=283)

The export service has a set of features that give the system the ability to export records to another document management systems, maintaining associated metadata and properties of integrity, authenticity, reliability, confidentiality and usability (technical and semantic interoperability).

2.3.4 CMIS
Content Management Interoperability Services is an OASIS (Organization for the Advancement of Structured Information Standards) standard that defines a domain model and Web Services, Restful AtomPub and browser (JSON) bindings that can be used by applications to work with one or more Content Management repositories.

The CMIS interface is designed to be layered on top of existing Content Management systems and their existing programmatic interfaces. It is not intended to prescribe how specific features should be implemented within those CM systems, nor to exhaustively expose all of the CM system's capabilities through the

---

1 JSR 170 - https://www.jcp.org/en/jsr/detail?id=170

CMIS interfaces. Rather, it is intended to define a generic/universal set of capabilities provided by a CM system and a set of services for working with those capabilities. [14]

3. CMIS – CONTENT MANAGEMENT INTEROPERABILITY SERVICES

The CMIS resulted from the need to integrate the various existing content management systems in an organization. Typically an organization has multiple content management systems, specialized in different types of content, such as Document Management, Digital Asset Management, Web Content Management, etc. All these systems have their own proprietary API, based on different technologies. The diversity of APIs difficulties the development of services that aggregate information from these systems, complicates the transfer and sharing of content between systems and prevents easy development of user interfaces that allow aggregated query of the contents. [15]

A client application, which adopts CMIS specification, is able to communicate with a variety of systems using only a service oriented standard interface. The content management system provides access to its proprietary formatted content and metadata through the CMIS services interface.

Figure 3. CMIS services interface access to different systems [15]

3.1 Domain model

CMIS specifies a core data model that defines the persistent information entities that are managed by the repository. CMIS provides a generic object model making it possible to implement by all content management vendors. The entities managed by CMIS are modeled as typed objects. There are five primary base types of objects:

- **Document**
  A document object represents a standalone information asset. Document objects are the elementary entities managed by a CMIS repository.

- **Folder**
  A folder object represents a logical container for a collection of "file-able" objects, which include folder objects, document objects, policy objects, and item objects. Folder objects are used to organize fileable objects. Whether or not an object is file-able is specified in its object-type definition.

- **Relationship**
  A relationship object represents an instance of a directional relationship between two objects. The support for relationship objects is optional.

- **Policy**
  A policy object represents an administrative policy, which may be "applied" to one or more "controllablePolicy" objects. Whether or not an object is controllable is specified in its object-type definition. The support for policy objects is optional.

- **Item**
  An item object represents a generic type of CMIS information asset. Item objects are not versionable and do not have content streams like documents but have properties like all other CMIS objects. The support for item objects is optional.

Additional object-types may be defined in a repository as subtypes of these base types.

An object must have one and only one primary object-type, which cannot be changed. An object may have zero or more secondary object types applied to it. A secondary type is a named marking that may add extra properties to an object in addition to the properties defined by the object's primary type. Secondary object-types can only be defined as subtypes or descendant types of the cmis:secondary base type.

Figure 4. CMIS Object Types and Property Types

All CMIS objects have a set of properties defined by the object type. A document object may have a content stream, which may be used to hold a raw digital asset such as an image or a word-processing document. A document object may also have one or more renditions associated to it. A rendition can be a thumbnail or an alternate representation of the content stream. Objects may have one Access Control List (ACL), which controls access to the object. An ACL represents a list of Access Control Entries (ACEs). An ACE in turn represents one or more permissions being granted to a principal (a user, group or role).
Figure 5 shows a representation of CMIS domain model, with its five primary types (Document, Folder, Relationship, Policy and Item) and Secondary type.

![CMIS Model Diagram]

### 3.2 Services

The CMIS specification defines a set of services available on the service interface to access and manage objects in the repository. All services must be implemented in the three connection protocols, and so can be differences derived from the technical features associated with the protocol. The technical details of each service, such as the parameters, returned data and possible exceptions, are available on the CMIS specification [14]. Below is listed the set of services:

- **Repository services**: These services are used to discover information about the repository, including repository IDs (could be more than one repository managed by the endpoint), capabilities (many features are optional and this is the way to find out if they are supported or not), available object types, and descendants. It could also support creating new types dynamically on the fly. The repository service methods are `getRepositoryInfo`, `getRepositoryList`, `getAllowedActions`, `getSearchableObjectTypes`, `createType`, `updateType`, and `deleteType`.

- **Navigation services**: These services are used to navigate the folder hierarchy in a CMIS repository, and to locate documents that are checked out. The navigation service methods are `getChildren`, `getDescendants`, `getFolderTree`, `getFolderParents`, `getObjectParents`, and `getCheckedOutDocs`.

- **Object services**: These services provide ID-based CRUD (Create, Read, Update, and Delete) operations on the objects in a repository. The object service methods are `createDocument`, `createDocumentFromSource`, `createFolder`, `createRelationship`, `createPolicy`, `createItem`, `getAllowableActions`, `getObject`, `getProperties`, `getObjectByPath`, `getContentStream`, `getRenditions`, `updateProperties`, `bulkUpdateProperties`, `moveObject`, `deleteObject`, `deleteTree`, `setContentStream`, `appendContentStream`, and `deleteContentStream`.

- **Mutifiling services**: These services are optional; they make it possible to put an object into several folders (mutifiling) or outside the folder hierarchy (unfiling). This service is not used to create or delete objects. The mutifiling service methods are `addObjectToFolder` and `removeObjectFromFolder`.

- **Discovery services**: These services are used to look for queryable objects within the repository (objects with the property queryable set to true). The discovery service methods are `query` and `getContentChanges`. The query method uses a language based on SQL-92 grammar.

- **Versioning services**: These services are used to manage versioning of document objects, other objects not being versionable. Whether or not a document can be versioned is controlled by the `versionable` property in the object type. The versioning service methods are `checkOut`, `cancelCheckOut`, `checkIn`, `getObjectOfLatestVersion`, `getPropertiesOfLatestVersion`, and `getAllVersions`.

- **Relationship services**: These services are optional and are used to retrieve the relationships in which an object is participating. The relationship service method is `getRelationships`.

- **Policy services**: These services are optional and are used to apply or remove a policy object to an object which has the property `controllablePolicy` set to true. The policy service methods are `applyPolicy`, `removePolicy`, and `getAppliedPolicies`.

- **ACL services**: These services are used to discover and manage the access control list (ACL) for an object, if the object has one. The ACL service methods are `applyACL` and `getACL`.

### 4. PROBLEM ANALYSIS

The edoclink can be seen as a content management system specialized in document management, offering features that cover the entire document life cycle, from their capture to their final destination. The system is supported on Microsoft.Net technology for Web environment with a SQL database server and a filesystem document repository.

The edoclink business model is based on the following concepts related to the management of documents:
• **Card**: It is the structure that ensures the registration and indexing of documents managed by the system; is the main system metadata repository.

• **Book**: It allows the definition of cards by type of documentation. These are the units in which the cards are grouped, with the aim of allowing the organization of the documents, the numbering of the cards and their accesses.

• **Process**: Folder/file concept, which allows the aggregation of a number of documents related to a particular subject, from multiple cards or distributions.

• **Classification**: It is the hierarchical cataloging a card or process, depending on their scope and thematic nature. A classification has a name and number of levels, which represent to its class hierarchy. Classes can be attached to records or processes, according to the configuration of the hierarchical level to use.

• **Distribution**: Workflow that is a decision-making process, which involved a set of users, in sequence or in parallel. The route of distribution, made up of steps, can be set by the time of its creation or be built step by step. Thus responsible for the last step can prolong it by a set of new steps.

### 4.1 Object types mapping

In edoclink CMIS interface, there is only exposed types of objects directly related to document management. All types associated with workflow features are not, for now, exposed in the interface. Thus, the following object types are relevant:

- **Class**
  The objects of class type are known in edoclink as “classifications”, they are used to “organize” the card’s type objects in a logical and hierarchical structure defined according to the organization’s classification plan. The class object type maps directly to the cmis:folder type.
  The definition of the type class has properties that allow the characterization of each instance by specifying the objects identity, hierarchical relationship with his father, behavior, attributes and access privileges of their children. In the class object you can indicate what types of children that this can include: processes and/or cards. When an object class can contain processes, this object is seen as a typification of these processes, becoming an object type that can be instantiated giving rise to processes, mapped to specific subtypes of cmis:folder type.

- **Book (type of card)**
  A book in edoclink is a name given to a type of record, the cards are always instantiated in a particular book. The type of card has properties that characterize the documents and they define access privileges for users and groups. This types are mapped to cmis:secondary subtypes. The cmis:secondary subtypes are then associated with cmis:document objects.

- **Document**
  The document type is the information base unit in edoclink system, specifying the set of properties directly related to the physical document (binary) and versioning. This type maps directly to cmis:document.

Figure 6 represents the mapping between the types of objects in edoclink and types of objects displayed in the CMIS interface.

![Figure 6. Mapping object types](image)

We chose to use the CMIS secondary types to represent the types of edoclink cards. These secondary types are associated with objects of type cmis:document, adding the properties of their edoclink book and also the properties of edoclink cards.

On edoclink, a card may be associated with more than one document, and a document may be associated with more than one card.

![Figure 7. Relationship between card and document](image)

To work around this limitation, it was decided to expose at the CMIS interface only on object of type cmis:document for each combination of edoclink “card – document”, where the document plays the differentiating criterion. With this decision, they are not exposed in the CMIS interface the cards that do not have at least one association to a document.

### 4.2 Architecture

The edoclink logical architecture follows the following division by layers:

- **Data Layer**, where the configuration data and metadata objects are stored.
- **Business Layer**, where all functional rules that manage the objects and access to data from the data layer are implemented.
- **Presentation layer**, which is the native web interface connected to the business layer.
- **Web services layer**, where lies the implementation of proprietary services interface, based on the business layer
Following this architectural model, the implementation of CMIS services is localized in a new "layer", identical to the web services layer in form of connection and access to the rest of the model.

The NCMIS library was chosen because of their greater architectural flexibility by allowing the abstraction of data access in relation to the service logic, and approaching to the needs of the interface to be implemented.

The NCM library defines an architecture that abstracts access to a repository from the interface services, based on a domain model with the CMIS objects and business rules. The library uses the design pattern Provider Model [16] to provide an API and a contract which developers can use to develop repository integration logic. This approach, using the library allows development to focus on the creation of the pieces (providers) that convert NCMIS domain objects to the edoclink domain model.

The mapping model designed for the implementation of the CMIS services interface in edoclink system requires the support of version 1.1 specification. This need is due to the use of secondary types to represent the edoclink cards by adding additional properties to documents. Thus, it was necessary to evolve the library with support for data model and services of the CMIS 1.1 specification version. The development focused on the new cmis:secondary type and on the library domain model, a new class was created, TypeSecondaryDefinition, which inherits from the base class TypeDefinition. This new class represents the type definition of cmis:secondary type and subtypes.

Analyzing the characteristics of the three libraries, the VB.NET was excluded due to using a different language from that used in edoclink. Among the remaining two, the library Documentis.Net own partial support of version 1.1 of the CMIS, however its architecture is inflexible to adaptation into a repository already implemented, as is the case of edoclink, due to having only one layer which is implemented serialization logic of the services interface and data access logic.

Table 1. Available .NET libraries

<table>
<thead>
<tr>
<th>Library name</th>
<th>Development language</th>
<th>Supported bindings</th>
<th>CMIS version</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCMIS</td>
<td>C#</td>
<td>RESTful, AtomPub</td>
<td>1.0</td>
</tr>
<tr>
<td>VB.CMIS</td>
<td>VB.NET</td>
<td>RESTful, AtomPub</td>
<td>1.0 / 1.1 (partially)</td>
</tr>
<tr>
<td>Documentis.Net</td>
<td>C#</td>
<td>SOAP, WebServices</td>
<td>1.0 / 1.1 (partially)</td>
</tr>
</tbody>
</table>

3 http://sourceforge.net/projects/dnetcmis/
4 http://ncmis.codeplex.com/
interacts directly with the business layer, implementing the translation logic of CMIS objects, and their properties, in edoclink objects. This logic of translation and mapping is implemented in providers designed on the model of NCMIS library. These providers are located in a new component, named "EdoclinkCMISProvider". Figure 11 illustrates the decomposition of the components of the CMIS services layer, and interaction with the edoclink business layer of the architecture.

![CMIS Services Layer Diagram](image)

**Figure 11. CMIS services layer components**

6. VALIDATION

Validation of the solution used the CMIS Workbench tool as a consumer client application of the CMIS services interface of the edoclink system. The CMIS Workbench\(^6\) application has been developed under the Apache Chemistry\(^7\) project of the Apache Software Foundation. This project aims to provide open source implementations of CMIS, serving as guidance model with best practices for the adoption of the interface on client applications and content repositories. The CMIS Workbench is a generic client that can be connected to any repository having a service interface compatible with CMIS.

![CMIS Workbench Image](image)

**Figure 12. CMIS Workbench**

The demo environment used the RESTful AtomPub binding. This binding is available in edoclink system in the relative URL 

"[host]/edoc/cmis/atom/". Examples executed covers the following operations: get the service document and repository information; get object types definition; repository navigation; and object creation.

7. CONCLUSIONS

This project was born from the need to provide on a document management system, the edoclink, a services interface that meets standards and interoperability specifications. The choice of CMIS was due to its current acceptance and adoption by various content management systems, and its flexibility of adaptation and independence related to the repository's developing technology. Thus, was set itself the aim of this project as being the analysis, design and development of an interface service, for the edoclink system, which complies with the CMIS specification.

The implementation of the CMIS services interface in edoclink began with a thorough analysis of possible mapping scenarios between edoclink data model and the CMIS data model. Before the solution presented in this article, were equated other possible mappings, set aside due to limitations found in the statement of edoclink objects in the interface. The solution presented here focuses on the objects and functions directly related to the documents and their organization in edoclink repository, achieving a compromise between the edoclink supported features and limitations imposed by the specification.

From a technical point of view, we tried to use components and architectural design models already established, with the aim of optimizing the available resources. In this sense, the design of the technical solution is based on NCMIS library, which was the one that offered better adaptability to the characteristics edoclink system.

The prototype still requires some work to this new interface can be considered to be finalized and included in edoclink product. However, all the work of analysis and design made possible to have a clear vision of what is a CMIS services interface, their capabilities, limitations and evolution capabilities. The designed mapping model will allow the interface to be evolved and consolidated over the next development cycles of edoclink product, which new functionality is included in the roadmap as an evolution strategy.

7.1 Future Work

The development project of a CMIS service interface in edoclink system is in a very early stage. The limitation of available time resources for the development of the solution prevented the same reach the desired level of stability, having focused on building a prototype that bore the proof of concept and validation of the feasibility of the project solution. Accordingly, in the future it is intended to consolidate service interface, continuing development work operations and mapping logic, setting the modulus of the CMIS services in edoclink for integration into a commercial version of the product, thus being available for use by their customers.

Being the data model of CMIS specification pretty flexible, it may be interesting to evaluate and explore the possibility of exposing in the CMIS services interface other features of edoclink system, such

\(^6\) https://chemistry.apache.org/java/developing/tools/dev-tools-workbench.html

\(^7\) https://chemistry.apache.org/
as the objects and data associated with workflows supported by the system, the "distributions " and their "stages". To achieve this objective, a suggestion left for future analysis is the use of the "cmis:item" object type, extending it in order to map the various concepts present in edoclink distributions, using the object type "cmis:relationship" to represent the documents connection to the stages of distributions.

In the future we intend to contribute to the NCMIS project, to provide the community developments that are being added to this library. As expected future developments, it is necessary to consolidate the support of CMIS version 1.1, introducing the connection protocol based on RESTful with JSON format, as well as operations, types of objects and properties missing. This contribution will certainly allow other .NET technology-based solutions that can use the library for their implementations.

8. REFERENCES

[1] AIIM. What is Enterprise Content Management (ECM)?


