FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Content Diffusion in ALERT® Clinical Applications

Igor José Martins Amado

PROVISIONAL VERSION

Report of Project/Dissertation
Master in Informatics and Computing Engineering

Supervisor: Gabriel David

Responsible for Monitoring: Tiago Silva

July 2009

Content Diffusion in ALERT® Clinical Applications

Igor Amado

Report of Project/Dissertation
Master in Informatics and Computing Engineering

Approved in oral examination by the committee:

Chair: Nome do presidente do júri (Title)

_____

External Examiner: Nome do arguente do júri (Title)

Internal Examiner: Gabriel David

29th July 2009

Confidential

Abstract

The product ALERT® is a suite of software applications that is filled with various types of Contents such as types of analysis, diagnoses, exams, procedures.

In the past, these Contents were managed through Excel files and different databases, distributed by various sources, and then loaded into the ALERT® applications using manual processes. This form of maintenance and processing of information was not automated and does not centralize the universe of Contents of ALERT®. To solve this and other problems, it was created the content repository, which centralizes all the Content information of ALERT®.

As key objective for this project, ALERT wants a mechanism for content diffusion between that Content Repository and the various ALERT® clinical applications.

The model of the repository is based on the common clinical data set approach, conceptual network, which has a design generic enough to be able to represent anything that is necessary. By examining how the industry supports these processes of diffusion, we found that different technological decisions lead to different forms of implementation. Particularly with regard to diffusion, we conclude that the definition of a process, appropriate tools and technologies are essential to its effectiveness.

In this sense a technical analysis was done in detail, focusing on the content repository, and the suite of products ALERT®, as regards the representation of content. The results of this analysis identified that the flexibility allowed by the repository's model, means certain mappings and transformations in the content diffusion process, as models of data are quite different.

A solution was presented as a web application that has an underlying Configurations module, with log and history, responsible for the management of metadata, mappings and transformations, necessary for allowing interoperability between Content Repository and other databases or documents metadata. To implement the proposed solution, a fully functional prototype was developed for serving as a proof of concept.

After implementation of the solution it was concluded that Content Diffusion is viable and credible between the Content Repository and ALERT® clinical applications, through a process of import and extraction with pre-defined rules.

Acknowledgements

I want to thank everyone that in some way helped me and supported me through the realization of this work. ALERT: for making available all resources necessary to this project's development; Professor Gabriel David for his guidance; Tiago Silva, António Pinto and Carlos Silva for the support and guidance inside the company; my Parents and Sister for the patience, love and support during all this process.

Igor Amado

Table of Contents

List of Figures

List of Tables

Abbreviations

ALERT	ALERT Life Sciences Computing, S.A. (the company)
ALERT®	ALERT's Products
AOM	Adaptive Objective Modelling
API	Application Programming Interface
Ch.	Chapter
CIXE	Content Import and Extraction Engine
CSV	Comma Separated Values
DAO	Data Access Object
DBA	Database Administrator
DDL	Data Definition Language
DML	Data Manipulation Language
DTD	Document Type Definition
ETL	Extract, Transform and Load
IDE	Integrated Development Environment
Java SE	Java Platform Standard Edition
JDBC	Java Database Connector
ORM	Object Relational Mapping
PL/SQL	Procedural Language/Structured Information Standards
Sec.	Section
SQL	Structured Query Language
SVN	Subversion
UML	Unified Modelling Language
XML	Extensible Markup Language
XRM	XML Relational Mapping
XSD	Extensible Schema Definition
XSL	Extensible Stylesheet Language

Chapter 1

Introduction

1.1 Context

This report documents the Master in Informatics and Computing Engineering project called "Content Diffusion in ALERT's Clinical Applications" held in the 2nd half of school year 2008/09, from February to July, over a period of 20 weeks.

The project was conducted in the company ALERT Life Sciences Computing SA, leader in Portuguese market in the segment of the clinical software.

The ALERT® is a suite of software applications that is filled with various types of Contents such as types of analysis, diagnoses, exams, procedures.

In the past, this Content was managed through Excel files and different databases, distributed by various sources, and then loaded into the ALERT® applications using manual processes. This form of maintenance and processing of information was not automated and does not centralized the universe of Contents of ALERT®.

On the client, each of the Contents does not have a unique identifier that relates in a standardized way, so it is virtually impossible to determine relationships between Contents or make data analysis. Similarly, the Contents in the Client are not assigned with standard codes, which would identify them in International Medical Standards.

To solve these problems, it was decided to create the Content Repository, which centralizes all the Content information of ALERT®, assigns unique identifiers and classifies each one of them with important parameters, such as Market, Version, Product, Author and Creation Date. The flexibility of the Content Repository, however, allows as many

classifications as are deemed necessary.

The aim of this project named Standardization is to standardize the identifiers of ALERT's Contents by creating and using an unique identifier for each of the objects associated with Content that exist in ALERT® clinical software.

The Contents of the ALERT® universe will be coded, classified and recorded in a central repository of Content and associated with the international standard in use in the ALERT® application. This will allow relating in a direct and quick way Content from this Content Repository and the ones from different customers, even if a particular Content may have different identifiers in different installations. This kind of unique identifiers creation needs to be structured in a flexible way. For that, a conceptual network was used following by the example of the model implemented on numerous coding systems like SNOMED-CT[SMCT09], currently maintained and distributed by International Health Terminology Standards Development Organization, and Unified Medical Language System from National Library of Medicine.

Because the model implemented on the coding system SNOMED-CT is very flexible and generic it is necessary to implement efficient mechanisms to import Content information from where they are currently stored, such as other ALERT Databases and data files (Excel and Flat files) that have their own metadata, to this generic model that will be the basis of the Content Repository Model. At the same time it will be inevitable the need of extraction of Contents from this Content Repository to ALERT® clinical products and to documents as listings.

This report will focus on the implementation of efficient mechanisms for Content diffusion – Import and Extraction – in ALERT's Clinical Products and will be developed as a module of this Content Repository.

1.2 Institution

ALERT, based in Vila Nova de Gaia, Portugal, is the parent company of a group of companies called the ALERT Group of Companies. This group is distributed to other parts of the world, including USA, Spain, Netherlands, Singapore, Brazil and the United Kingdom. Its mission:

> "Improving health and prolong life, achieve profitability for the benefit of society and inspire others to excellence, by our example."

Figure 1: ALERT's Company Logo

ALERT Group of Companies is dedicated to the development, distribution and implementation of software for health ALERT®, designed to create paperless medical environments [ALE09a].

All development is done in Portugal, with subsidiaries responsible for distribution, marketing and installation of products in their markets.

The ALERT® is currently distributed in 31 countries and is implemented in over 600 hospitals and nearly 8,000 health centers. It is available in 6 languages - English, Spanish, Italian, Dutch, European and Brazilian Portuguese and French. It is planned to release the ALERT® in German, Slovak, Croatian, Chinese, Russian, Japanese and Arabic in the short term.

As some distinctive features, the ALERT® presents a touch screen interface type, use of different profiles for different health professionals, establishment of work flows between them and biometric identification.

Figure 2: ALERT® software in use at a healthcare unit

ALERT® RHIO defines the complete solution when it comes to healthcare for a country or region which, by its turn is subdivided into solutions for hospitals (ALERT® PAPER FREE HOSPITAL), primary care (ALERT® PRIMARY CARE) and private clinics (ALERT® PRIVATE PRACTICE).

When it comes to hospitals, in particular, there are specific products for each environment: triage (ALERT® TRIAGE), emergency service (ALERT® EDIS), external consult (ALERT® OUTPATIENT), internment (ALERT® INPATIENT), and operating block (ALERT® ORIS). There are also non-clinical solutions (ALERT® CRM e ERP) and for data warehousing (ALERT® ADW).

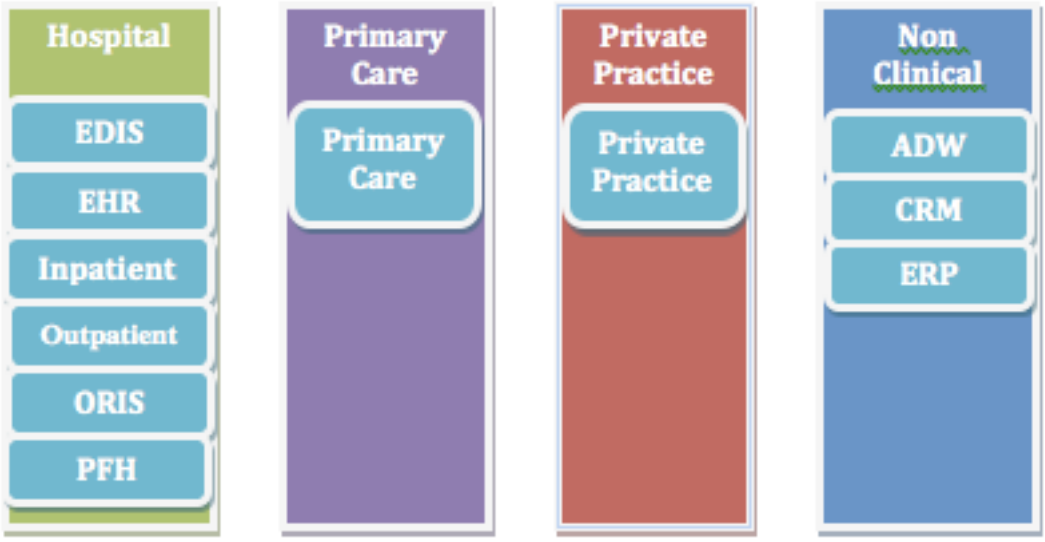


Figure 3: ALERT® Products

1.3 Project

1.3.1 Motivation

The Content Repository to be implemented will host not only various types of clinical Content, such as types of analysis, exams or diagnoses, and also non-clinical Content such as, for example, pricing of procedures or diagnoses. After its completion, this system will centralize the Contents of ALERT® clinical applications, allowing the extraction of Content for later importation into production environments.

This repository will allow creating a network of Contents, where each and every one of them is represented as concepts, and relations between them.

ALERT is a company with global presence, whose products are in use and implemented in more than 8,000 institutions throughout the world. Thus, the creation of this system, capable

of providing, at any moment, Content to any installation of the suite of ALERT® products will provide a clear added value in updating and maintaining Content in a list of customers increasingly wide.

Because of this kind of centralized Content Repository there comes a need for functionalities to act on this Repository that make the Contents diffusion in ALERT® possible. These functionalities include import Content to the Content Repository from file documents (excel and flat files) or other databases, not necessarily with the same metadata as the Content Repository, and the extraction of Contents for the ALERT® or for clients as listings, automatically. Essentially it shows the need of mapping Content information between different data models, taking into account their metadata information, in order to allow importing Contents to Content Repository and extracting Contents from the Content Repository.

1.3.2 Objectives

This project will initially involve the creation of a Content Repository, where the creation and updating of Content by multi-disciplinary teams, currently responsible for managing various types of Content, will be performed.

Next, it will be established mechanisms for importing Contents from existing databases and documents like excel files, and for extracting and packaging of Content associated with clinical versions of ALERT® products or for creating documents with listing of Contents to present to customers allowing them to choose what Contents do they need in their installation. These packages will then be versioned, and tested in Data Quality Control environment for updating the Content in production environments, with minimal human intervention.

Citing the official proposal [FEU09a], the developed software should primarily meet the following requirements:

- The system should be able to export and package Content for all versions of ALERT® clinical products, both for development and Data Quality Control environments, and also for production environments. Therefore it will have to support multiple types of data, by establishing procedures for extraction and processing of Content from the Content Repository;
- The system needs to allow creating installation packages able to update the databases of clinical ALERT® products, efficiently and optimized to reduce the impact of updating systems in production;
- The system needs to allow creating total packages of Content, partial updates and packages that support the upgrade of a certain type of Content, in response to request of clients in contact with the Support department.

- The installation packages need also to allow the cleaning of outdated and unused Contents and not used in the environments where they are executed;
- The system must keep a History and Log with all the information about the import/extraction process that will allow identifying what is causing problems that may occur during the process.

The role of this system is vital to the maintenance of updated Content in the ALERT® clients, avoiding time consuming human intervention tasks subject to error, and decreasing the time between the delivery of Contents by the responsible teams and the use of this Content in production environments.

1.4 Report Overview

Besides the Introduction, where context, motivation and objectives of this project are presented, sections for this report include in a second chapter State of Art for repositories and problems that may advent of not implementing them where it should, leading to elevated data dispersion. Also parts of this chapter are, State of art for data mapping and transformation, code generation for content diffusion and technological review.

Chapter three approaches subjects like problem description, past situation and the new Content Repository approach that requires a module for content diffusion.

Chapter 4 includes requirements analysis and proposed solution. Also in this chapter is an overview of the proposed architecture and system modelling.

In chapter 5 talks about is prototype development. This includes subjects like implementation decisions, implemented architecture, user interface layer, business logic layer, database layer and a test case for the prototype is made.

Finally, chapter 6 is an overview of the conclusions for the project and future work.

Chapter 2

Repositories and Data Diffusion

2.1 State of Art

In this chapter is described the State of Art of repositories and what can be used to implement a model generic enough in order to allow it to represent anything that might need to be represented in a repository.

For any repository there is a need to implement efficient mechanisms that will allow diffusion of its information to other stakeholder elements. A mix between Data Mapping and Code Generation, which is also covered in this section, can accomplish this.

2.1.1 Repository

An information repository is an easy way to deploy a secondary tier of data storage that can comprise multiple, networked data storage technologies running on diverse operating systems, where data that no longer needs to be in primary storage is protected, classified according to captured metadata, processed, de-duplicated, and then purged, automatically, based on data service level objectives and requirements. In information repositories, data storage resources are virtualized as composite storage sets and operate as a federated environment.

There are different understandings and definitions of repositories of information or digital repositories. The question most relevant to this diversity is the variety of contexts, communities, goals and practices relating to the creation and operation of repositories. Systems world-

wide, covering all subjects, allowing anyone to add or edit information, the institutional or systems for issues, only to authorized users, with the approval procedures and quality control.

Following the definition from “Digital Repositories JISC Briefing Paper” [DJP05] Information Repository are where digital Content, resources, are stored and can be searched and retrieved for later use. A repository supports mechanisms to import, export, identification, storage and retrieval of digital resources. However, even this definition is general and can be applied to different systems.

Information repositories were developed to mitigate problems arising from data dispersion and eliminate the need for separately deployed data storage solutions because of the concurrent deployment of diverse storage technologies running diverse operating systems. They feature centralized management for all deployed data storage resources. They are self-contained, support heterogeneous storage resources, support resource management to add, maintain, recycle, and terminate media, track of off-line media, and operate autonomously.

Data dispersion refers to the unprecedented amount of data, structured and unstructured, that business and government continue to generate at an unprecedented rate and the usability problems that result from attempting to store and manage that data. While originally pertaining to problems associated with paper documentation, data dispersion has become a major problem in primary and secondary data storage on computers.

2.1.1.1 Problems caused by data dispersion

The problem of data dispersion is affecting all areas of commerce as the result of the availability of relatively inexpensive data storage devices. This has made it very easy to dump data into secondary storage immediately after its window of usability has passed. This masks problems that could gravely affect the profitability of businesses and the efficient functioning of health services, police and security forces, local and national governments, and many other types of organization. Data diffusion is problematic for several reasons:

- Difficulty when trying to find and retrieve information.
- Data loss and legal liability when data is disorganized, not properly replicated, or cannot be found in a timely manner.
- Increased manpower requirements to manage increasingly chaotic data storage resources.
- Slower networks and application performance due to excess traffic as users search and search again for the material they need.
- High cost in terms of the energy resources required operating storage hardware.

Because information repositories are intended to reduce IT staff workload, they are designed to be easy to deploy and offer configuration flexibility, virtually limitless extensibility, redundancy, and reliable failover.

2.1.1.2 Clinical Datasets Approach – Conceptual Network

Clinicians and organizations use different clinical terms that mean the same thing. For example, the terms heart attack, myocardial infarction, and MI may mean the same thing to a cardiologist, but, to a computer, they are all different. There is a need to exchange clinical information consistently between different health care providers, care settings, researchers and others (semantic interoperability), and because medical information is recorded differently from place to place (on paper or electronically), a comprehensive, unified medical terminology system is needed as part of the information infrastructure.

Clinical Dataset often use conceptual networks as their inspiration for implementing their data model. A conceptual network is a network that represents semantic relations between the concepts. This is often used as a form of knowledge representation. Visually it could be represented as a directed or undirected graph consisting of vertices, which represent concepts, and edges. But in Figure 4 you can see a draft of how you could represent this conceptual network as a data model.

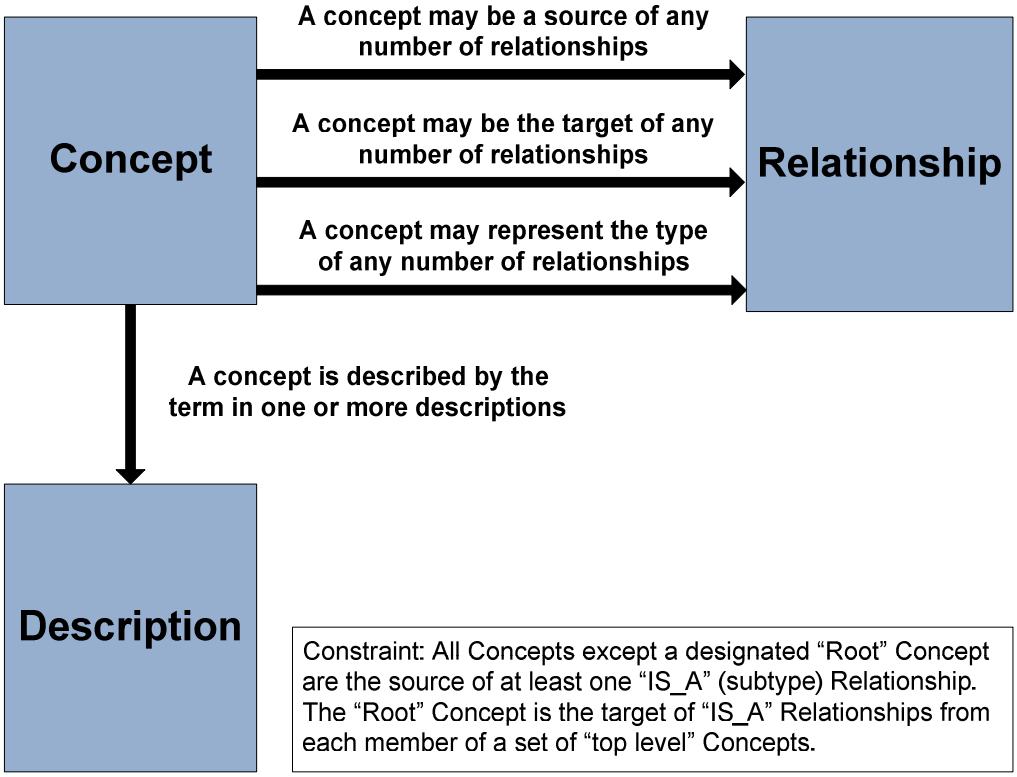


Figure 4: Conceptual Network Model [SCTM08]

- **Concepts** - The anchors for meaning
    - Basic unit of meaning designated by a unique numeric code, unique name (Fully Specified Name), and descriptions, including a preferred term and one or more synonyms.
- **Descriptions** - Terms or names (synonyms) assigned to a concept
    - Terms (strings of readable characters) used to express the meanings of the concepts in human language
- **Relationships** - Link concepts either within a hierarchy or across hierarchies
    - Concept-to-concept links used to express information in computer-processable language
        - First purpose: formal logical meanings
        - Other purposes: tracking retired concepts, representing “facts” that may vary, and supporting coding by suggesting valid qualifiers
- **Subsets** - Group of concepts, descriptions and/or relationships chosen to be relevant for use in a given circumstance or scenario

2.1.1.2.1 UMLS

Designed initially by Donald Lindberg, M.D., Director of the US National Library of Medicine [NLM08] in 1986, the Unified Medical Language System [UMLS08] is a controlled compendium of many vocabularies, which also provides a mapping structure between them. The purpose of NLM's Unified Medical Language System is to facilitate the development of computer systems that behave as if they "understand" the meaning of the language of biomedicine and health. To that end, NLM produces and distributes the UMLS Knowledge Sources (databases) and associated software tools (programs) for use by system developers in building or enhancing electronic information systems that create, process, retrieve, integrate, and/or aggregate biomedical and health data and information, as well as in informatics research. By design, the UMLS Knowledge Sources are multi-purpose. They are not optimized for particular applications, but can be applied in systems that perform a range of functions involving one or more types of information, e.g., patient records, scientific literature, guidelines, and public health data.

The UMLS is composed of three main knowledge components: Metathesaurus, Semantic Network and SPECIALIST Lexicon.

The relationship between the various items below provides a logical understanding of the structure and purpose of these three components:

- ***Metathesaurus***, the core database of the UMLS, a collection of concepts and terms from the various controlled vocabularies, and their relationships;

- ***Semantic Network***, a set of categories of relationships that are being used to classify and relate the entries in the Metathesaurus;
- ***SPECIALIST Lexicon***, a database of lexicographic information for use in natural language processing;

For the representation of a repository of information only the *Metathesaurus* component would be of interest.

The Metathesaurus forms the base of the UMLS and it comprises over 1 million biomedical concepts and 5 million concept names, all of which are from over 100 controlled vocabularies and classification systems used in patient records, bibliographic, administrative health data and full text databases. The Metathesaurus is organized by concept or meaning. In essence, its purpose is to link alternative names and views of the same concept together and to identify useful relationships between different concepts. It provides a basis of context and inter-context relationships between these various coding systems and vocabularies to provide a common basis of information exchange between the variety of clinical databases and systems. In the Metathesaurus, all the source vocabularies are available in a single, fullyspecified database format.

This dataset follows the model referred in 2.1.1.2.

2.1.1.2.2 SNOMED-CT

SNOMED (Systematized Nomenclature of Medicine) [SMCT09] is a systematically organized computer processable collection of medical terminology covering most areas of clinical information. It allows a consistent way to index, store, retrieve, and aggregate clinical data across specialties and sites of care. It also helps organizing the Content of medical records, reducing the variability in the way data is captured, encoded and used for clinical care of patients and research. It is owned and maintained by the College of American Pathologists [CAP09].

SNOMED began as a terminology system for pathology in 1965 and over the next 40 years evolved into SNOMED Clinical Terms (SNOMED CT). SNOMED CT resulted from the merger of SNOMED Reference Terminology (SNOMED RT) developed by the College of American Pathologists (CAP) and Clinical Terms Version 3 (CTV3) developed by the National Health Service (NHS) of the United Kingdom. It is designed for use in software applications like the electronic patient record, decision support systems, and to support the electronic communication of information between different clinical applications. Its designers' goal is that SNOMED CT should become the accepted international terminological resource for healthcare, supporting multilingual terminological renderings of common concepts.

The SNOMED CT core structure includes concepts, descriptions (terms) and the relationships between them with the objective of precisely representing clinical information across the scope of health care.

SNOMED CT is one of a suite of designated data standards for use in U.S. Federal Government systems for the electronic exchange of clinical health information.

Sample Computer Applications Using SNOMED CT

- Electronic Medical Records
- Computerized Provider Order Entry Such As E-Prescribing Or Laboratory Order Entry
- Remote Intensive Care Unit Monitoring
- Laboratory Reporting
- Emergency Room Charting
- Cancer Reporting
- Genetic Databases

The benefit of recording information in a standard terminology such as SNOMED CT is linked to the benefits of the electronic care record and the benefits of recording clinical information in a structured form

- It provides a consistent terminology across all care domains
- It allows precise recording of clinical information
- It has an inherent structure
- It is a developing international standard

SNOMED-CT tends to be used worldwide in various applications and is considered the most comprehensive and evolving.

This dataset follows the model referred in 2.1.1.2.

2.1.2 Data mapping and transformation

Technologies for overcoming heterogeneities between autonomous data sources are key in the emerging. At the heart of structural heterogeneity is the data-mapping problem. The data-mapping problem is to discover effective mappings between structured data sources. These mappings are the basic “glue” for facilitating large-scale ad-hoc information sharing between autonomous peers in a dynamic environment. Automating their discovery is one of the fundamental unsolved challenges for data interoperability.

Overcoming structural heterogeneity is a long-standing problem in database research.

The approaches to data mapping are numerous. One contribution is the one of Indiana University [WCFE04]. The main contribution of its research is a simple declarative data map-

ping calculus (DMC) for reasoning about the data-mapping problem. This calculus is a minimal extension of the standard relational model for cleanly expressing structural transformations for data mapping.

The data-mapping problem is illustrated in Figure 5. Consider the three databases containing patient exam information.

Database E1

Exams		
Name	**Exam**	**Result**
John	ExamA	45
John	ExamB	24
Sophie	ExamA	57
Sophie	ExamB	43

Database E2

Exams		
Patient	**ExamA**	**ExamB**
John	45	24
Sophie	57	43

Database E3

ExamA	
Name	**Result**
John	45
Sophie	57

ExamB	
Name	**Result**
John	24
Sophie	43

Figure 5: Mappings example between Exams representations

By this example you can see that each database contains the same information. As shown, there are many natural ways to organize even the simplest datasets. For example, E1 and E2 maintain the information in a single relation, while E3 contains a relation for each result. To move between these representations of exam data, schema matching and both data-data and data-metadata transformations must be performed. For example, mapping data from E1 to E2 involves promoting the values in column Exam in E1 to column names in E2 and “matching” the Name and Patient attributes. To move information from E3 to E1, relation names must be demoted to data values.

Other related data mapping solution are the works of Bilke and Naumann and Kang and Naughton [SMUD05] on schema matching, and the Clio project on schema mapping. To my knowledge, these works have not considered the full space of data-metadata transformations, with only the Clio project considering any aspects of such mappings. Their work complements and extends these works with a new perspective on the data mapping problem and a novel solution to this problem for the complete relational transformation space. This DMC approach solution can be considered as a useful addition to a multi-strategy data mapping approach. Relation-

al languages for database interoperability motivate work on DMC. The DMC is a descendant of the Uniform Calculus developed by Jain et al. [UDMR95], specialized to investigate relational data mapping. This calculus is a novel development of Jain's language that clearly captures a very minimal extension of the standard relational model for structural transformations for data mapping. DMC complements and extends this research with a *logical* characterization of the full space of relational data mapping transformations.

2.1.3 Code Generation for Content diffusion

For the extraction process of some repositories, which is the case of Content Repository, there is a need of some sort of code generation when extracting to some database taking into account its metadata information.

Code generation is the technique of using or writing programs that write source code. Code generators are tools built to serve engineers in the creation of applications. Just as woodworkers use customized tools called jigs to allow them to build furniture more quickly and accurately, code generators allow engineers to concentrate on building the application while the generator handles the grunt work tasks.

Taking a look to the code generation process

Meta data is used to describe the characteristics about the program to be generated. The metadata can be represented in various formats such as relational data, properties files, XML, or other formats. A working program serves as the model that the generated programs should emulate. A code generation program uses the metadata and code fragments from the model program to generate new programs.

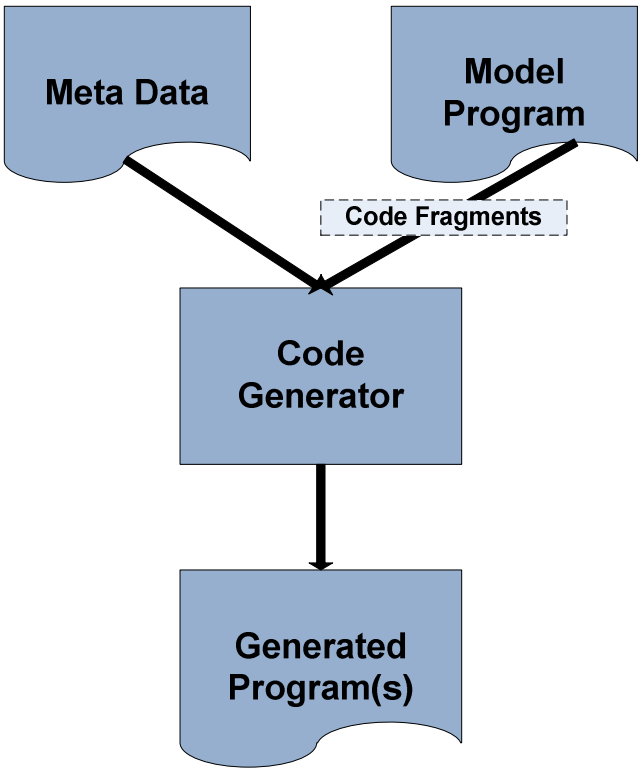


Figure 6: Generic Code Generation process [XSLCG05]

Generated programs share the characteristics of the model program. If the model program is syntactically correct and well tested, the generated program will be syntactically correct and well tested. If a bug is found in the model program, it can be fixed in the code generator and all of the generated programs can be regenerated and corrected. In short, code generation can save a lot of coding and testing.

Despite the benefits of code generation, it can promote a cut and paste type of mentality. It is wise to consider generalizing behavior and using inheritance or composition as well as code generation in an overall strategy.

Another important consideration in code generation is determining whether customization of generated programs is allowed. When customization is allowed, regeneration may not be possible to pick up new features.

For years, every good SQL programmer and DBA has used code generation techniques to generate SQL or other programs from SQL. The reason that this technique worked so well was that relational database management systems have excellent metadata describing all of the database objects. The SQL language made it trivial to query this metadata, combine it with code fragments, and output generated programs.

Other languages, such as Perl, are also widely used to generate build scripts, code, and the like. However, when using scripting languages, there may not be a standard data format and

protocol that the language is optimized to use as in the case of SQL. On the positive side of the equation, it is much easier to write modular code in Perl than SQL scripts.

XML [XML09], as with a relational database, excels at representing structured data. XML happens to be a more portable and lightweight format. Just as SQL was able to easily query the metadata from relational tables, XSL [XSL09] and XPath [XPH09] can easily query data from XML documents, combine it with code templates, and output generated programs.

Whereas SQL code generation scripts were often ugly monsters, XSL was designed to have modular templates of code that could be easily applied to nodes in the metadata document. It provides the best of both worlds. Like SQL, it is optimized for a given data format and protocol-programmatic XML parsing isn't required. And like scripting languages such as Perl, code templates can be easily modularized.

2.2 Technological Review

In this section is made a review of technologies that use this concept of mapping and transformations of metadata and data between different data models.

Also a review of technologies in use by ALERT is made.

2.2.1 Mapping and transformations based Tools

The concept of data mapping is used in several types of tools being Object Relational Mapping tools and Data Warehousing tools some of the most well known.

Data mappings can be done in a variety of ways using procedural code, creating XSLT transforms or by using graphical mapping tools that automatically generate executable transformation programs. These are graphical tools that allow a user to "draw" lines from fields in one set of data to fields in another. Some graphical data mapping tools allow users to "Auto-connect" a source and a destination. This feature is dependent on the source and destination data element name being the same. Transformation programs are automatically created in SQL, XSLT, Java programming language or C++. These kinds of graphical tools are found in most ETL Tools (Extract, Transform, Load Tools) as the primary means of entering data maps to support data movement.

2.2.1.1 Object Relational Mapping

Whether we are developing a small or a big application, we always have to deal with data. It's even a critical part of an application. Problem is this is tedious, repetitive work, which consumes a lot of the time we would prefer to spend on other parts of the application, without forgetting

that the less interesting the work is, the higher the risks of errors.

To solve these problems, multiple solutions exist. Their goal is to simplify the creation of data access layers, automate data access, or generate data access code.

The principle of object-relational mapping is to delegate to tools the management of persistence, and to work at code-level with objects representing a domain model, and not with data structures in the same format as the relational database. Object-relational mapping tools establish a bidirectional link with data in a relational database and objects in code, based on a configuration and by executing SQL queries (dynamic most of the time) on the database.

Other solutions exist, such as those based on code generation. They all have their pros and cons, just as it's the case for mapping tools themselves of course. The perfect tool for all situations does not exist.

In terms of tools, the offer is huge. This is true for .NET as well as for Java, even if the offer for Java is more advanced for historical reasons. There is anyway an impressive quantity of tools for both sides.

The ORM approach has the following advantages:

- Java domain model uses natural Java programming style (uses Collections and Iterators to navigate relationships)
- ORM runtime can provide capabilities such as Caching, Auditing that would have to be hand-coded using the DAO approach

The ORM approach has the following disadvantages:

- Data and behavior are not separated
- Each ORM technology/product has a different set of APIs and porting code between them is not easy

The real values in using an ORM tool is to save time, simplify development (i.e. the ORM tool handles the complexity for the developer), increase performance or scalability, and minimize architectural challenges related to inability of the ORM tool or developer's experience.

2.2.1.2 Data Warehousing ETL Mapping and Transformations

Extract, transform, and load (ETL) in database usage and especially in data warehousing involves:

- Extracting data from different sources
- Transforming it to fit operational needs (which can include quality levels)
- Loading it into the end target (database or data warehouse)

The advantages of efficient and consistent databases make ETL very important as the way data actually gets loaded.

This article discusses ETL in the context of a data warehouse, whereas the term ETL can in fact refer to a process that loads any database.

ETL can also function to integrate contemporary data with legacy systems.

Usually ETL implementations store an audit trail on positive and negative process runs. In almost all designs, this audit trail does not give the level of granularity, which would allow reproducing the ETL's result in the absence of the raw data.

Extract

The first part of an ETL process involves extracting the data from the source systems. Most data warehousing projects consolidate data from different source systems. Each separate system may also use a different data organization / format. Common data source formats are relational databases and flat files, but may include non-relational database structures such as Information Management System (IMS) or other data structures such as Virtual Storage Access Method (VSAM) or Indexed Sequential Access Method (ISAM), or even fetching from outside sources such as web spidering or screen-scraping. Extraction converts the data into a format for transformation processing.

An intrinsic part of the extraction involves the parsing of extracted data, resulting in a check if the data meets an expected pattern or structure. If not, the data may be rejected entirely.

Transform

The transform stage applies to a series of rules or functions to the extracted data from the source to derive the data for loading into the end target. Some data sources will require very little or even no manipulation of data. In other cases, one or more of the following transformations types to meet the business and technical needs of the end target may be required:

- Selecting only certain columns to load (or selecting null columns not to load)
- Translating coded values (e.g., if the source system stores 1 for male and 2 for female, but the warehouse stores M for male and F for female), this calls for automated data cleansing; no manual cleansing occurs during ETL
- Encoding free-form values (e.g., mapping “Male” to “1” and “Mr” to M)
- Deriving a new calculated value (e.g., sale_amount = qty * unit_price)
- Filtering
- Sorting
- Joining data from multiple sources (e.g., lookup, merge)

- Aggregation (for example, rollup - summarizing multiple rows of data - total sales for each store, and for each region, etc.)
- Generating surrogate-key values
- Transposing or pivoting (turning multiple columns into multiple rows or vice versa)
- Splitting a column into multiple columns (e.g., putting a comma-separated list specified as a string in one column as individual values in different columns)
- Applying any form of simple or complex data validation. If validation fails, it may result in a full, partial or no rejection of the data, and thus none, some or all the data is handed over to the next step, depending on the rule design and exception handling. Many of the above transformations may result in exceptions, for example, when a code translation parses an unknown code in the extracted data.

Load

The load phase loads the data into the end target, usually the data warehouse (DW). Depending on the requirements of the organization, this process varies widely. Some data warehouses may overwrite existing information with cumulative, updated data every week, while other DW (or even other parts of the same DW) may add new data in a historized form, for example, hourly. The timing and scope to replace or append are strategic design choices dependent on the time available and the business needs. More complex systems can maintain a history and audit trail of all changes to the data loaded in the DW.

As the load phase interacts with a database, the constraints defined in the database schema — as well as in triggers activated upon data load — apply (for example, uniqueness, referential integrity, mandatory fields), which also contribute to the overall data quality performance of the ETL process.

2.2.1.3 XML and XSLT

XML is a markup language for documents containing structured information.
Structured information contains both content (words, pictures, etc.) and some indication of what role that content plays (for example, content in a section heading has a different meaning from content in a footnote, which means something different than content in a figure caption or content in a database table, etc.). Almost all documents have some structure. A markup language is a mechanism to identify structures in a document. The XML specification defines a standard way to add markup to documents.

Short for E**x**tensible **S**tyle **L**anguage **T**ransformation, the language used in XSL style sheets to transform XML documents into other XML documents. An XSL processor reads the XML document and follows the instructions in the XSL style sheet, then it outputs a new XML

document or XML-document fragment. This is extremely useful in e-commerce, where the same data need to be converted into different representations of XML. Not all companies use the exact same programs, applications and computer systems.

XSL Transformation (XSLT) is a member of the XML family of languages. It describes how an XML structure is transformed into another XML structure.

You can define mappings using XSLT together with XPath. XPath is also a specification of the XML family. Using XPath you can address any node in an XML document. XSLT implements XPath expressions to select substructures of an XML document. Using templates in XSLT you can define the mapping rules for the selected substructures.

2.2.2 Development Tools

This section is meant to review tools used for accomplishing the daily tasks of this project in the areas of Integrated Development Environments, version control and Presentation technologies.

2.2.2.1 PL/SQL Developer

PL / SQL Developer [PSDEV09] is a tool developed by a Dutch company "All Round Automations". It is a dedicated IDE is a dedicated unit to develop code for a database Oracle. This IDE supports several features that improve productivity, such as:

- Editor, PL / SQL with syntax coloring, auto-complete with a code assistant, refactoring, code folding, beautifier (automatic code indentation), among others;
- Debugger for PL / SQL;
- Windows to analyze the results of SQL queries, command line (SQLPlus), diagrams of the data model;
- Creation and maintenance of projects;
- Tools to analyze the performance of code and SQL queries;
- Tools for creating and maintaining various types of objects that comprise the data model as tables, constraints, views, triggers, packages, functions, procedures, and others;
- Macros to automatically insert code for common macros and keyboard;
- Objects explorer: allows us to view all objects in the DB to which the user has access;
- Extensible architecture that allows installation of plug-ins.

This tool is adopted by ALERT as default database development tool.

2.2.2.2 Eclipse

Eclipse [ECLP09] is best known for its Integrated Development Environment (IDE), but it's a multi-purpose open source platform, which can be used for a variety of software development related tasks. It stands as a powerful user-friendly platform that increases productivity and efficiency noticeably. Eclipse relies on plug-ins to provide integrated utilities for most popular programming languages.

This IDE is adopted by ALERT as default development environment for Java SE and FLASH.

2.2.2.3 SVN

SVN [SVN09] (also known as Subversion) is a version control system. It's targeted at large projects, and especially useful when concurrent editing of files is needed. It uses client-server architecture: a server stores the most current version of the project and its history, and clients connect to the server in order to check out a complete copy of the project. Clients then work on their local copy and later check in their changes to the server (repository).

SVN features include:

- Support for concurrent work on the same project, by several developers;
- Automatic merge of changes, if no conflicting changes are made;
- Versioning of files, directories, renames, and file meta-data;
- Branching and tagging (e.g. create a branch for debugging while keeping the main branch for
- New features development);
- Parseable and human-readable output;
- Efficient delta compression (when a file is committed to the server, only the lines that have suffered changes from the previous version are transmitted and stored, saving on resources).

All the database scripts and interface source code for ALERT® are kept on a Subversion repository and managed through SVN versioning system. For interface development, access to the repository is commonly accessed through Eclipse's Subclipse plug-in. For the database there is a plug-in for PL/SQL Developer.

2.2.3 ALERT ® Technologies

In figure 7 are represented the main technologies used in all products developed by ALERT.

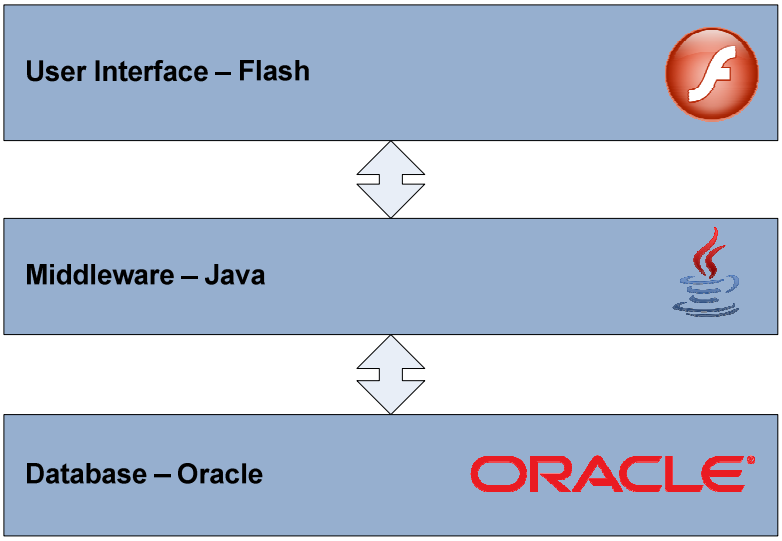


Figure 7: ALERT ® Main Technologies

In resume three layers compose the application. Flash is used for interface between the users and the application. Java is used as middleware and allows flash to communicate with the database layer. Oracle is used as database information.

2.2.3.1 Database – Oracle

Introduced in the late 1970s, Oracle [ORAC09] database is Oracle's flagship product and was the first database product to run on a variety of platforms. Nowadays it is one of the most popular databases of the world.

ALERT® has Oracle database, because it is flexible, efficient and has small costs of management. The version currently in use by ALERT® is Oracle10g, which was introduced on the market in 2005.

2.2.3.2 Middleware – JAVA

Java is an object-oriented programming language, which aims to:

1. Allow the same program to be executed on multiple operating systems.
2. Contain built-in support for using computer networks.
3. Execute code from remote sources securely.

4. Include the good aspects of other object-oriented languages, making it easy to use.

Perhaps the built-in support for computer networks and the ability to execute remote code securely were the characteristics that most contributed to choosing Java. ALERT® needed a fast, robust and secure way of interchanging data between multiple interfaces on the users' equipments and a central database in a server. Java makes this link possible through the use of web services.

Flash does not communicate directly with the database. Instead, there is an intermediate Java layer that manages the connection between interface and database. Although the Java component of the software is relatively small when compared to the other layers, and is essentially made of automatically generated classes, it represents a vital part of the system. It exposes the database functions, providing corresponding services, which can be accessed remotely. All information between the presentation and logic/data tiers is exchanged through these Java services, using Flash Remoting. This allows for interface development to be independent of database structure (and vice-versa), thus any structural changes in database are transparent to a Flash developer.

This layer is responsible for:

- **Database connection management**: Java manages every database connection and method invocation.
- **Session management**: logins, session timeouts and edit timeouts are managed within this layer.
- **Service logging and reports**: Java maintains log files to track service exceptions.

2.2.3.3 Presentation – Adobe Flash

Flash [FLS09] is an authoring tool widely used to create a diversity of interactive content and applications.

Utilities created with Flash can be as simplistic as a promotional banner, or as complex as 3D games, charting tools and complete websites. Although Flash is mostly used for web content (in great part due to its great flexibility, the reduced size of the files it produces, and because it's a cross-browser tool), it's also frequently adopted by developers to build the User Interface components of their software.

There are several reasons why Flash was chosen as ALERT®'s interface authoring tool: Flash is extremely flexible. Among other things, it works with vectorized components and does not rely on strict component layouts. This makes it possible to run a Flash application on displays with different sizes and resolutions, without compromising its look and feel.

Making ALERT® software remotely accessible to healthcare professionals has been on the company's plans since its foundation. When the time comes, this migration will be extremely simplified by having a Flash-based interface, since it was designed as a technology for the web. Apart from programmers, end user requirements to access ALERT® remotely will be virtually inexistent: a browser with Flash player plug-in is the only software required, and both are found on most computers or can be easily downloaded.

In addition to being a powerful design tool, Flash is bundled with its own object-oriented scripting language, ActionScript, enabling the creation of complex applications behind an appealing and flexible user interface.

Other technologies such as Oracle Forms, Windows forms, Java Applets or Ajax (HTML and Javascript) could be eventual alternatives for Flash as ALERT®'s interface. In spite of having their own advantages, whose discussion is out of the scope of this report, Adobe Flash was considered a more adequate solution for the reasons explained.

2.2.3.4 Adobe Flex

Adobe Flex [FLEX09] is a software development kit released by Adobe Systems for the development and deployment of cross-platform rich Internet applications based on the Adobe Flash platform. Flex applications can be written using Adobe Flex Builder or by using the freely available Flex compiler from Adobe.

In February 2008, Adobe released the Flex 3 SDK under the open source Mozilla Public License. Adobe Flash Player, the runtime on which Flex applications are viewed, and Adobe Flex Builder, the IDE built on the open source Eclipse platform and used to build Flex applications, remain proprietary.

Flex development process is composed by these steps:

1. Define an application interface using a set of pre-defined components (forms, buttons, and so on)
2. Arrange components into a user interface design
3. Use styles and themes to define the visual design
4. Add dynamic behavior (one part of the application interacting with another, for example)
5. Define and connect to data services as needed
6. Build the source code into an SWF file that runs in the Flash Player

This technology is starting to be widely adopted in the rich interface application area, and is overpowering Flash because of it's becoming a more developers friendly language, not

only because its IDE is Eclipse based but also because its development process accelerates the overall time spent on visual components development.

The user interface of all ALERT applications is evolving to the use of Adobe Flex as standard technology for its implementation.

Chapter 3

Problem Description

The ALERT® is a suite of software applications that is filled with various types of Contents such as types of analysis, diagnoses, exams, procedures, medication, or allergies.

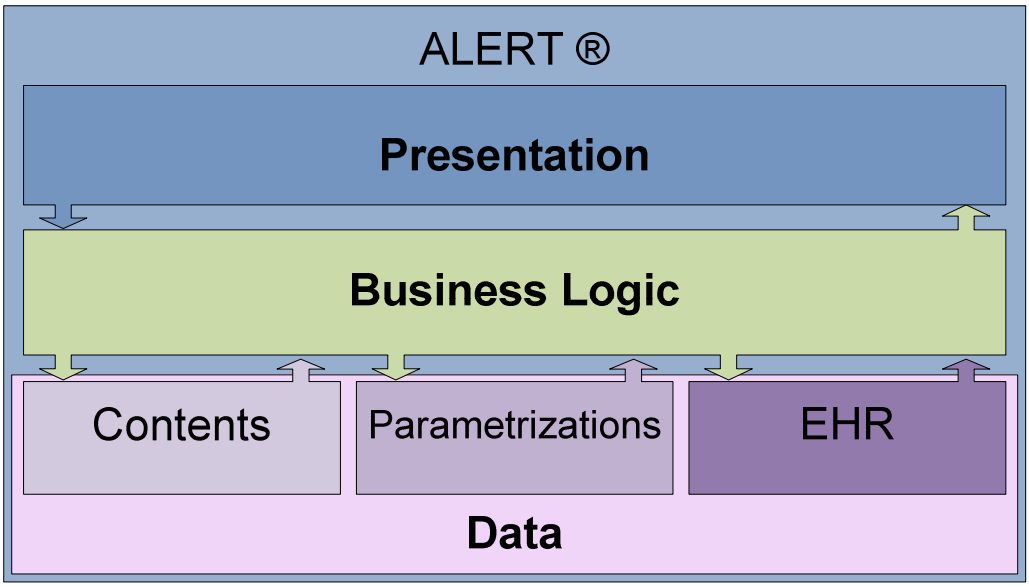


Figure 8: Overview of ALERT® Product Architecture

As you can see in Figure 8 ALERT® is composed by three main layers:

- **Presentation –** This is responsible of presenting the user with interfaces that allow him to navigate among all the applications Contents and execute actions that affect them.

- **Business Logic –** This layer is responsible by information exchange between the presentation and data layers. It executes the actions requested by the presentation layer that will in some way affect information in Data layer.
- **Data –** Here we have **Content** Tables that have the information about the Content itself, **Parameterizations** Tables with information where certain Contents are available, and **Transactional** Tables are responsible of relating Content information with Patient/Professional in order to build its Electronic Health Records

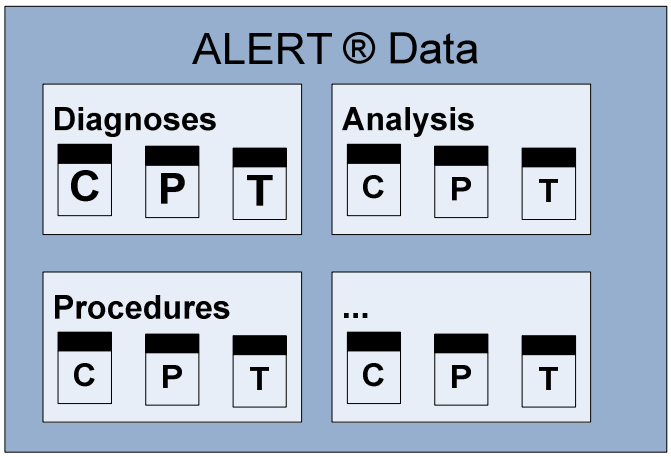


Figure 9: Overview of ALERT® Data components
(C-Contents Tables; P-Parameterizations Tables; T-Transactional Tables)

For this project's implementation the focus is in the Data layer. The components that will be saved to the Content Repository are the Contents component - this is where you have the information that represent a Content – and Parameterizations component – this is where you have the information that allows you to specify where your Contents will be available taking into account numerous possible classifications like Market by country, Language, Product, Version of the Product, and others.

3.1 Past Situation

In the past, this Content was managed through Excel files and different databases, distributed by various sources, and then loaded in the ALERT® using manual processes.

As you can see in Figure 10, all Content is dispersed in different formats and from different sources mainly databases and files documents.

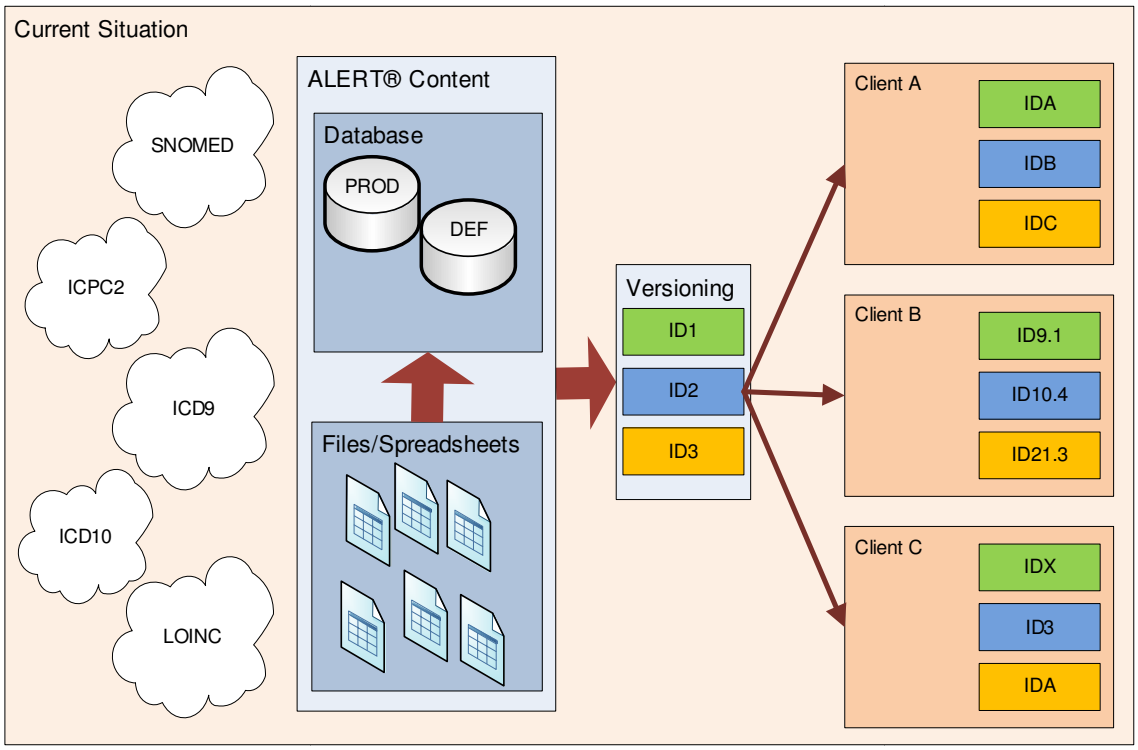


Figure 10: Overview of Past Situation for ALERT® product's Content dispersion

In ALERT there are several environments where they have Content information needed to be available to ALERT® applications. This refers not only to clinical Contents such as, for example, types of analysis, exams or diagnoses but also includes non clinical Contents, related to the coding and pricing of procedures, exams, analysis, among others, performed in hospital environments, health centers and private practice.

3.1.1 Past Content Workflow

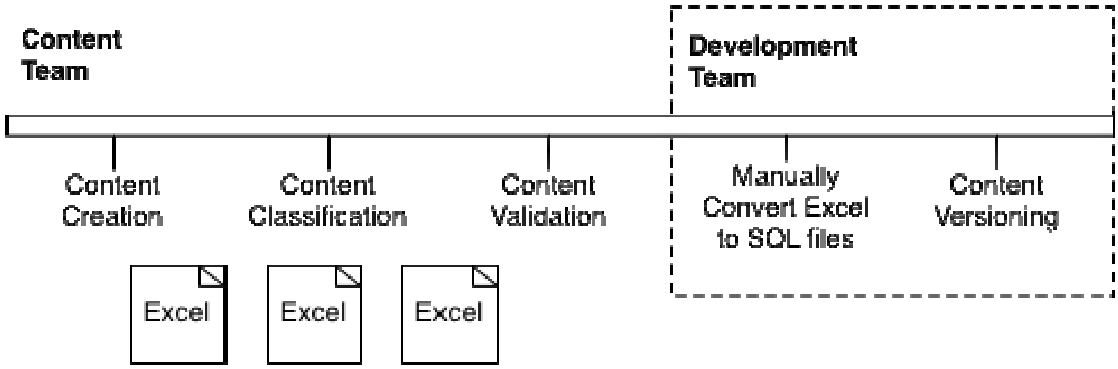


Figure 11: Past Workflow for Contents

In the past, Content workflow had two teams involved. Content team was responsible for

creating content, classifying them and validating its information. This was done by working in an excel document that would be transferred by the person responsible for each phase of the process. Then finally the development team would take the validated excel document and convert its Contents to SQL DML files to create them in a Content database. This document didn't always have the same format so the process couldn't be automated. In the end the script is versioned for Data Quality Control.

3.1.2 Identified Problems

This form of content validation, maintenance and processing is not automated and does not centralize the universe of Contents of ALERT®.

This dispersion, can lead to difficulty when trying to find and retrieve Contents needed to certain Client installations and possible Data loss and legal liability when data is disorganized, not properly replicated, or cannot be found in a timely manner. This kind of Content dispersion has increased manpower requirements to manage increasingly chaotic data storage resources such as, for example, multiple Databases with non-standardized schemas and Excel files.

On the client side, each Contents does not have an unique identifier that relates in a standardized way, so it is virtually impossible to determine relationships or make data analysis. Similarly, the Contents in the Client are not assigned with code standards, which would identify them in International Standards. This would allow interoperability with different Electronic Medical Records implemented in different locations, and possibly make data analysis in different markets.

Creating a new Content is not easy because of all this type of Content dispersion, it is important to implement a Content development environment where one easily could create new Content with validation and classification phases following a well-defined workflow.

3.2 Content Repository

In this section will be presented an overview of the solution, named Content Repository, a new Content Development Environment currently being developed to solve the problems and limitations referred in section 3.1.2.

3.2.1 General Process

To solve the problems identified in section 3.1.2, it was decided to create a Content Repository, which will centralize all Content information of ALERT® products, gives an Unique Identifier and classifies each and every one of the Contents with different parameters such as Market, Version, Product and Author.

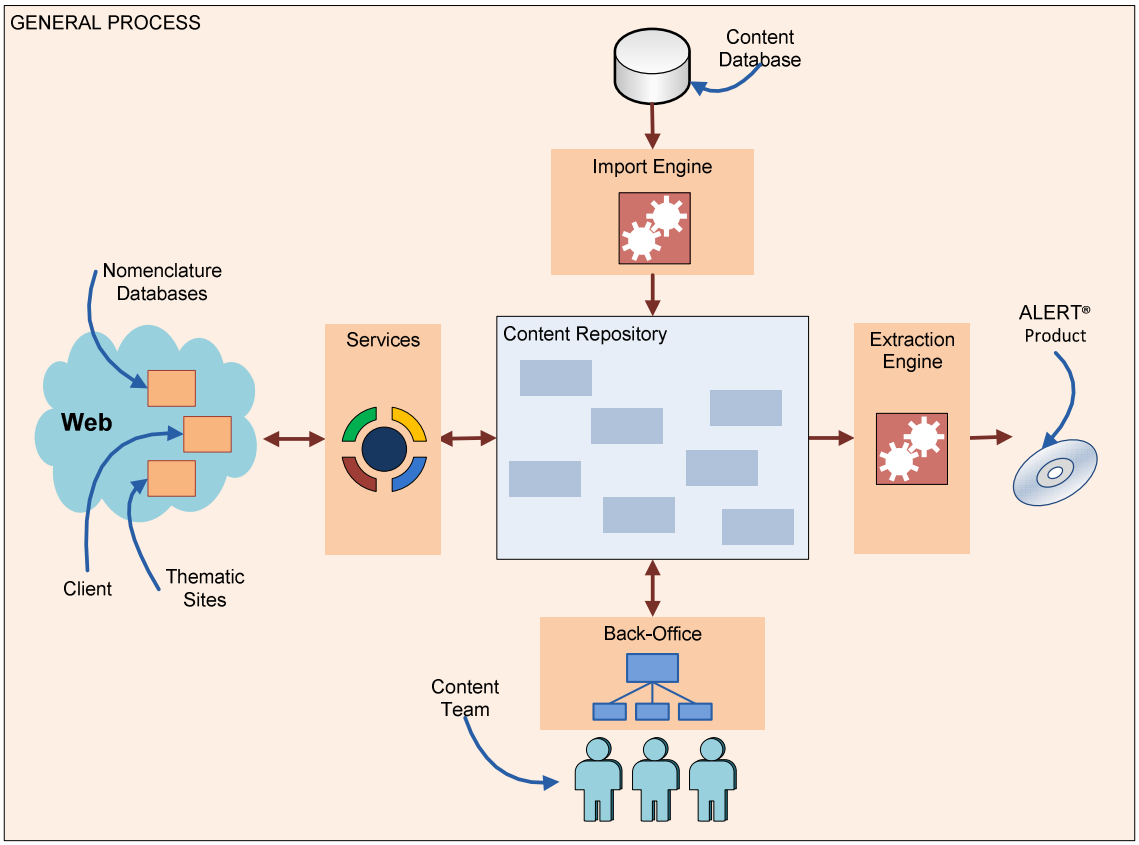


Figure 12: General Content Repository Process

By analyzing what was the best way of how to properly represent the Contents, it was concluded that the model in which SNOMED-CT is based, given its flexibility and power in dealing with different structures simultaneously, would be more appropriate. This model was not only adopted by SNOMED-CT but also by other entities that publish clinical data sets, for example the UMLS referred in the State of Art section of this document.

After the study of this model and its features, it was determined which ones would be taken to the Content Repository and which would not apply and / or that should be subject to revision. The study was based on official technical references of SNOMED, developed by IHTSDO.

All modules that were not represented in SNOMED-CT model, or whose representation is insufficient, were developed to maintain consistency with the other structural models.

3.2.2 Content Repository Workflow

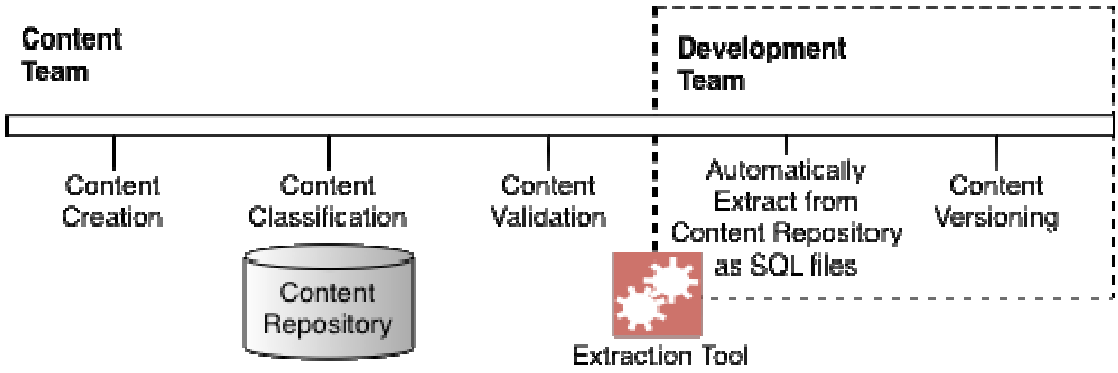


Figure 13: Content Repository Workflow for Contents

Now with the Content Repository, Content workflow has two teams involved. Content team is responsible for creating content, classifying them and validating its information with a workflow that is automatically managed by the Content Repository. In the end, Development team will use this extraction tool to automatically generate SQL DML files and version them to Data Quality Control.

3.2.3 Basic Structure

The Content Repository is composed of modules of information and features that relate and exchange information with each other. Some of the modules contain information (metadata) to monitor and manage Content allowing them to be manipulated by other modules.

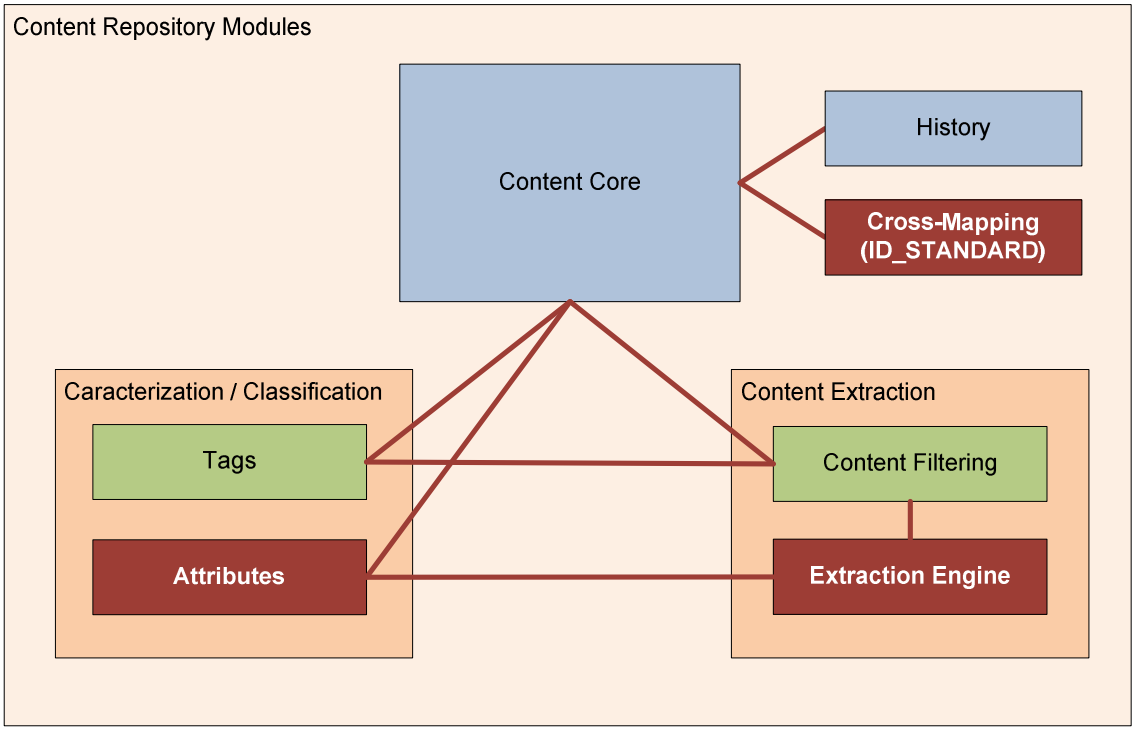


Figure 14: Basic Content Repository Structure

All Content is associated with an history where all the information about actions taken is recorded since the creation of each Content with information from the ground of the amendment, the state after the change, the author and the date of amendment. In addition to the treatment process of history, the Content Core has its own internal structures that record the changes, relating them directly to the changed Content.

The Contents referring to International Standards are recorded in their own structure, which includes the functionality of mapping, if possible, to each Content a STANDARD ID which indicates that Content as corresponding to each international pattern.

To give meaning and to deal with the Content, the Characterization and Classification module indicates the Tags (e.g. version or market) and attributes (eg, Minimum Age and Dosage) associated with each concept. This information is essential to control the extraction process (including filtering).

The Content Extraction Module to be developed is responsible for carrying out the filtering (based on tags) and extract the information, properly formatted, of each filtered Content (complete with its attributes).

3.2.4 Content Core

At the Core of the Content Repository is the basic structure of the Conceptual Network, which organizes all the existing Content.

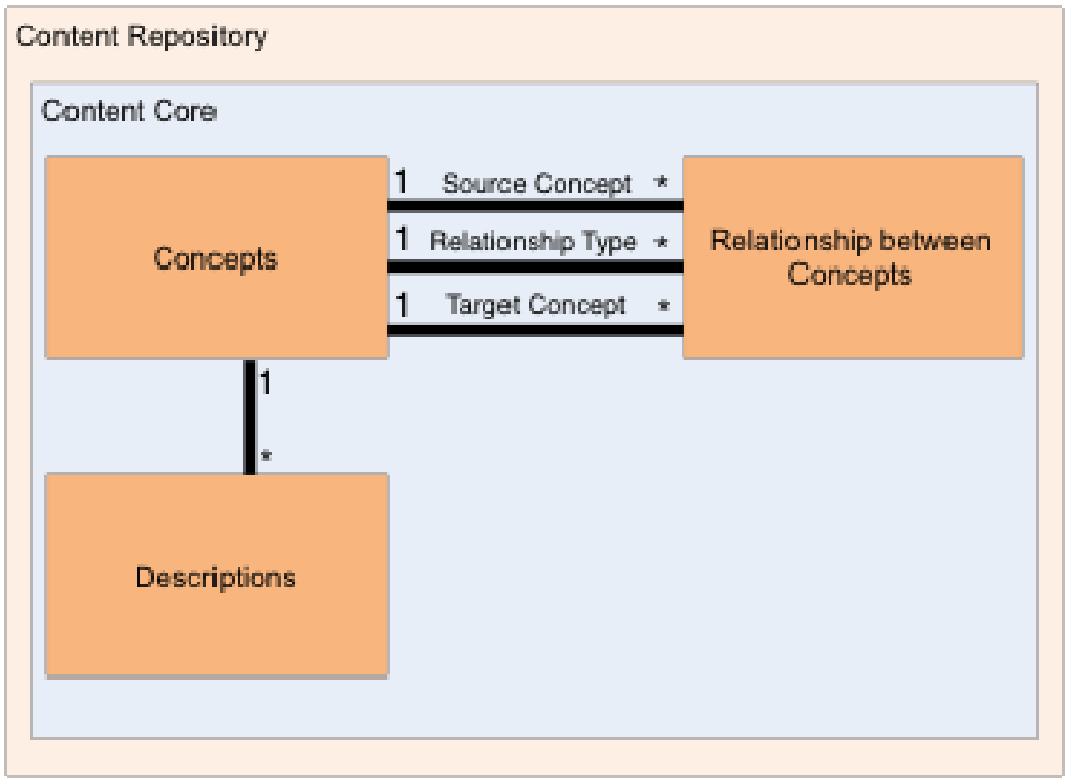


Figure 15: Content Core Representation

The Conceptual Network is composed of three basic elements: concepts, relations between concepts and Descriptions of Concepts. Relations are defined as special kinds of concepts and are set between two concepts. The wording of concepts can take different forms, depending on their use: a full description defines in an absolute and exclusive way each Content, a preferred term to define each Content in a more abbreviated way and there may be (not mandatory) several Synonyms for the same concept.

All elements that are necessary in the structure of the Conceptual Network should be defined as concepts and their definition connections must be defined with other concepts.

The Contents are particular cases of Concepts in the Conceptual Network. They are defined from classes of concepts specific to each Universe of Contents (analysis, diagnoses, samples, etc.). A Content (and, in general, each concept) has always a single Unique Identifier named ID_CONTENT, generated sequentially and never reused, not even if a Content is disabled. Note that, even if disabled, a Content continues to exist and be part of the Conceptual Network, which maintains its identifier, but its disabled was possibly defined because it prescribed or maybe replaced by an other Content, action that can be confirmed in history module.

3.2.5 Content Filtering

The filtering process is activated by user action, which needs to define which filters to apply on the Contents he wishes to extract for a Client installation. The set of indicated conditions by this user defines the general extraction filter that will feed the extraction mechanism developed for this repository.

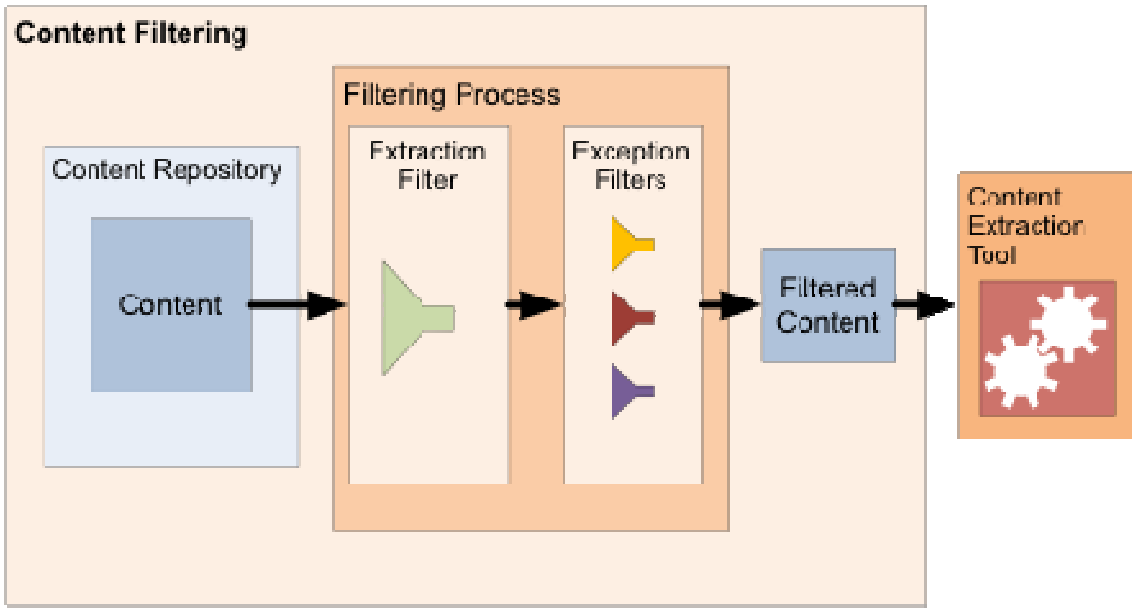


Figure 16: Overview of the filtering process

The extraction filter is a user defined set of values to be verified by tags that indicate which markets, products, or other factor version classifier implemented for the Content. Those tags that classify the Content are the rule of classification. However, if the Content has combinations of tags that do not apply, user shall register such combinations as Exceptions. Moreover, there may be combinations of tags not defined in the rule, but that should be considered in certain exceptional cases.

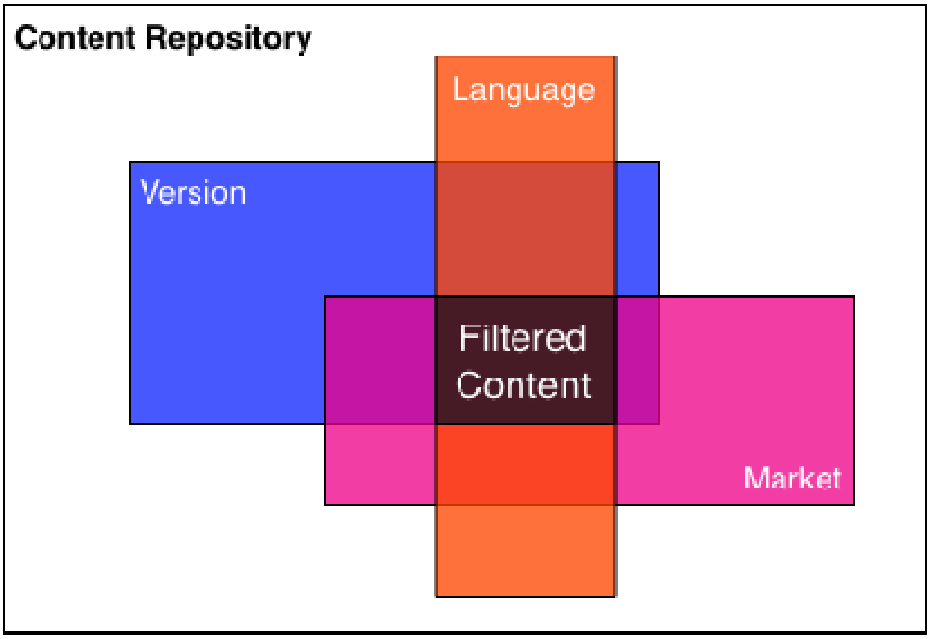


Figure 17: Filtered Content example

For example, a Content can be defined in the markets UK, U.S. and PT, in versions V2.4.3 and v2.5. These ratings, derived from all possible combinations between two tags of the market and Version (Table 1).

Table 1: Possible Combinations for Tags

	V2.4.3	**V2.5**
UK	x	x
US	x	x
PT	x	x

This rule defines the general classification of the Content depending on version and market, with six cases to consider (the number of combinations increases if we add other classifications, such as product, for example). However, it may happen that the Content is not available in the U.S. market in case of version v2.5. In addition, you can still occur that the Content is available in the U.S. market only to V2.4.3.

In conclusion, the practical applications of the Content can be seen in Table 2.

Table 2: Example of Tag Values for a Content

	V2.4.3	**V2.5**
UK	✓	✓
US	✓	✗
PT	✓	✓
BR	✓	✗

Thus, over the general Rule that allows classifying the majority of situations in a more direct and simple way, there are two situations of exception to consider: An exception to an exclusion of cases defined by rule and an exception to inclusion of a case not defined in the Rule.

3.3 Content Diffusion

In any type of repository there is a need to act functionalities that make Content diffusion possible. In the Content Repository case, these diffusion functionalities refer to the need of Content importation to the Content Repository and Content extraction from de Content Repository automatically not only from/to ALERT® database structure but also to documents for Content listings.

3.3.1 Identified Implications

In the case of Content Import, this applies mainly in the initial phase of loading from existing Content in ALERT main Content Database named DEF, although it is a process that can be implemented where necessary, to apply on other Content databases or files with Content that register interest to the repository.

The Content Import Engine would have the purpose of allowing the import of new or existing Content, to the Content Repository. These can be represented in different formats such as the various current Content databases or documents like Excel / Flat Files that have Content information.

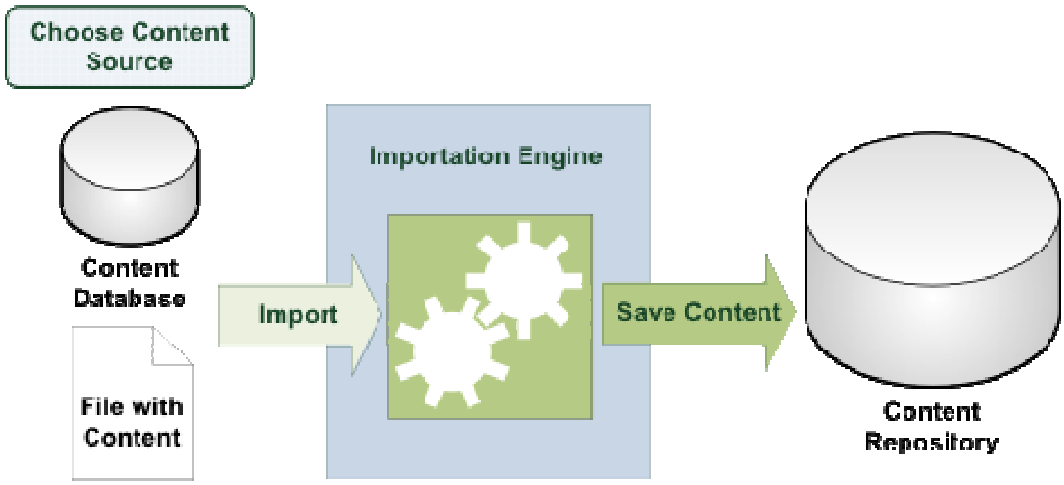


Figure 18: Overview of the importation process

In the case of Content Extraction, the process needs to take place whenever it is necessary to extract Content information from the Content Repository for installation in ALERT® product or for listing. This process needs to be performed in an automated way, based on requests made to an extraction tool.

The Content Extraction Engine works in the reverse direction of the import engine and its purpose is to allow the extraction of Content from the Content Repository for placing Content on the ALERT® product to install Content at the Client's installation or for creating documents such as Excel / Flat Files to allow creating listings of Contents. It is important in this kind of process that the Content information contained in each of the extractions is recorded for later reference, for example in case of upgrades or changes. The information on parameters used in each extraction should also be registered, so it is always possible to find out how the extraction took place in the event of an anomaly.

This extraction engine receives the list of Contents to extract from the filtering process. The filtering process takes into account the existence of rules, which should be tested in a first Phase, and the possible existence of exceptions, of exclusion or inclusion, which are tested in a second phase. Only after these two phases the selected Content will be added to the list of Contents.

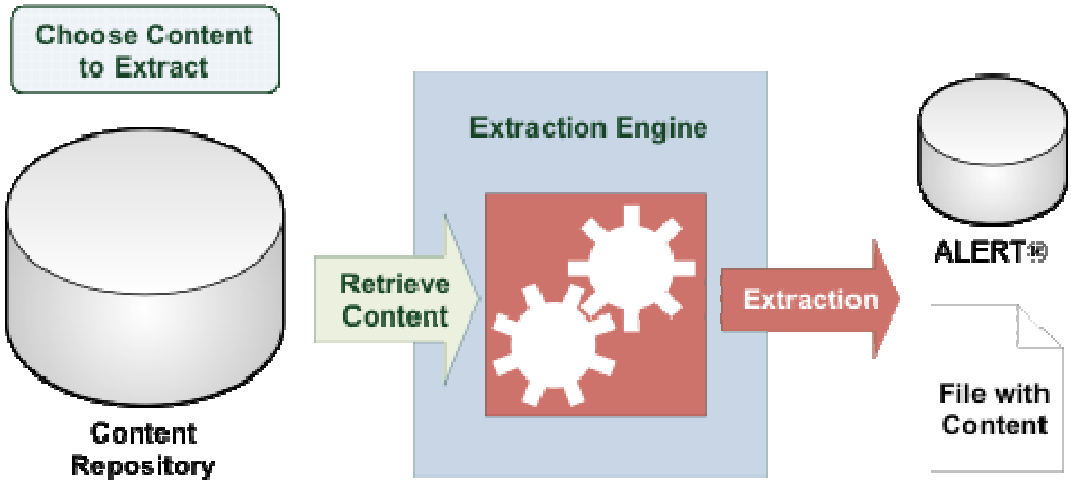


Figure 19: Overview of the extraction process

These functionalities of Content importation from different sources to the Content Repository and Content extraction from the Content Repository to different target databases representing different ALERT® product versions or file documents with different structures are what refer to Content Diffusion. Content Repository needs to be able to allow content diffusion in an automated way.

Chapter 4

Proposed Solution

This section describes requirements for the project and the proposed solution for meeting those requirements.

When we talk about import and extraction we are always talking from the Content Repository point of view. Import refers to Content going from outside world to the repository, and extraction refers to Content going from the repository to the outside world.

From this point on the proposed tool will be referred to as CIXE, which stands for Content Import and Extraction Engine.

4.1 Requirements Analysis

The proposed solution for this project was constructed based on a survey of requirements made to the Team I was integrated. This survey of requirements is based on model for specification of software requirements, using Use Cases from Rational Unified Process. Thus, the result of this survey consists of a model of Use Cases containing the main Use Cases to capture functional requirements of the system.

In this section you have the determined requirements from the survey that affected the proposed solution.

4.1.1 General Features

The main features required for CIXE are as follows.

1. **Database Access and Document Configurations**

This feature is responsible for managing all the access configurations to databases that CIXE needs to import Content from or extract Content to. CIXE lets the user specify a set of databases from which to obtain their metadata. This is the same for Excel/Flat Files metadata.

2. **Extraction, storage and presentation of metadata of a database or document**

CIXE extracts the metadata of the selected database or document and visually presents it to the user. The obtained metadata is stored in a repository, that way avoiding the need to extract that information in each use.

3. **Create Configurations of Mappings (with History)**

CIXE can associate the attributes of the database/document schema to attributes of the Content Repository. This process is called data mapping. For each mapping you can associate transformations responsible for transforming one structure to another. The application stores the mappings in a repository, enabling them to reuse. When a new configuration for a new version of a database is created, the application applies automatically all the mappings to the already known elements from the previous version.

4. **Import Content from Database or Document to Content Repository (with Log)**

Taking into account the mappings made by the user, CIXE allows extracting Content from the selected database or document and applying them the necessary transformations to change their representation from their format to the format of the Content Repository.

5. **Extract Content from Content Repository to Database or Document (with Log)**

Based on the mappings made by the user CIXE allows the extraction of Content from the Content Repository to other databases, using SQL-DML code generation, or to Excel/Flat Files documents.

4.1.2 Users Characteristics

The main aim of CIXE is to allow the creation of mappings that will then make possible to automate the Content import and extraction process.

Thus, it is primarily the **Development Team** who needs to perform mappings between the Content Repository and other databases/documents. It is thus necessary that the user of the system are familiar with the relational model of the Content Repository, as well as the schema of the database or document they are working with.

After this first mappings configuration phase, CIXE is ready to be used to automate the import and extraction processes by the **Content Team** who doesn't need to know anything about how the mapping process is done.

4.1.3 Actors

There are three actors for this system:

1. **User –** this is the user that is able to import Content to the Content Repository as well as extracting Content from the Content Repository to other databases or documents, using configurations previously created by the development team.
2. **Developer –** aside from inheriting the same requirements as the **User** this user can create mapping configurations to other databases or documents
3. **Administrator –** this user inherits all the requirements from the other users but also it is responsible for User Management and assigning correct roles to users

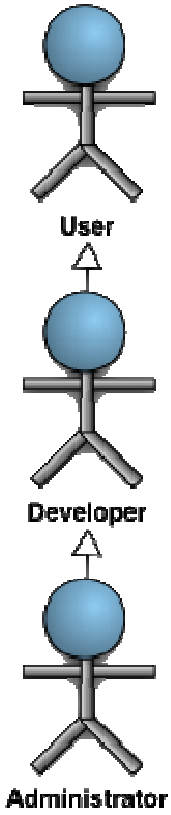


Figure 20: System Actor's inheritance representation

By default any user from the Content Repository have the role **User** associated with him that gives them the permission to import and extract Content.

4.1.4 Non-Functional Requirements

4.1.4.1 Reliability

The main non-functional requirement is reliability.

CIXE will have to handle a lot of information because of the thousands of Contents there are in the universe of ALERT®, even when dealing with only one type of Content. For this reason it is important to allow the flow of Content information to run without any kind of error being introduced in the application. That would compromise the quality of Content information transferred between Content Repository and other databases or documents, and most certainly transferred information would become useless for the user.

4.1.4.2 Efficiency

Other important non-functional requirement very important for this kind of tool is efficiency.

As it was mentioned earlier, CIXE needs to handle large sets of information, like import or extraction of all Contents of one pre-determined content type. This can be translated to thousands of Contents. Because of that, this tool needs to be as efficient as possible, in order to minimize the waiting time period users have until the end of the Import or Extraction process.

4.1.5 Functional Requirements

4.1.5.1 Packages Overview

CIXE is composed by six packages that refer to the main functionalities determined for this tool. In Figure 21 you can see an overview of all packages of this tool.

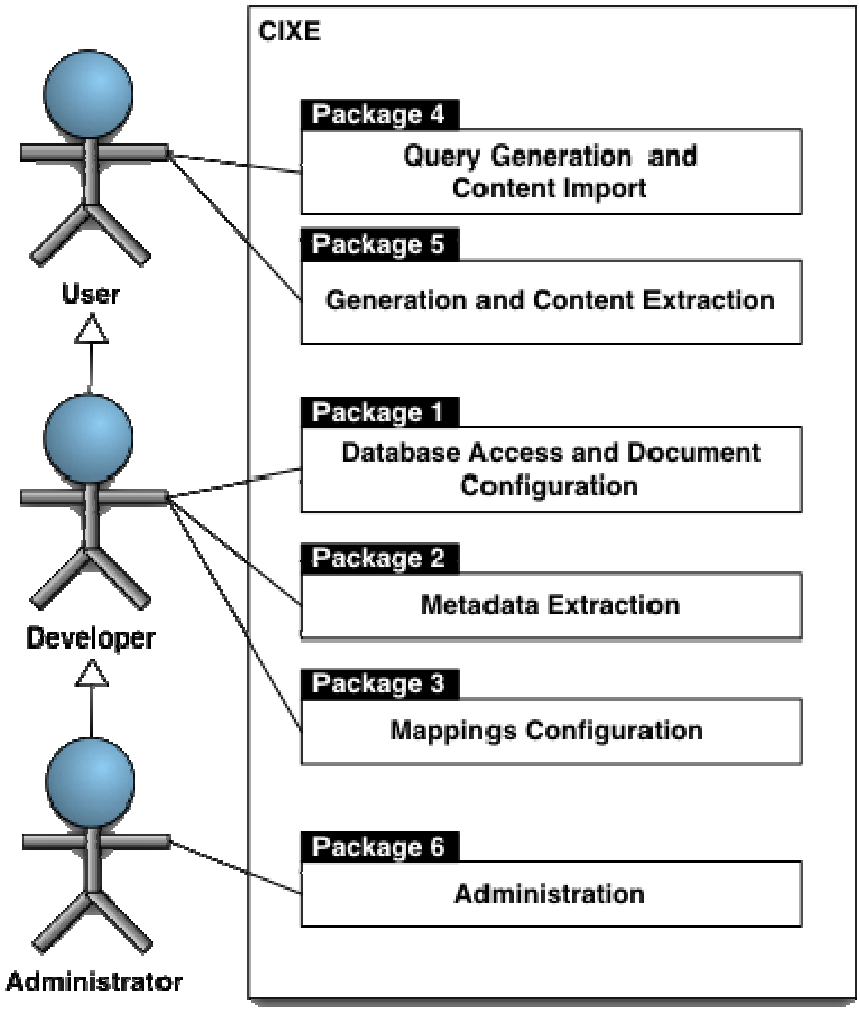


Figure 21: Overview of System's Packages

4.1.5.2 Use Cases

In this section are presented all use cases for CIXE system. From this point on CRUD refers to data operations for Creating, Retrieving, Updating and Deleting.

4.1.5.2.1 Package 1 - Database Access and Document Configuration

In this Package we have all use cases that have as main goal allowing creating configurations with access information to remote databases that have Content information of some interest to the repository. Also it has use cases for defining the types of documents that are allowed in the system.

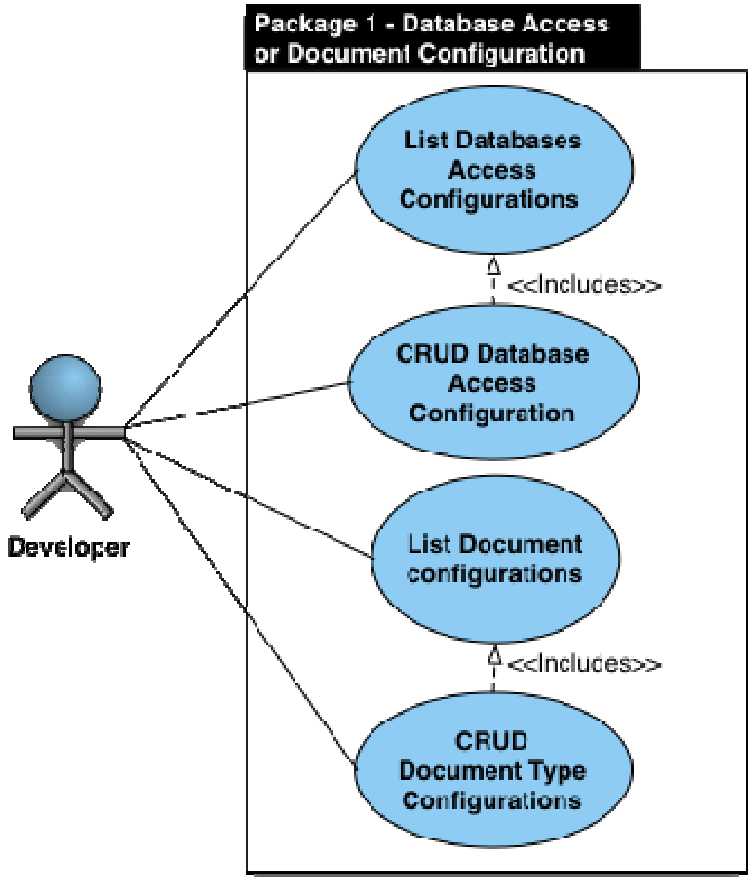


Figure 22: Package 1 Use Case Diagram – Database Access or Document Configuration

List Databases Access Configurations

Makes available to the user a list of access configuration to databases that have Content information of some interest to the Content Repository.

CRUD Database Access Configurations

After listing database access configurations, the **Developer** is able to create a new configuration, and update or delete existing configurations. Some examples of databases types are Oracle and Myself.

List Document Configurations

This use case makes available to the user a list of document types allowed to work with CIXE. It's up to the **Developer** to ensure all documents that have Content information of some interest to the Content Repository, have their corresponding type configuration created here.

CRUD Document Configurations

After listing document type configurations, the **Developer** is able to create a new configuration, and update or delete existing configurations. Some examples are Excel documents and Flat files.

In Figure 23 is presented the workflow for this configuration management. First **Developer** start the application and choose configurations option. After that **Developer** choose if he want to access the database access configurations or document types configurations. Depending on the option a list of database access or document types configuration are presented, and the ability to execute CRUD operations to these configurations are made available.

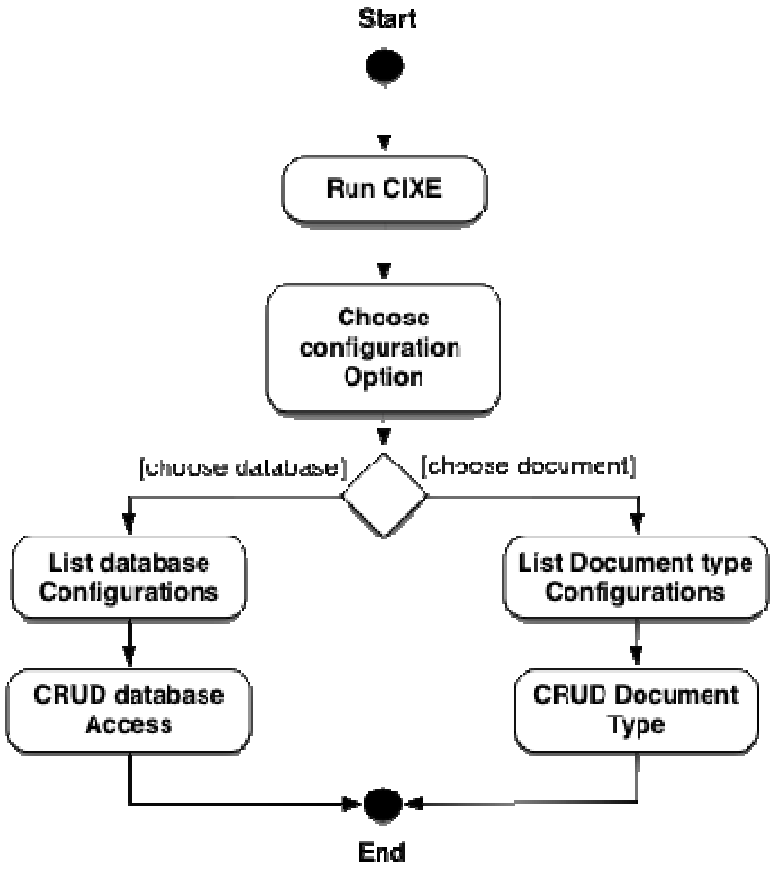


Figure 23: Package 1 – Listing and editing Configurations for database access and document types Activity Diagram

The workflow for CRUD operations for the two configurations types described earlier is the same. **Developer** is presented with a list of configurations and from that point the **Developer** can add a new one, fill the required information and save configuration. Or the **Developer** can select an existing configuration, edit its information and save changes, or simply delete it.

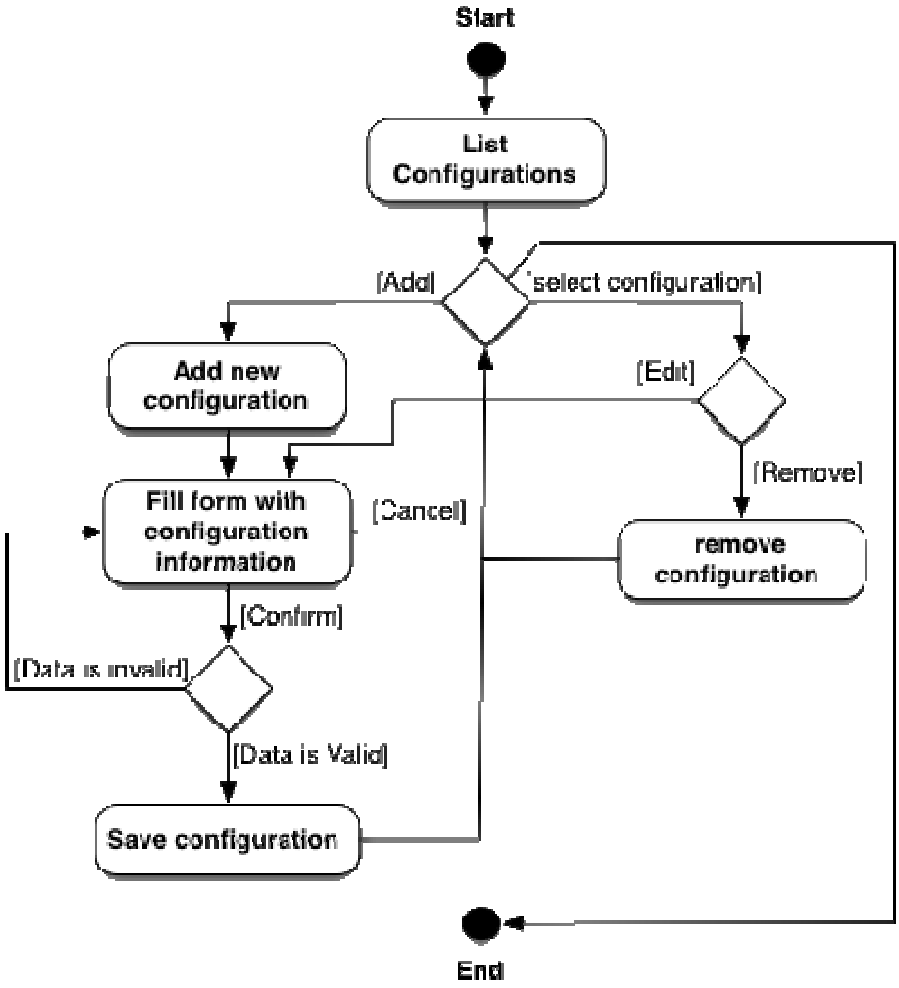


Figure 24: Package 1 – General Activity Diagram of Configurations editing for database access and document types

4.1.5.2.2 Package 2 - Metadata Extraction

This is the package of use cases responsible for the extraction of all metadata of the databases and documents CIXE needs to connect to. Also it is responsible for presenting that metadata in a format that allows mapping between this and the Content Repository metadata.

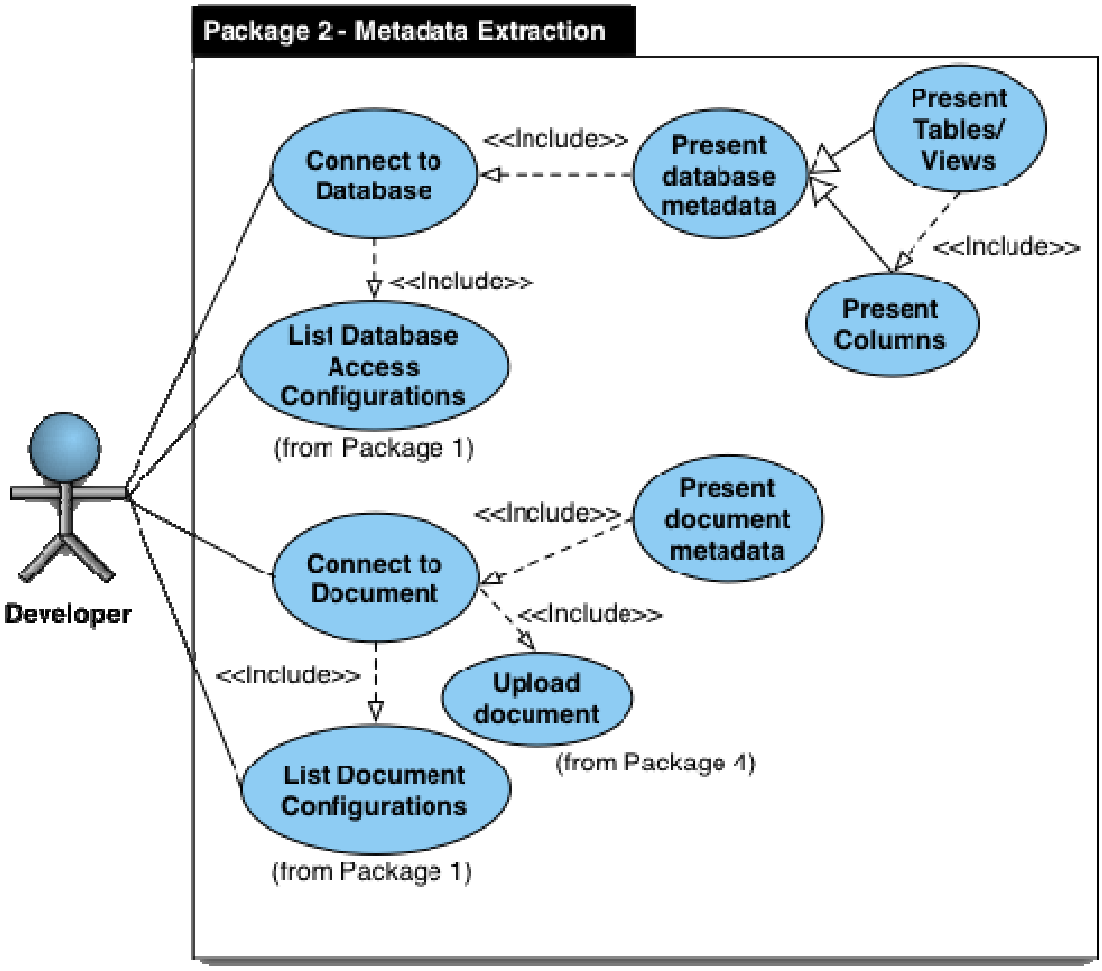


Figure 25: Package 2 Use Case Diagram – Metadata Extraction

Connect to Database

After retrieving access information for a database and depending on its availability, the application connects to the database in order to extract its metadata.

Present Database metadata

After connecting to a database, all its metadata, particularly tables, is visually presented in the screen.

Upload Document

Because this is a WEB-Based application, after selecting a document type configuration, **Developer** needs to upload a document. It is mandatory that the document be from the same type as the defined in the configuration.

Connect to Document

After uploading the document to the server, the application connects to the document and parses its contents in order to extract its metadata.

Present Document metadata

After connecting and parsing the document, its metadata is visually presented in the screen. Document metadata, particularly Excel and Flat Files, can be represented in the same way as databases metadata.

Excel

- Document is considered as a database

- Worksheets are equivalent to Tables
- Cells: First Line is column names; the rest are records
- Columns metadata are Excel columns

Flat Files

- Document is viewed as a table

- First Line is column names; the rest are records
- Columns have a separator character

The workflow for metadata extraction is done as shown in Figure 26.

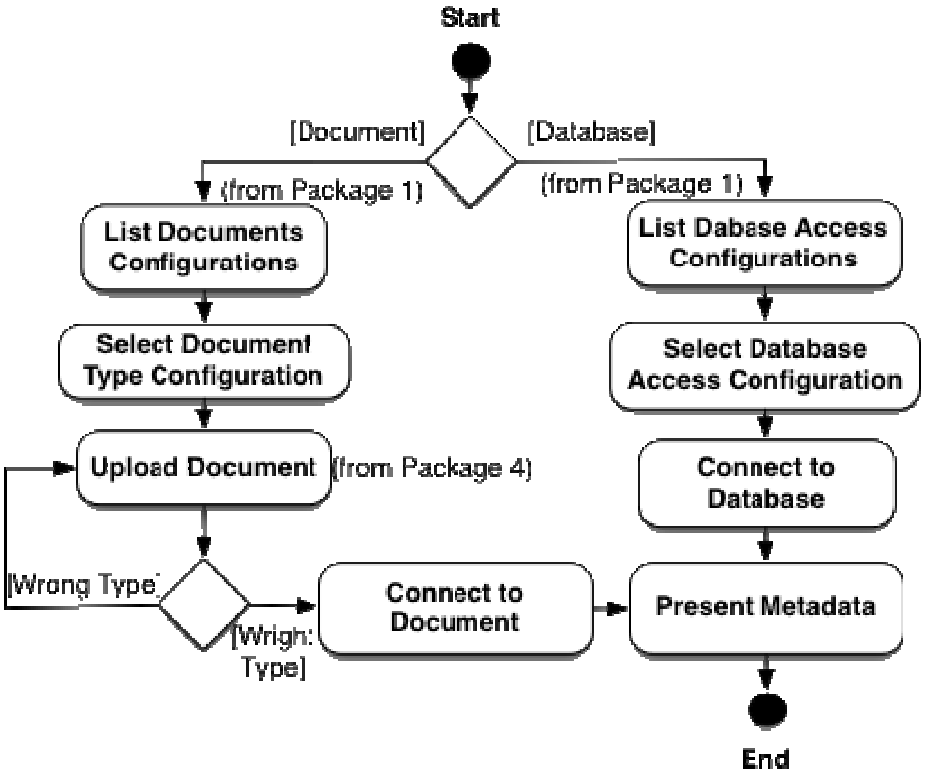


Figure 26: Package 2 – Metadata Extraction Activity Diagram

When we have a list of configurations we need to select the one we want to use. Selecting a document types configuration we then need to upload a document of the same type as the one required by the selected configuration, if valid we then connect to the file, parse its contents and present its metadata. If a database access configuration was selected a connection would have been made the database and its metadata presented.

4.1.5.2.3 Package 3 - Mappings Configuration

This package has all use cases that have as main goal help in the process of mapping information between Content Repository metadata and other Databases or Documents metadata.

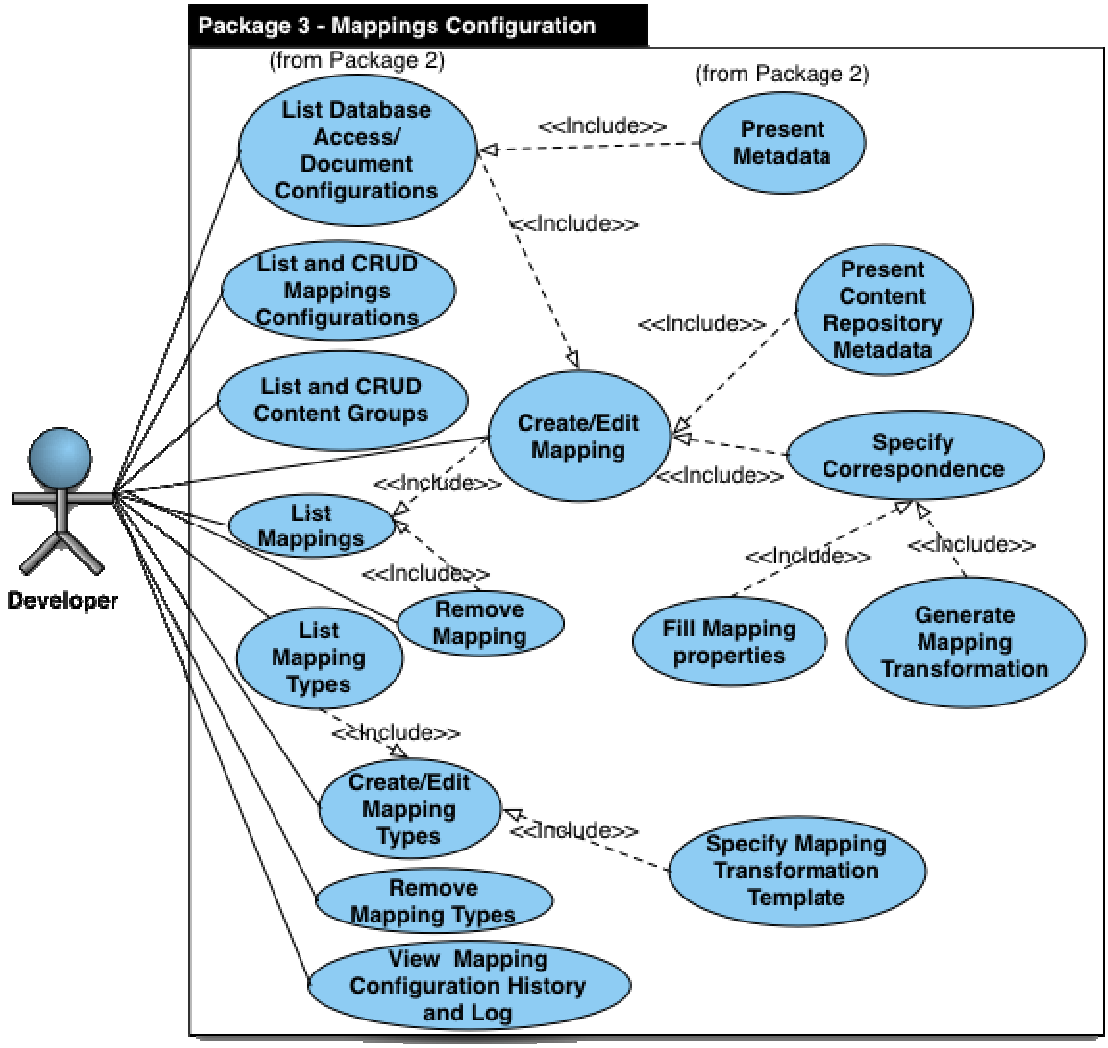


Figure 27: Package 3 Use Case Diagram – Mappings Configuration

List and CRUD Mappings Configurations

The **Developer** can list all available mappings configurations. He can create new mappings configurations, retrieve, and update or remove existing mappings configurations. A mapping configuration is what aggregates all specified mappings for Content Groups of one database access or document configuration. For one mapping configuration you have many possible Content Groups.

View Mapping Configuration History and Log

This Use case means that the **Developer** must be allowed to view a history of changes made to a mapping configurations and a log of imports and extractions for this configurations.

List and CRUD Content Groups

After selecting an existent mapping configuration or creating a new one, the **Developer** has to select the Content Group he wishes to specify mappings on. Content Groups refers to ALERT ®'s Content Universes such as Diagnoses, Analysis or Procedures. One Content Group aggregates all mappings specified for one of ALERT®'s Content Universe.

List and CRUD Mapping Types

In Figure 28 you can see an activity diagram for creating a Mapping Type.

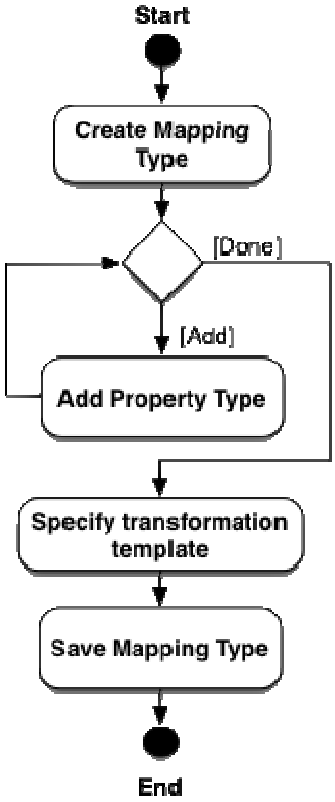


Figure 28: Package 3 – Create Mapping Type Activity Diagram

CIXE must allow the **Developer** to create Mapping Types that will have a transformation template associated with them. This allows the **Developer** to build Custom mappings types and define property types required for such mapping to take place.

Property types and transformation templates will allow the **Developer** to create mappings that are instances of this Mapping Types that will have their own properties and transformation. This Transformation is obtained by replacing mapping's properties values in the transformation template.

List and CRUD Mappings

As mentioned earlier in this section **Developer** must be able to create Mapping Types. After having Mapping Types, the **Developer** must be able to create instances of that Mapping Type for defining mappings, with transformations, between the Content Repository and other Databases or Documents.

The workflow for creating a mapping is represented in Figure 29. First **Developer** needs to choose if he wants to create a new mapping configuration or if he wants to work on an existing one. Then he should choose the Content Group he wants to work with. When inside a Content Group, **Developer** should be able to create mappings. When creating a new mapping, **Developer** should be presented with visual representation of Content Repository's metadata and the database/document's metadata this mapping configuration relates to. After choosing the Mapping Type to use, **Developer** needs to specify correspondence required for this mapping. To do that **Developer** then needs to fill the form with information required for the chosen Mapping Type.

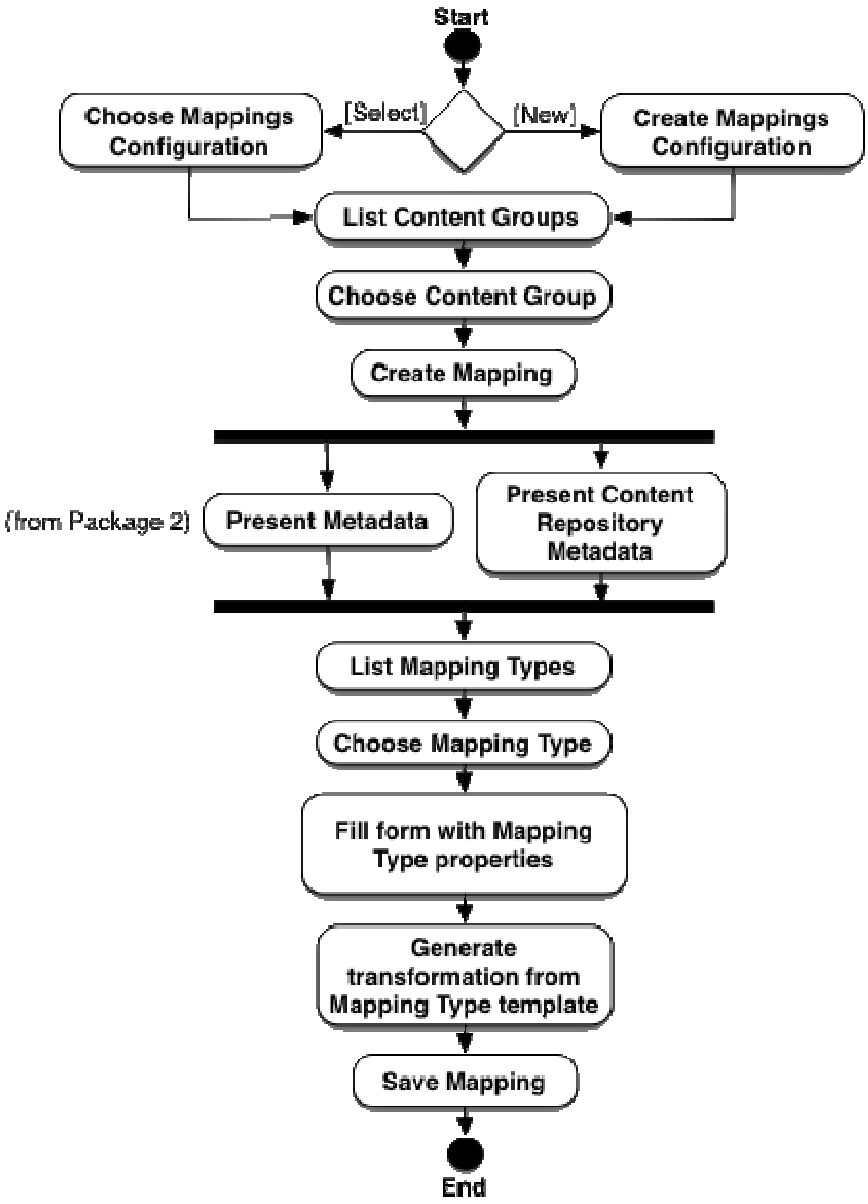


Figure 29: Package 3 – Create Mapping Activity Diagram

4.1.5.2.4 Package 4 – Query Generation and Content Import

This package has all use cases related to the Content Import process. This includes choosing what type of Content Group **User** wants to import and, in case of a document, upload and parse its contents while in case of a database, generating SQL queries of type select statements to query database for information. Both parsing and SQL queries generation is made based on mappings information.

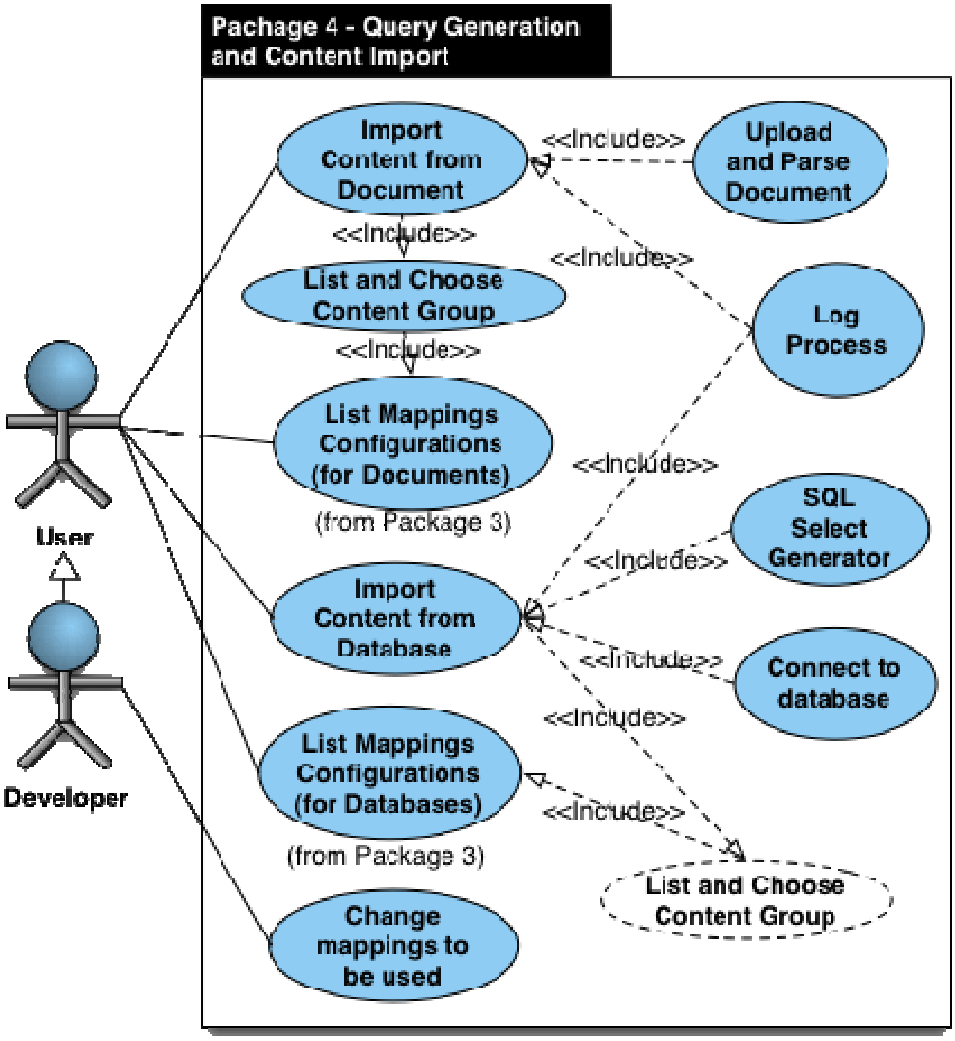


Figure 30: Package 4 Use Case Diagram – Query Generation and Content Import

Import Content from Document

When **User** wants to Import Content from a Document this document needs to have already one mapping configuration previously specified by a **Developer** already associated with its type.

Import Content from Database

When **User** wants to Import Content from a Database this database needs to have already one mapping configuration previously specified by a **Developer**.

Log Process

Every time an import process is run a log must be kept with important information such as user that ran the process, time, and all valid and error messages returned by the process.

SQL Select Generator

Generates a select statement to run on a database to query information about Content to import. This is done based on mappings.

Change Mappings to be used

Before running the import process, a Developer has also the possibility to choose what mappings he wants to be used for this process.

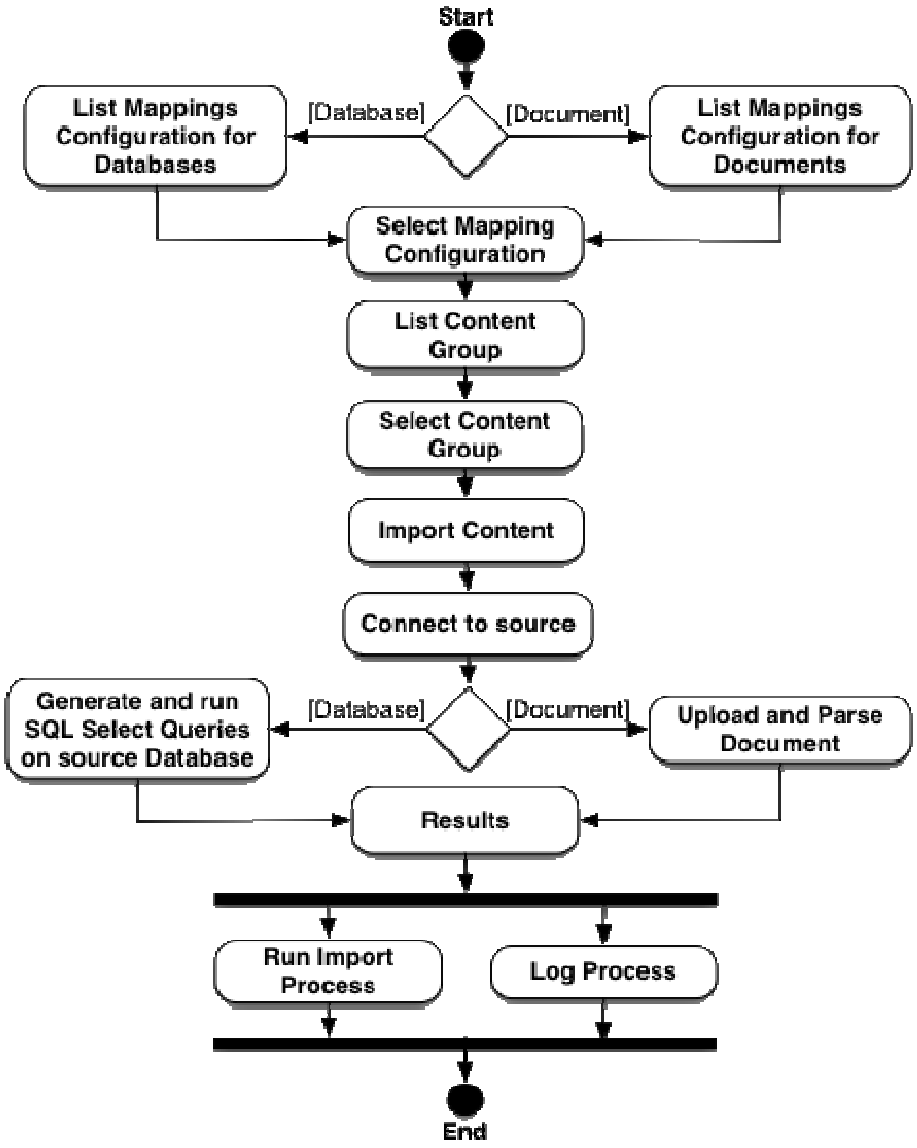


Figure 31: Package 4 – Content Import Activity Diagram

To better understand how the workflow for the import process should be done you can see Figure 31. **User** starts by choosing the mapping configuration they need for the Document

or database they want to import from (mapping configurations should be named in a way that it allows easy understanding for what they are for). **User** then selects from a list of Content Groups the one witch the source has information about. After that it chooses the import option.

If the source for previously selected mapping configuration is a Document, CIXE asks the **User** to upload a file witch then is parsed based on specified mappings information. If it is a Database then it will generate a query to retrieve required information, based on specified mappings.

4.1.5.2.5 Package 5 – Generation and Content Extraction

This package has all use cases related to the Content Extraction process. The main features present in this package are extracting Content to a document, by generating a structured document with Content information based on specified mappings, and to a database, by generating SQL DML statements (insert, update, delete) with Content information based on specified mappings.

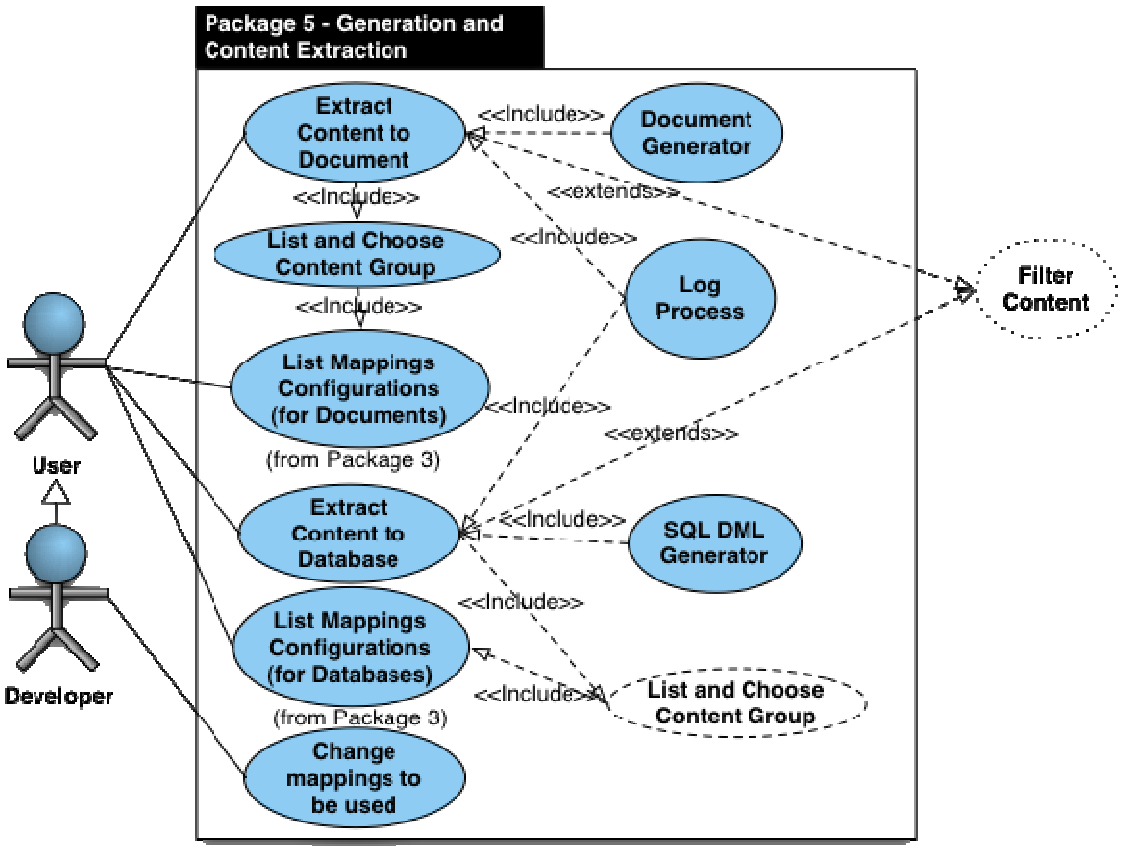


Figure 32: Package 5 Use Case Diagram – Query Generation and Content Extraction

Extract Content to Document

When **User** wants to Extract Content to a Document, a mapping configuration to the type it wants to extract to must already exist.

Extract Content to Database

When **User** wants to Extract Content to a Database this database needs to have already one mapping configuration previously specified by a **Developer**.

Log Process

Every time an Extraction process is run a log must be kept with important information such as user that ran the process, time, and all valid and error messages returned by the process.

SQL DML Generator

Generates SQL DML Insert, Update or Delete statements to run on a database to create the Contents to extract. This is done based on mappings.

Filter Content

The extraction process can also be ran at anytime in application runtime when **User** after specifying filters runs the extraction process

Change Mappings to be used

Before running the import process, a Developer has also the possibility to choose what mappings he wants to be used for this process.

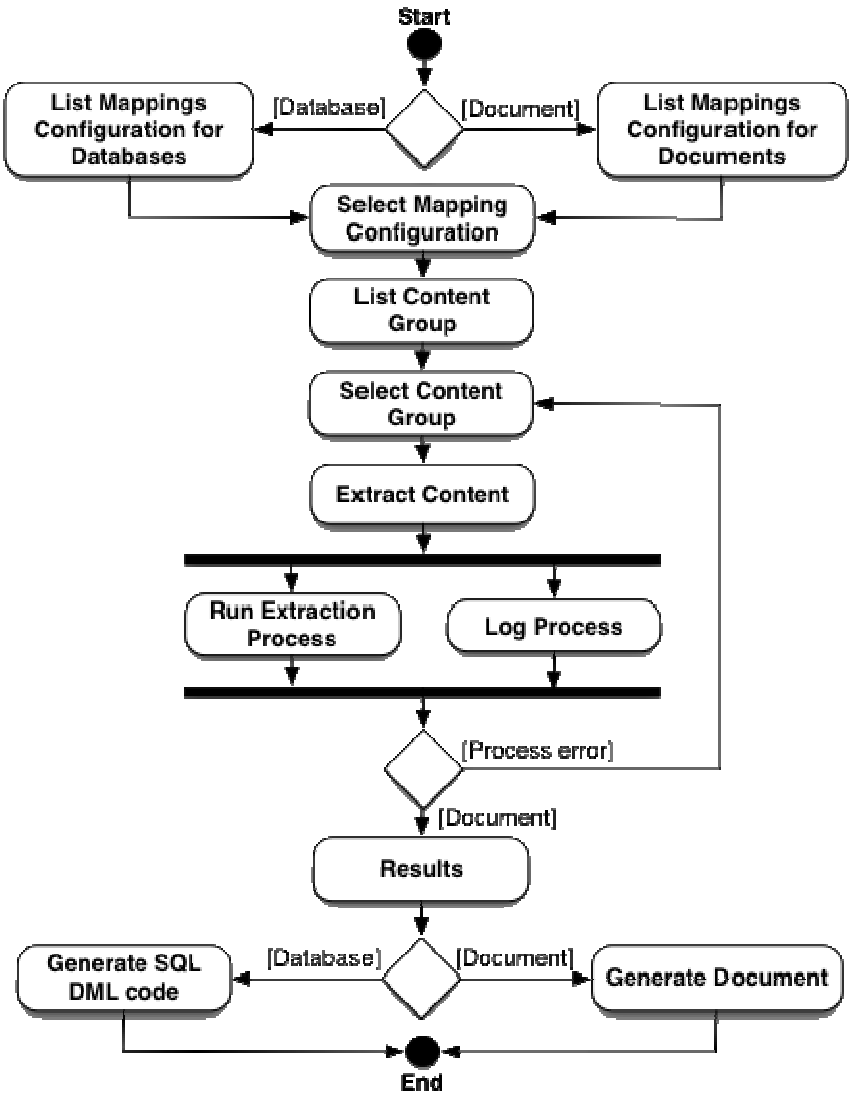


Figure 33: Package 5 – Content Extraction Activity Diagram

To better understand how the workflow for the extraction process should be done you can see Figure 33. **User** starts by choosing the mapping configuration they need for the Document or database they want to extract to (Mapping configurations should be named in a way that it allows easy understanding for what they are for). **User** then selects from a list of Content Groups the one he wishes to extract. After that it chooses the extract option. The Content is retrieve from the Content Repository.

If the target for previously selected mapping configuration is a Document, CIXE will generate a structured document based on specified mappings information. If it is a Database then it will generate SQL DML statements that will create the extracted Content on target database, based on specified mappings.

4.1.5.2.6 Package 6 – Administration

This package has the use cases that are responsible not only to create roles in this system and set its permissions but also list Content Repository Users and assign them roles.

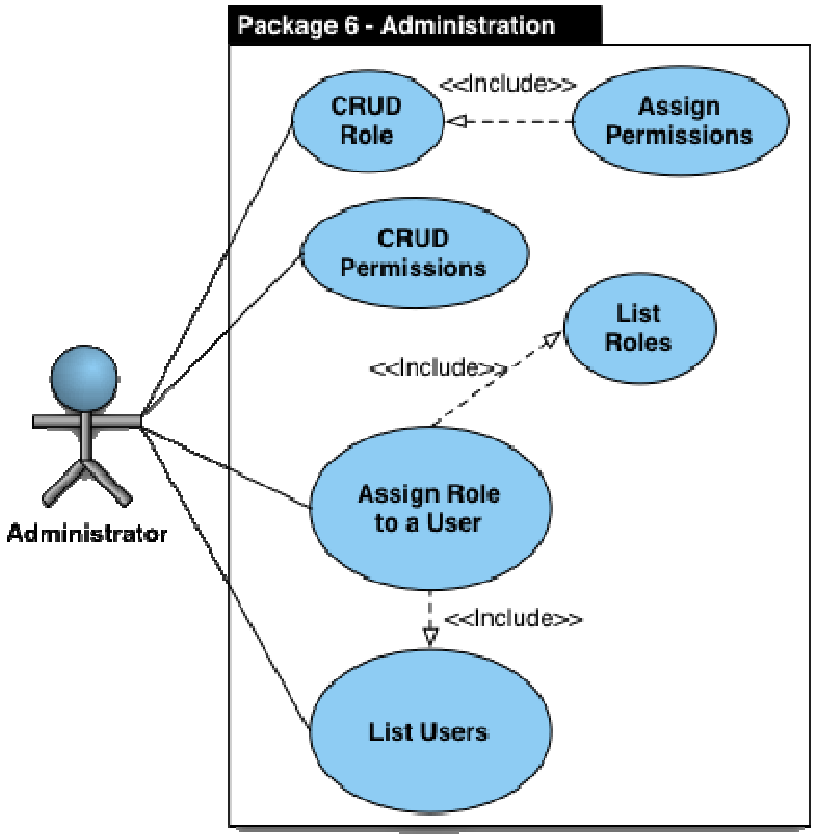


Figure 34: Package 6 Use Case Diagram – Administration

CRUD Capability

The **Administrator** can create capabilities that are required to limit access to some functionality.

CRUD Roles

The **Administrator** can create roles and specify for each capability if it's active or not.

Assign Role to a User

After listing existent users and roles, **Administrator** can assign a role to a user.

By default every user has a **User** role assigned.

4.2 CIXE Architecture Overview

From a conceptual point of view, CIXE needs to be a WEB based application to allow several remote users to access it at the same time.

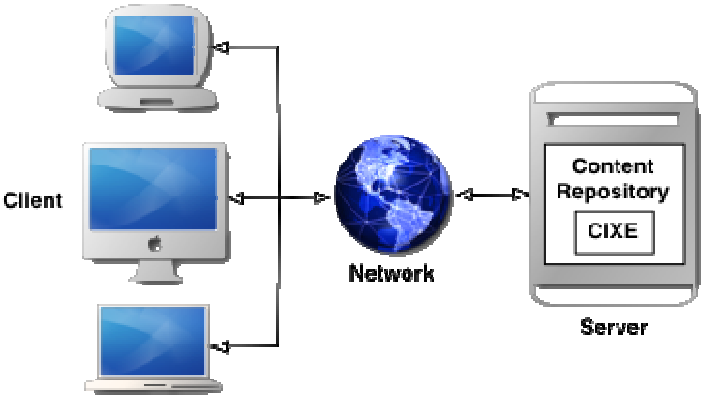


Figure 35: CIXE Client – Server Architecture

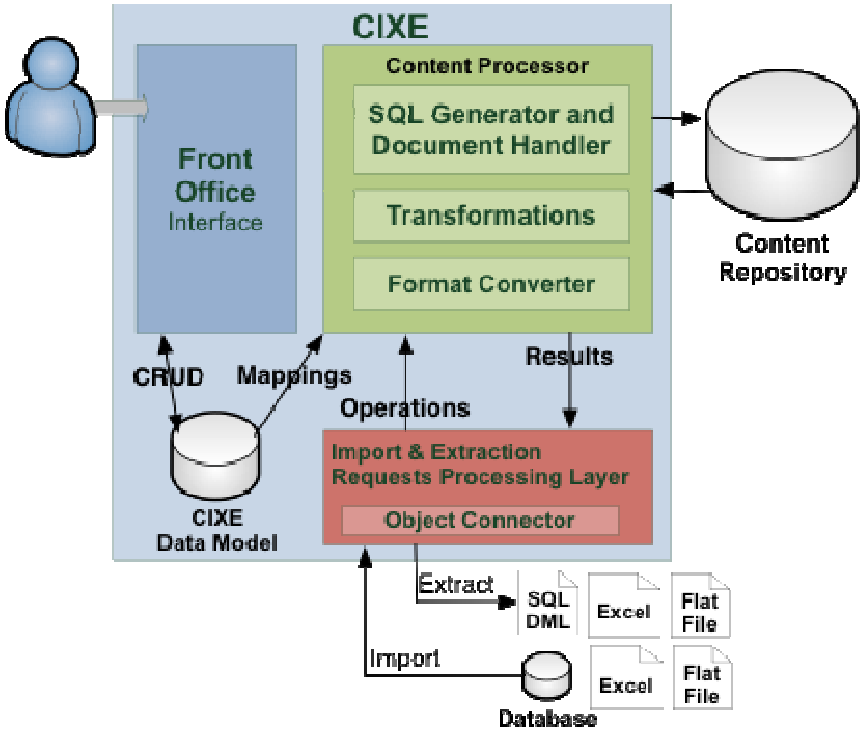


Figure 36: CIXE Architecture Overview

CIXE needs to be able to connect to other databases and documents to retrieve Content information and also to generate Documents and SQL DML scripts for databases. This Content information transference is possible by defining mappings and associate transformation with them, responsible not only for transforming between metadata but also allow other types of transformations such as, for example, multiplying two attributes before saving it to the Content

Repository. In Figure 36 you can see an overview of CIXE components. During the Import and Extraction process the Content information is represented in a generic format that is supported by the Format Converter component.

4.2.1 Front Office – User Interface

CIXE needs to have a Front office so that users can interact with the system and all of its functionalities. This component has all the interfaces for Use Cases determined during requirement analysis.

4.2.2 CIXE Data Model

In CIXE data Model it's stored all the important information for the application. This includes information about Mappings and configurations of mappings for external databases, documents, their access information, history and log for mappings configurations and administration.

4.2.3 Content Processor

The Content Processor is responsible for, for generating requests to retrieve Content information and for transformations between Content Repository and other databases or documents metadata.

4.2.3.1 Format Converter

Format Converter is responsible for converting imported Content information from databases and to a generic format and also to convert from that generic format to documents and databases.

4.2.3.2 Transformations

This component is responsible for applying Mapping's transformations to the transferred content. During the importing or extracting Content process, Content information is represented in a generic format, and to this format are applied transformations to transform one metadata structure to other metadata structure.

4.2.3.3 SQL Generator and Document Handler

With this component it is possible to generate SQL statements and also handle Documents' Content information.

4.2.3.3.1 During Import

During Import process, for databases, it is responsible for generating SQL Select statements necessary for retrieving information from other databases. As for documents it is responsible for parsing its contents. This parsing and generation is always done based on Databases and Documents metadata and mappings information.

4.2.3.3.2 During Extraction

During Extraction process, for databases, it is responsible for generating scripts of SQL DML statements, necessary for creating extracted Contents in other databases. As for documents it is responsible for generating its contents. This generation is always done based on mappings information.

4.2.4 Import and Extraction Requests Processing Layer

The Requests layer handles all Import and Extraction requests and is responsible for communicating and handling all the process with other layers.

4.2.4.1 Object Connector

Object Connector component is responsible for connecting to databases using access configurations, and for uploading the document to be processed for Content information.

4.2.4.2 Import and Extraction Process

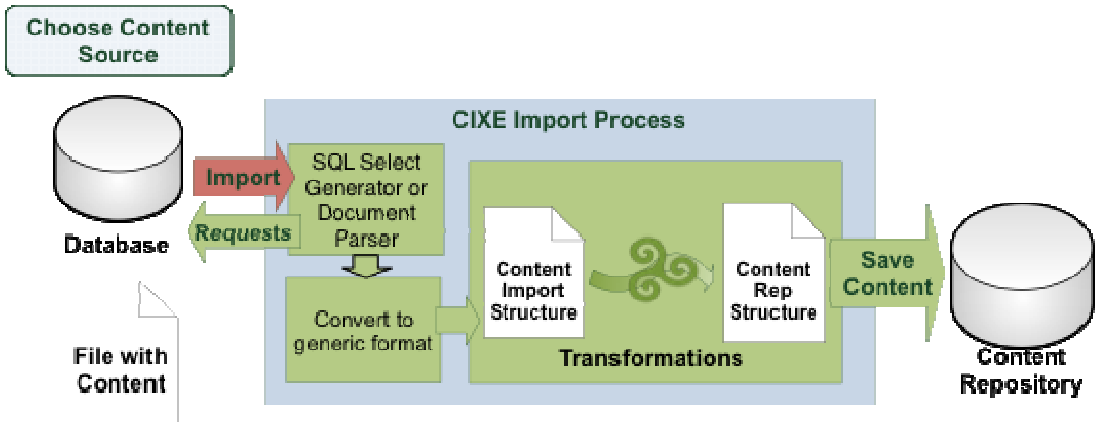


Figure 37: CIXE Import Process Overview

A conceptual overview of Import process is shown in Figure 37. After choosing Content source, the Content Processor then, if a database is the source, generates SQL requests, if a document parses its contents. This is always done taking into account mappings specified between them. After that its contents are converted to a generic format. To this format, mappings transformations are applied resulting in a structure equivalent to the Content Repository metadata, allowing the application to save Contents to the repository.

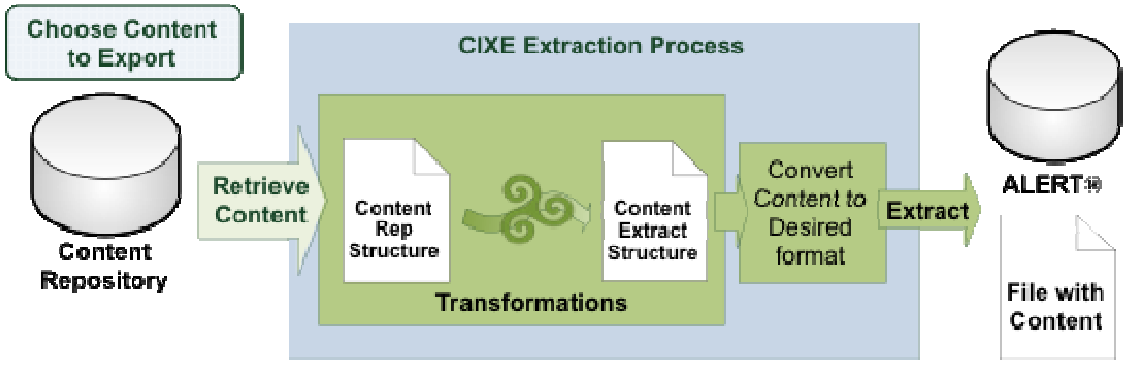


Figure 38: CIXE Extraction Process Overview

Figure 38 shows a conceptual overview of Extraction process. After choosing Content to Export, CIXE retrieves Content information from the Content Repository. At this time this information is with Content Repository metadata structure. To this format mappings transformations are applied resulting in a structured equivalent to the desired for extraction, allowing the application to generate SQL DML statements or documents with Content information.

4.3 System Modelling

Based on requirement analysis it was determined that CIXE has mainly five system logical units:

1. **Database and Document Metadata Repository**

This repository is responsible for allowing representing, in its structure, metadata information related to documents or databases that Content Repository has mappings specified.

2. **Mappings Configuration Repository**

In this module are saved all information about mappings and transformations necessary to enable CIXE to know how it can map Content information between other databases and documents metadata.

3. **Mappings Configuration History**

Because there is a necessity of keeping an history of mapping configurations, this module is responsible for that. This way it is possible to know every change that was made for each mapping configuration and what user was responsible for that.

4. **Import and Extraction process Log**

Import and Extraction process lead with enormous amount of data, something can go wrong and make the process end badly. Because of this type of problems, it is imperative to keep a log of this process. CIXE must be able to keep a log of any problems that occur during the process, with messages that can help to understand what happened, and then more easily solve the problem.

5. **Role Management**

This is responsible for keeping information about CIXE capabilities, representing roles available in the system and allowing assigning them to Content Repository Users.

All of the described modules belong to CIXE domain and will be described in the following sections using UML Modelling. For a final representation of the complete UML model of this please refer to Appendix A.

4.3.1 Database and Document Metadata Repository

In Figure 39 a representation of the class diagram of the structure used so save all information about the objects which metadata can be mapped to the Content Repository.

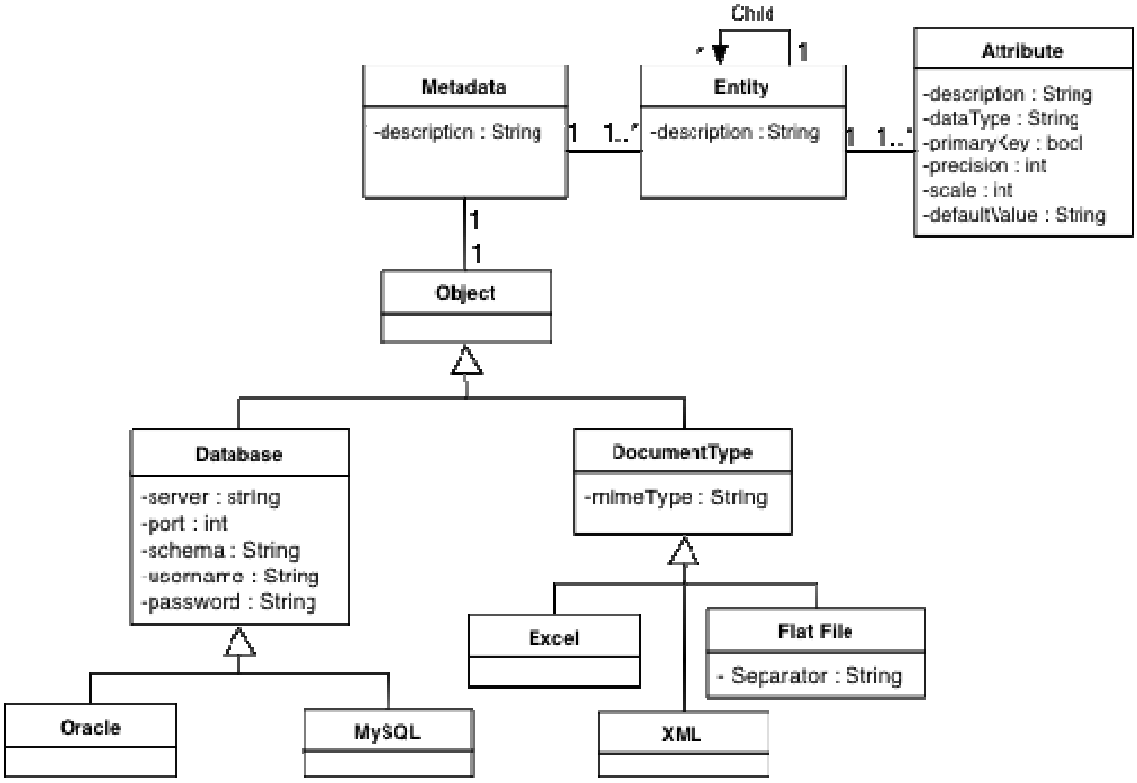


Figure 39: Database and Document Metadata Repository UML Model

The Objects that need to be mapped to the Content Repository are Databases and Documents.

A database has important information required to allow CIXE connecting to it such as server name, the port it uses for communication, the name of the schema (this is where the metadata and Content information are), the username and the password used authenticate in the database server.

As for documents, because CIXE doesn't have to save the documents itself the only information that's necessary to save is the type of documents that are to be available to use with CIXE. A document can be of different formats but the main formats that required to be supported by CIXE are Excel documents, and Flat Files. For Flat files there is a need to know which is the character used to separate information.

An Object has associated Metadata. Metadata for database and documents like Excel's and Flat files can be represented with a set of Entities (tables) and Attributes (columns). Every Entity can have a set of Attributes. Required Attribute information is its name (description), data

type, if it is a primary key, precision and scale, if a numeric value, and attribute's default value. Because of XML documents metadata, it was added a self-association for Entity class named child. XML Entities can have other Entities as Childs, and so on.

4.3.2 Mappings Configuration Repository

Mappings Configurations are what allows the user to create configurations with groups of mappings and transformations necessary for mapping Content information between databases/documents and the Content Repository.

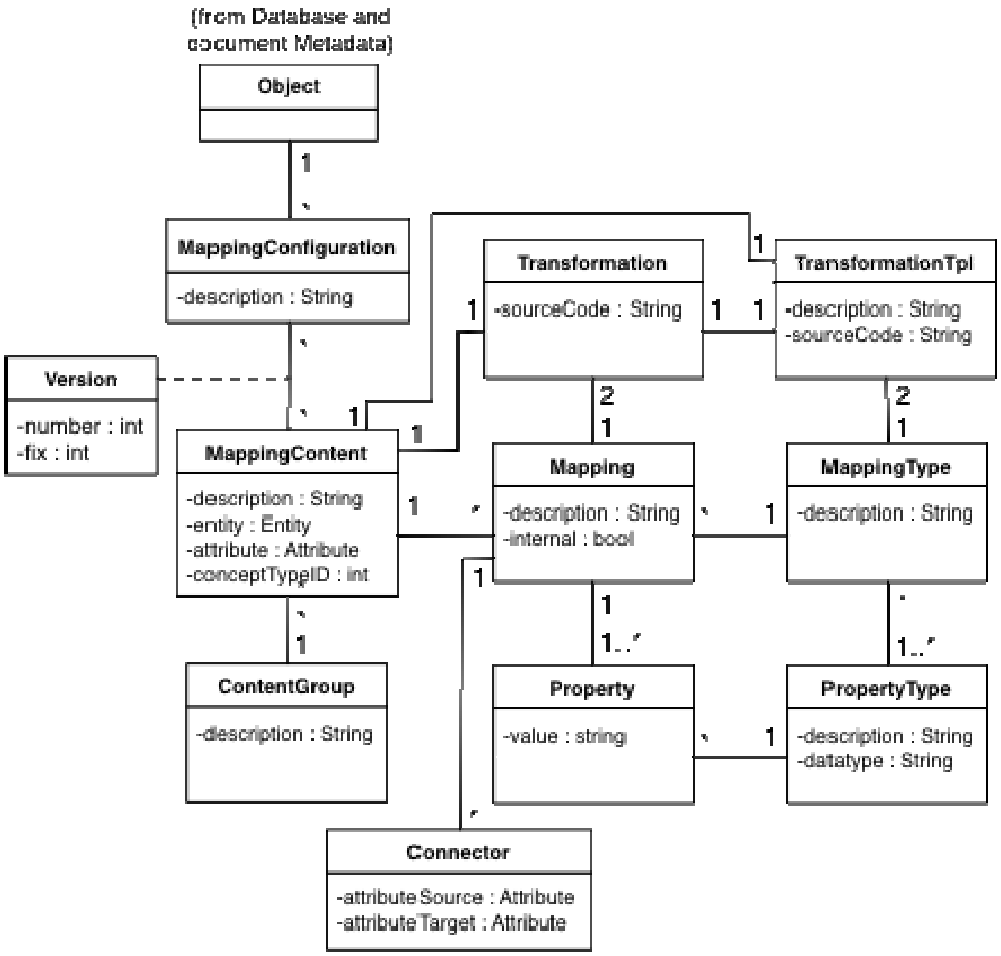


Figure 40: Mappings Configuration Repository UML Model

Several MappingConfigurations can be created in accordance to the necessities of the users. Something to take into account is that these configurations can be made for different Versions of ALERT® products and Object configurations can be used in many Versions. To start, MappingConfigurations can have several MappingContents. A MappingContent class allows mapping to a certain Content Entity of a database or document, which attribute works as an ID for the Entity and specify to what type of concept this Entity refers to.

ContentGoup allows users to group MappingContent by what Universe of Contents of the Content Repository they belong to such as, for example, Diagnoses, Analysis or Procedures group MappingContents. In the Content Repository, inside that type there are lots of subtypes of Contents and those are identified by conceptType.

MappingContents can be reused in different MappingConfigurations so, for example, when creating a new MappingConfiguration for a new product Version, that possibly comes with a new object with new metadata, users can specify a source MappingConfiguration and business logics are responsible for checking compatibility of each MappingContent with the new Object's metadata, saving the ones that are really necessary. MappingContent can have many Mappings defined for the Entity it refers to.

Mappings allow user to specify correspondences between Content Repository's Attributes and other databases or documents' Attributes. These Mappings can be internal or not, meaning that if it uses only information from the mapped Entity, it's internal, but if also uses information related information from other Entities it's external. Being external, a Mapping has Connectors that allow specifying the attributes that relates the two Entities. To each Mapping 2 Transformations are associated, one for Import and another one for Extraction.

4.3.2.1 Custom Mapping Types Support

Using as basis the Adaptive Object pattern 41 as an approach, CIXE supports the functionality of allowing users to create new custom types of mappings. TypeObject [TOP09] pattern allows dynamically specifying new business entities to the system in runtime. In this case TypeObject is used to separate mappings of mapping types. Mappings have properties, which are also implemented with TypeObject pattern using PropertyType. Specification of new types of mappings can be made by an user interface. The final result is what is called TypeSquare and is represented in Figure 41.

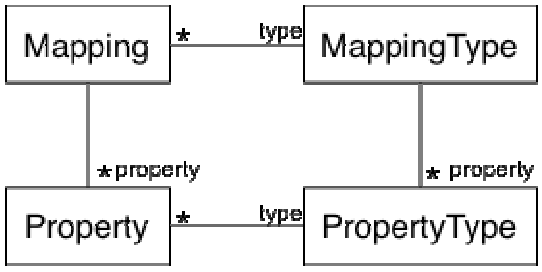


Figure 41: Mapping Types TypeSquare Diagram

With this kind of representation it is possible to CIXE to have an environment where users not only can create new MappingTypes and PropertyTypes they think are necessary, but

also create instances of that types to be used in Mappings Configurations. Also following the example of TypeObject patern, transformations and transformation templates (types) exist.

Every MappingType has two transformation files templates associated with it, one for import and another one for extraction. PropertyTypes of one MappingType can be used in this template. PropertyTypes in the transformations source code are represented with their name and in a format that allows to easily identifying them.

When creating a new Mapping of a certain MappingType, all the Properties values defined for that Mapping (based on MappingTypes' PropertyTypes), are used to replace its correspondent PropertType occurrence in the transformation template source code.

4.3.3 Mappings Configuration History

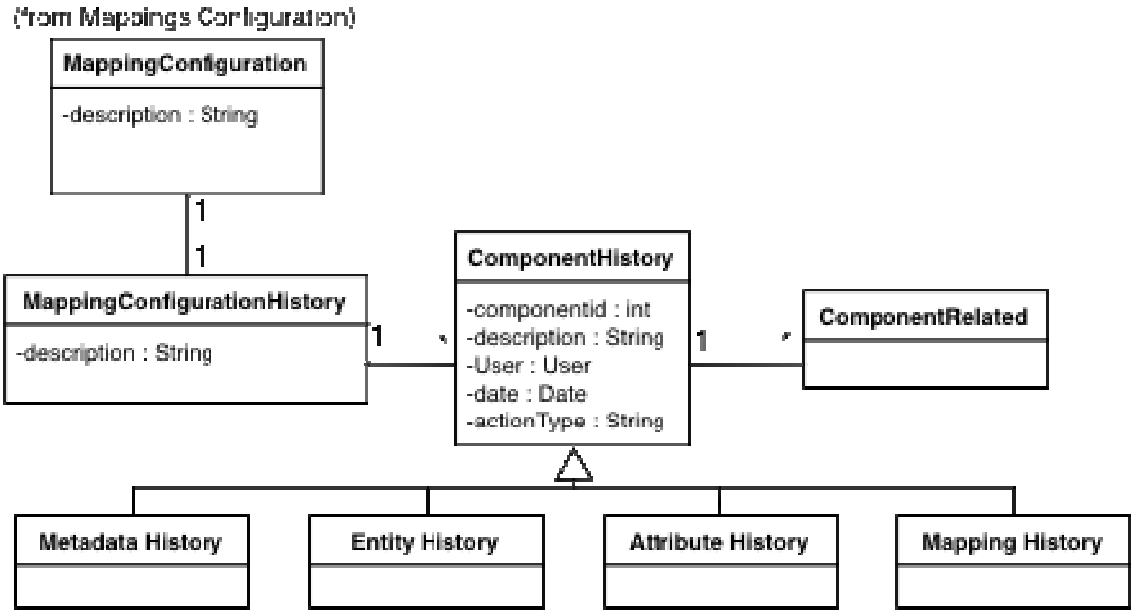


Figure 42: Mappings Configuration History UML Model

It is important to have the possibility of viewing what changes users made in Mappings Configurations. If something wrong happens during import or extraction when processing some Mapping, there is a way of checking past changes. That might be enough to find the reason why something went wrong.

Each MappingConfiguration has a MappingConfigurationHistory associated with it. This MappingConfigurationHistory has many ComponentHistories.

ComponentHistory is an history of changes of a certain component of the database that relates to all different components that directly and indirectly interact with MappingConfigurations. A History of Metadata, Entity, Attribute and Mapping changes is maintained. Because of situations that an user action such as, for example, replacing an attribute

by another, this record of Component History would have one more ComponentRelated for that action. ComponentRelated is generically representing Metadata, Entity, Attribute and Mapping classes.

4.3.4 Import and Extraction process Log

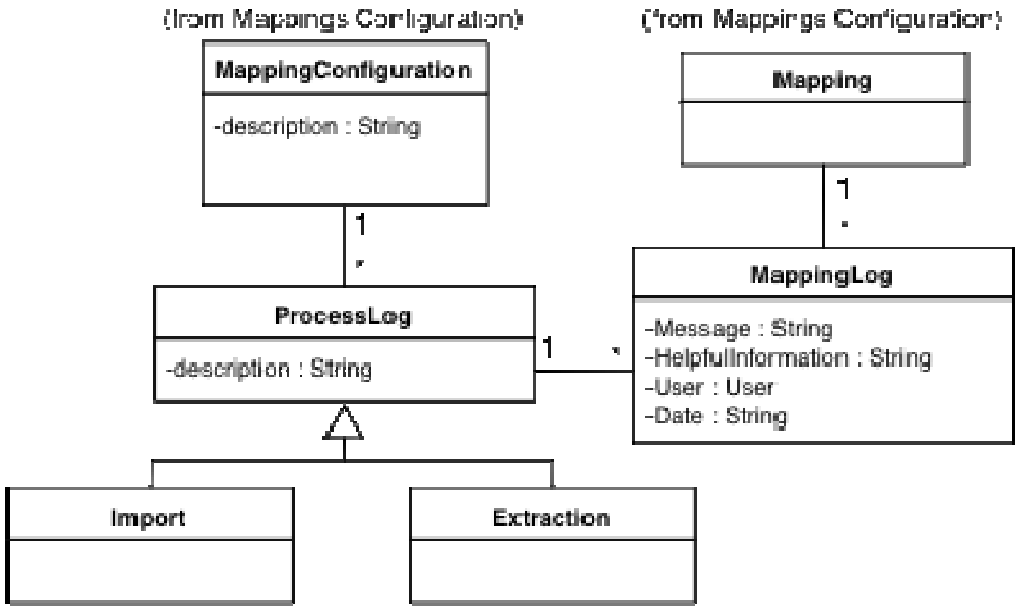


Figure 43: Import and Extraction process Log

CIXE must be able to keep a log of any problems that occur during the process, with messages that can help to understand what happened, and then more easily solve the problem. Having that in mind a MappingConfiguration have many ProcessLog where all error, or alert messages are stored.

ProcessLog Class is a Superclass for Import and Extraction process Log. The Import and Extraction process are made based on mappings, because of that reason the ProcessLog has many MappingLog that keep information important for the logging process such as generic Message to be presented to the user, HelpfullInformation that contain error messages from the business logic with extra information (this can be a possible cause for this situation, or a Stack Dump), the user that ran the process and the date that this occurred.

4.3.5 Role Management

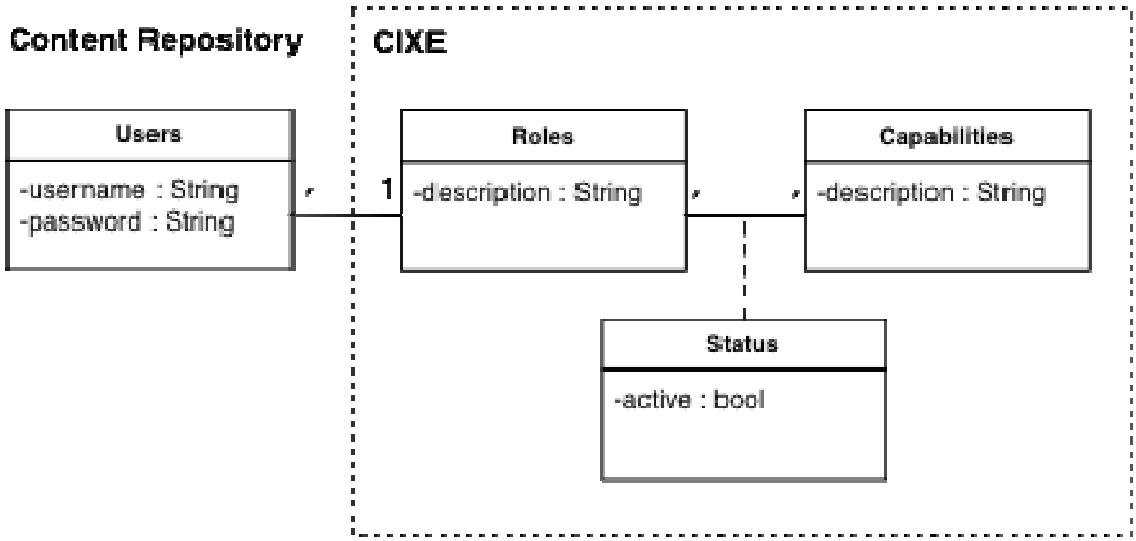


Figure 44: Role Management UML Model

Content Repository haves a user management package. This is responsible for keeping information about CIXE capabilities, representing roles available in the system and allowing assigning them to Content Repository Users.

Capability represents functionality. A Role has defined for each Capability if it is active or not. Roles are then assigned to users

4.4 Summary

After surveying for requirements the team that will need the tool, and determining all use cases, an architecture and domain model for the tool were proposed to meet those requirements.

The proposed solution for allowing import to the Content Repository and extraction from the Content Repository is a web application that has an underlying Configurations module, with log and history, responsible for the management of mappings and transformations between Content Repository and other databases or documents metadata with support for new mapping types.

Mappings and transformations are required for the tool to know what to do with the data from one point to another in the import and extraction process. These are used in Import process to specify what Content information to query from databases or to parse from documents as well what transformations to apply to them. In Extraction process, these are used to specify rules needed to transform information into the target metadata for then creating extracted Contents on a document with a parser, or on a database with SQL DML Scripts.

Chapter 5

CIXE - Prototype Development

A fully functional prototype of CIXE was developed in order to make a proof of concept for the proposed solution. Here are presented the architecture, technologies and implementation decisions for prototype implementation.

5.1 Implementation Decisions

5.1.1 XML as Generic format during Import and Extraction process

The generic format referred in section 4.2 used for representing Content information during Import and Extraction process is required for representing Contents from any format in a format that is unique without loss of information.

The chosen format for this representation was XML because its structure allows working with an Entity and Attribute based metadata and apply rules of transformations from a XML representation to a different one by using XSLT.

5.1.2 XSLT for Mapping and transformations

For mechanism of Content Import and Extraction it was decided to use mappings and transformations of metadata between the source and the target from this process. This is done using XSLT as basis for implementing transformations to metadata structure. It is possible to

make use of its possibilities for applying not only metadata transformations but also operational transformations, such as, adding two attribute's values and define a resulting metadata.

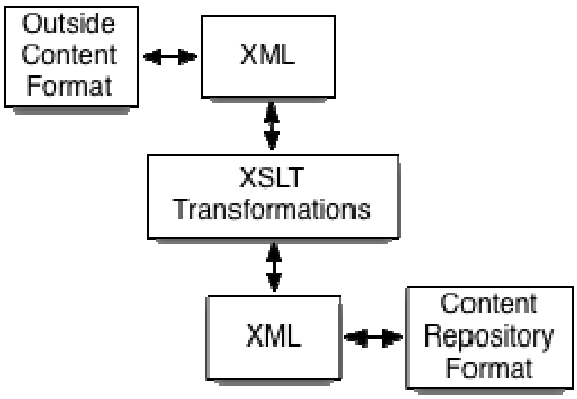


Figure 45: Content Information Representation Overview

Outside Content information format is converted to XML format, then XSLT transformation rules defined in mappings are applied to it and a new format is generated which has the same metadata as the Content Repository.

5.1.3 Adobe FLEX Framework for interface

Adobe Flex was a requirement because it's the user interface of the Content Repository and will be used in all future ALERT® applications.

5.1.4 Java as business logic and middleware

CIXE has a lot of XSLT processing in its backbone so Business logic is implemented in Java SE because current XSLT processor in XSLT latest version (2.0) with the best performance is Saxon [SAXN09], a Java SE based XSLT processor.

5.1.5 Hibernate for ORM and XRM

Hibernate [HBNT09] was chosen for data persistence between Java and Database layer because it allows the development using Object Relational Mapping between Java SE classes. One of the main features that were the reason for choosing Hibernate is its capability for XML Relational Mapping. This allows automatically mapping a XML Document's metadata to a database metadata. This helps in the Import and Extraction process because Content Repository information in XML format can be automatically mapped to the database layer.

5.1.6 Oracle as database

Oracle Database is the default database used by all ALERT ® applications and was a requirement because it's the database used in all ALERT® applications.

5.2 Implemented Architecture

This section talks about the architecture defined for implementing this solution.

5.2.1 Logical Architecture and integration with Content Repository

Logical architecture for CIXE is divided in three layers: User Interface, Business Logic and Database.

One thing that was asked to take into account is the use of open source technologies in my decisions of technologies to use, but also I could use technologies that they already have licenses to, which is the case for Oracle Database and Adobe Flex.

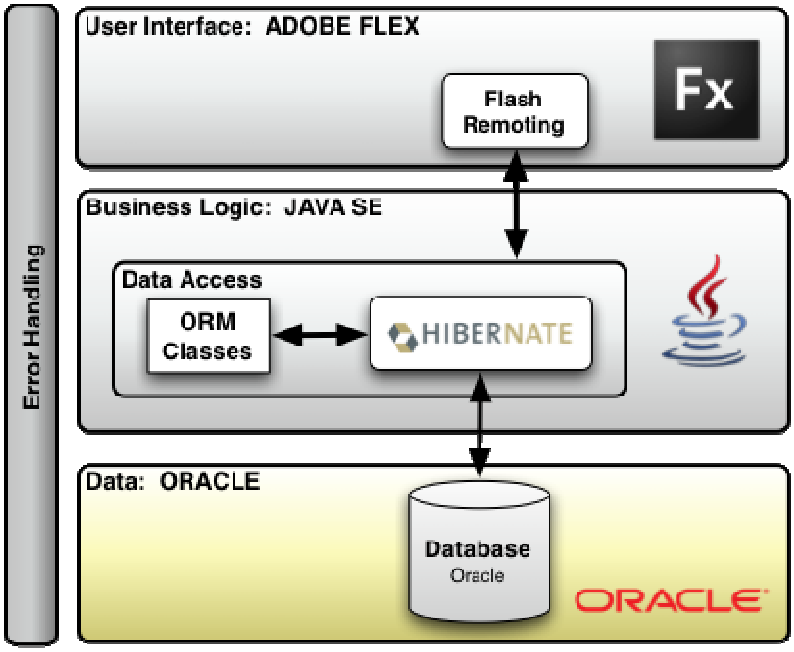


Figure 46: Logical Architecture Diagram

Content Repository uses for its user interface Adobe Flex so this had to be the same for CIXE Module User interface. Interface layer communicates with Business Logic layer using Flash Remoting. Flash Remoting is a Flash architecture that allows Flex to access business rules defined in the Business Logic Layer.

Java was chosen for the reasons referred in section 5.1.4.

Hibernate was chosen for the reasons referred in section 5.1.5.

Database layer uses Oracle Database, for the reasons referred in section 5.1.6.

5.2.2 Physical Architecture

Content Repository CIXE Architecture is based in a Client – Server configuration.

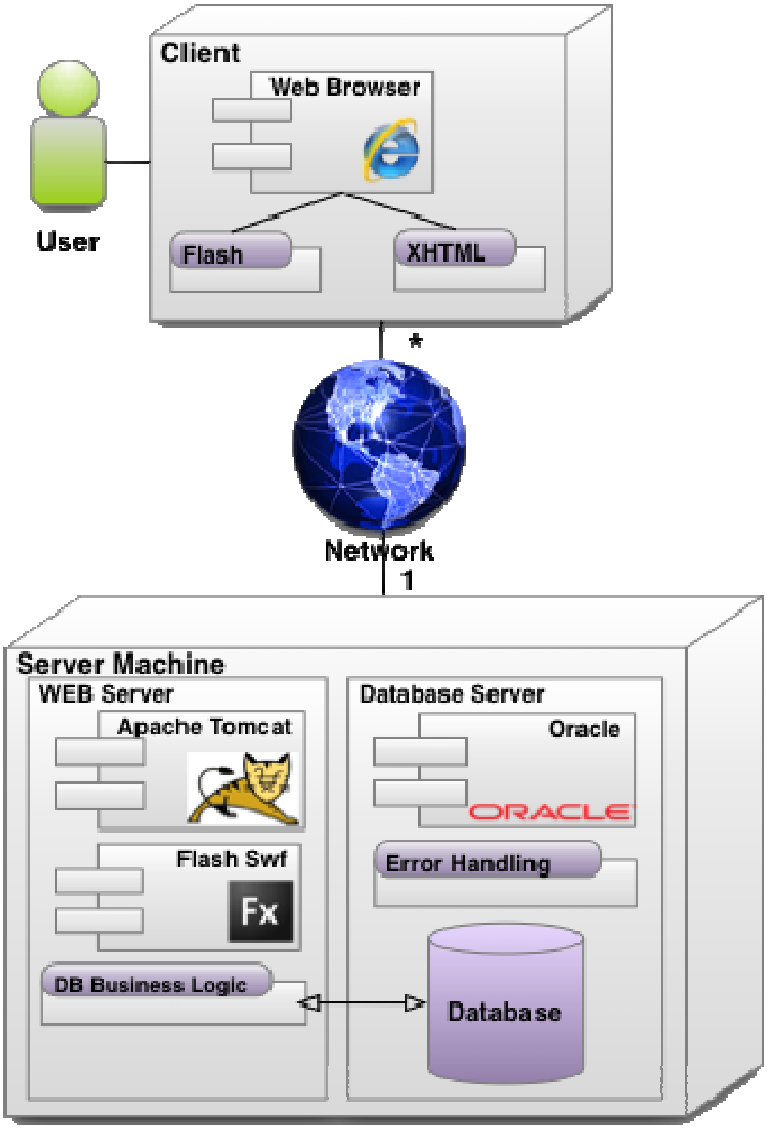


Figure 47: Physical Architecture Diagram

As the Content Repository, CIXE has a physical dispersion that consists in a user being in a workstation that accesses the application using a WEB Browser that is compatible with HTML and Flash technologies.

CIXE application file is in a Server Machine with a Web Server responsible for making this application available to remote users. ALERT's default Web Server for applications with an Java SE server side component for business logic, which is the case of CIXE, is Apache Tomcat

[TCAT09]. This Server Machine also has a Database Server with Oracle responsible for data management.

5.3 User Interface Layer

CIXE interface was implemented in Flex and Actionscript as interface's backoffice. The application is composed by lots of screens that illustrate implemented features logical units.

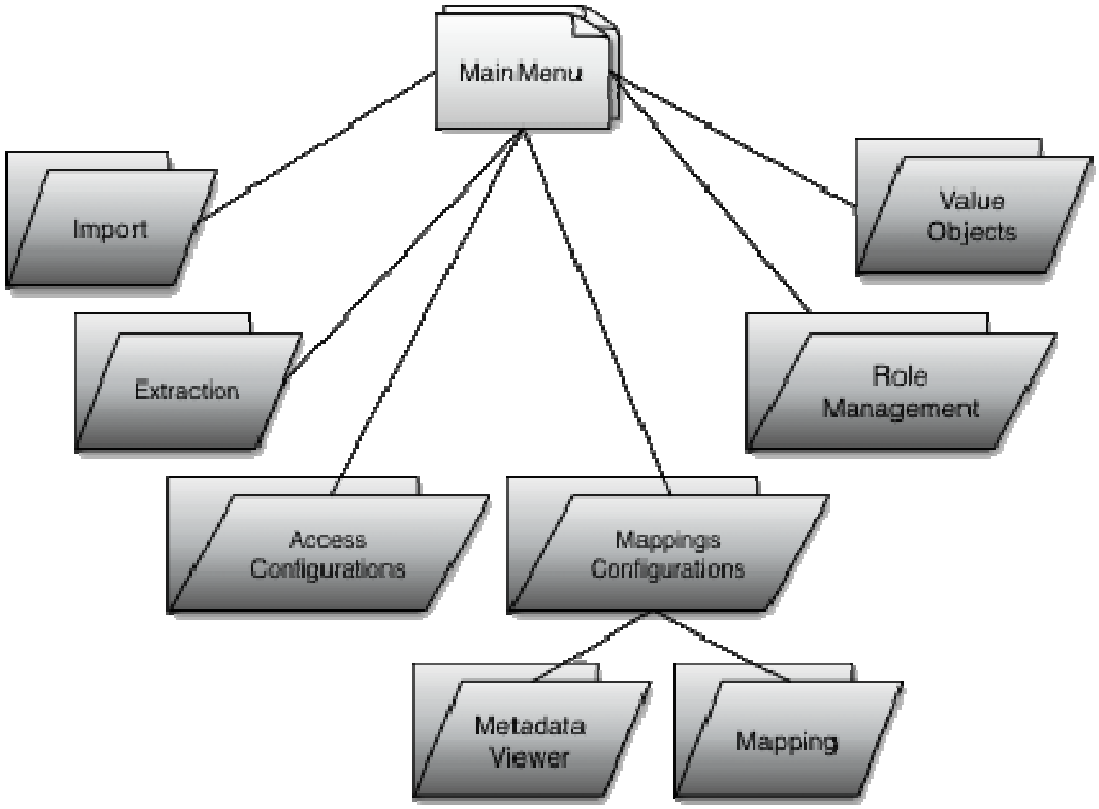


Figure 48: Overview implemented Interface

Figure 48 gives an idea of how the implemented interfaces are structured (actual class files are not represented). In the following sections a list of implemented interface components for each feature is presented.

5.3.1 Import

These are interface components to help user in the Import process.

- List Documents Mappings configurations (from Mappings Configurations)
- Upload Document form
- List Databases Mappings configurations (from Mappings Configurations)
- List Content Groups (from Mappings Configurations)

- Import Content

These interfaces were implemented following the specified requirements in Package 4 of the system in section 4.1.5.2.4.

5.3.2 Extraction

These are interface components to help user in the Extraction process.

- List Documents Mappings configurations (from Mappings Configurations)
- List Databases Mappings configurations (from Mappings Configurations)
- List Content Groups (from Mappings Configurations)
- Extract Content

These interfaces were implemented following the specified requirements in Package 5 of the system in section 4.1.5.2.5.

5.3.3 Access Configurations

These are interface components that allow users to create database access and document types configurations.

- List Documents Types configurations
- Create / Edit /Delete Document Types configurations form
- List Database Access configurations
- Create/ Edit /Delete Database Access configurations form

These interfaces were implemented following the specified requirements in Package 1 of the system in section 4.1.5.2.1.

5.3.4 Mappings Configurations

These are interfaces used to help in the mapping process. This folder includes other folders with interface files. Metadata Viewer – for interfaces that implement Metadata viewing of database and files feature; Mapping – for interfaces related for operating with mappings and mapping types.

- List Documents Mappings configurations
- List Databases Mappings configurations
- Create/Edit/Delete Mapping configurations form

- List Content Groups
- Create/Edit/Delete Content Groups form
- Present Entities Metadata
- View History
- View Logs

➢ **Metadata Viewer**

- List Entities
- Present Entity Metadata in form of a table

➢ **Mapping**

- List Mappings
- Create/Edit/Delete Mapping form
- List Mappings Types
- Create/Edit/Delete Mapping Types form

Form for creating new mapping from a mapping type needs to be built dynamically, taking into account the TypeSquare information referred in section 4.3.2.1 of the system model. This TypeSquare has property types associated with mapping types. To solve this necessity, when accessing the screen for creating a new mapping instance from a mapping type, interface was implemented so that it will check each property types dataType attribute, and by that decide what kind of component to present to the user.

Some examples of dataTypes and respective components to show:

- **Entity:** shows a combo box with a list of entities as datasource
- **Attribute:** shows a combo box with a list of attributes as datasource
- **Concept:** shows a combo box with a list of concepts as datasource
- **varchar:** shows a text input box

These interfaces were implemented following the specified requirements in Package 2 and 3 of the system in sections 4.1.5.2.2 and 4.1.5.2.3, respectively.

5.3.5 Role Management

These are interface components that allow the Administrator to manage system roles and capabilities that control the components that are available to users.

- List Content Repository Users
- List and Create / Edit /Delete Roles
- List and Create / Edit /Delete Capabilities

- Assign roles to users.

These interfaces were implemented following the specified requirements in Package 6 of the system in section 4.1.5.2.6.

5.3.6 Value Objects

Value Objects are Actionscript classes that are mapped to Java SE classes, with the same name, attributes and methods, in order to allow communication of the interface with the Business Logic layer.

5.4 Business Logic Layer

This layer is implemented in Java SE and is responsible for defining all business rules for implementing application functionalities. Figure 49 illustrates the how this layer was implemented with an overview of class file structure.

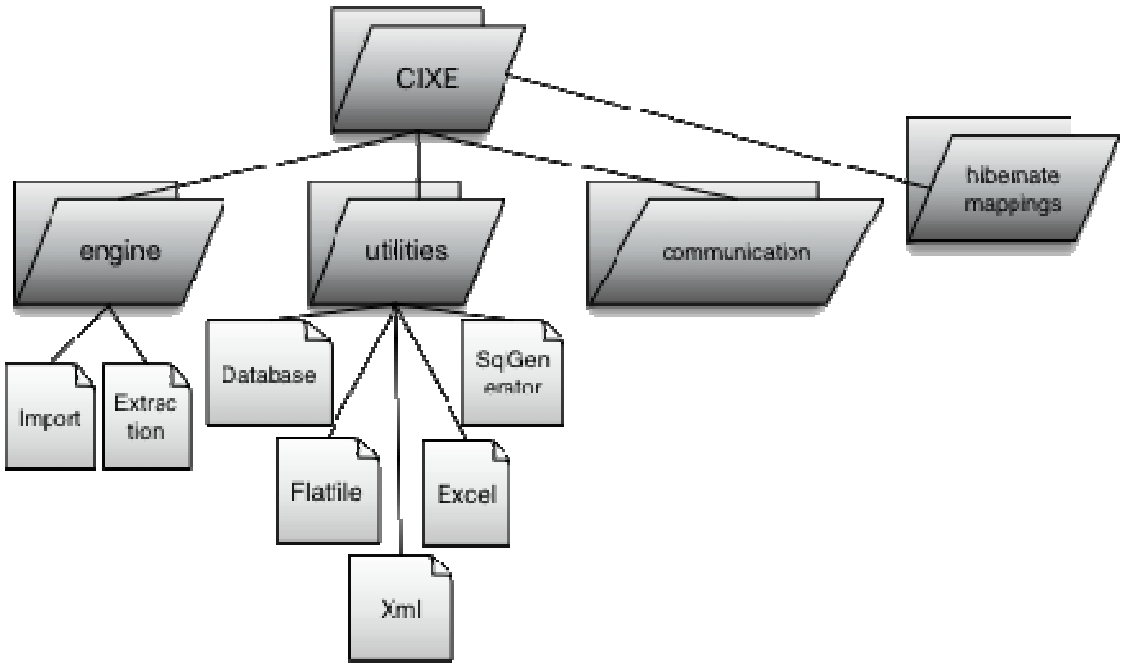


Figure 49: Overview of implemented Business logic

CIXE Business Logic Layer is organized in three main modules. These are Utilities, Communication and Engine. Hibernate mappings folder has all necessary mapping

configuration files to map not only class objects from Java to database tables and attributes but also nodes of XML documents.

5.4.1 Business Logic Communication

This module is responsible for communication between Business Logic and other layers.

As it was mentioned earlier in section 5.3.6, Flex communicates with Business Logic layer using Value Objects (actionscript classes) that are mapped to Java classes. These Java classes have associated with them related to them are stored in flex folder.

Communication and persistence with Content Repository and CIXE database layer is done with Hibernate by mapping Java classes and XML entities to database entities. XML to database mapping feature of Hibernate, makes it easier to map information from XML document entities to relational database entities. Each one of classes mapped by hibernate have an associated Data Access Object pattern [DAOP09] based class that is responsible for CRUD operations to that mapped Java class.

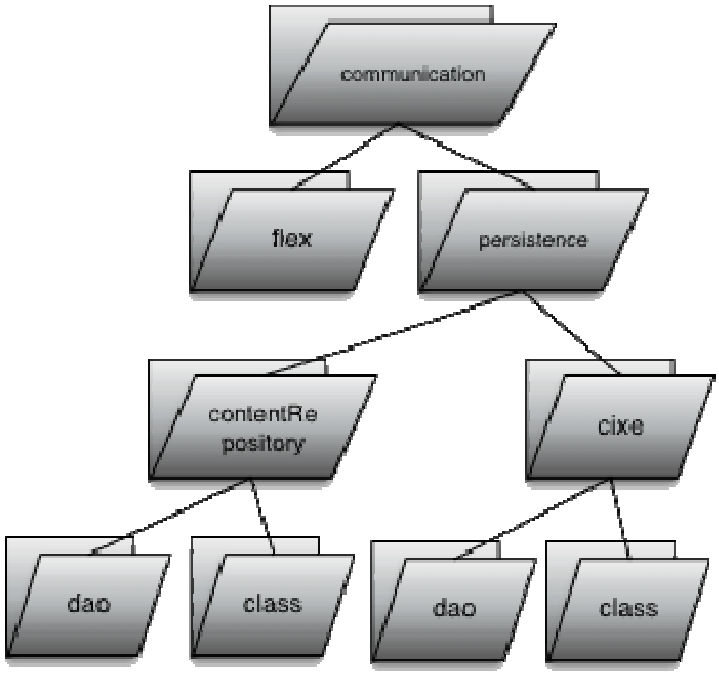


Figure 50: Overview of Business logic communication

5.4.2 Application Utilities Components

Business Logic for Content information workflow during Content Import and Extraction processes requires certain components to serve as support for handling the information transferred between Content Repository and other Objects. These utilities include functionalities, such as:

- Connecting to a document and retrieve metadata information
- Converting Content information to XML format
- Apply transformations to Content information in XML format resulting in a new XML format possibly with new metadata
- Generating Documents from a XML document
- Generating SQL statements
    - Select statements for creating queries that will get information from databases
    - DML statements for creating, updating or removing Content information from a database.

5.4.2.1 Connection

CIXE is able to connect to other objects.

Databases

Connection to other databases is made by using a JDBC connector compatible with the type of the database you are trying to connect. For the time being this is supported for Oracle and MySQL.

Documents

As for documents, these need to be uploaded to the server so that the application is able to open the document.

5.4.2.2 Metadata Extraction

After connection, CIXE is able to extract metadata information from other databases and documents that will have Content information that needs to be mapped to Content Repository.

Databases

Getting metadata information from databases requires making queries to specific system tables that have information about entities and attributes that can be saved to CIXE metadata model represented in section 4.3.1.

Oracle database metadata from tables of the user that is currently connected can be retrieved with the following statement:

Table 3: Retrieving Oracle Metadata

```
select m.table_name, m.column_name, m.data_type,
m.nullable, m.data_precision
from user_tab_cols m
```

As for MySQL the same is possible with the following statement:

Table 4: Retrieving MySQL Metadata

```
select m.table_name, m.column_name, m.data_type,
m.nullable, m.data_precision
from information_schema.columns m
```

Documents

As for documents, in order to get their metadata, they need to be parsed by the application.

Flat Files are simply parsed taking into account a separator character. These are considered Entities and the first line of information are associated attribute names.

Excel documents are parsed with a library called Apache POI [APOI09]. This is the most complete open source Excel document processor for Java. With this library we can retrieve information from Excel. When parsing Excel documents, these are considered as databases where each sheet is represented by an entity. The first line of each sheet is considered as entities' attributes.

5.4.2.3 Content Extraction

After connection, CIXE can extract Content information from other databases or documents. The Entities and Attributes required for extracting Content from them are the ones that are used in mappings as properties.

We start by determining a list of Entities and Attributes that are used in mappings and attributes used as Connectors (in case of an external mappings), including their respective Entities.

Databases

To extract Content from a database it is necessary to generate a SQL Select statement with Entities and Attributes from mappings and connectors. XSLT SQL Generation referred in

section 5.4.2.7 has a module that does this by receiving the list of Entities, Attributes and Connectors and returning the resulting SQL Select Statement.

Documents

As for documents they need to be parsed the same as referred in section 5.4.2.2 in order to extract Content information.

5.4.2.4 XML Conversion

After Content Extraction, Content information is converted to a generic format, namely XML.

Databases

After extraction from other databases, Content information is represented in a ResultSet[1] so this is converted to XML.

Documents

After extraction from documents, information is parsed and converted to XML.

The structure used for representing Content information in XML is something like this:

Table 5: Content Information XML representation example

```
<?xml version="1.0" encoding="UTF-8" ?>
<CONTENTS>
    <CONTENT>
      <ID_DIAG type="numeric">4233</ID_DIAG>
      <ID_DIAG_PARENT type="numeric">2323</ID_ DIAG_PARENT>
      <AGE_MIN type="numeric">8</AGE_MIN>
      <AGE_MAX type="numeric">35</AGE_MAX>
      <GENDER type="varchar">M</GENDER>
    ...
    </CONTENT>
    ...
</CONTENTS>
```

5.4.2.5 Mappings XSLT Transformations

Mappings have associated two transformations (one for Import and another one for Extraction) that consist on XSLT transformation source code used for transforming Content information from one metadata to another when importing or extracting.

[1] ResulSet – Java Platform SE class usually for dealing with results from queries to databases

5.4.2.5.1 Generating Transformations from templates

Transformations for mappings instances are generated from their correspondent mapping type transformations templates. This process consists in parsing transformation templates and replacing property types occurrences by the property instance values.

Take by example a mapping type with property types, identifier and value_type with property instances values of ID_DIAG and varchar, respectively, for a certain mapping.

Table 6: Transformation Template example

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="CONTENTS">

<structure>
    <xsl:for-each select="CONTENT">
        <example>
            <ID_TEMP>
                <xsl:value-of select="$(identifier)"/>
            </ID_TEMP>
            <VALUETYPE>$(value_type)</VALUETYPE>
        </example>
    </xsl:for-each>
</structure>

</xsl:template>
</xsl:stylesheet>
```

For an Import transformation like the one in Table 6, the process for generating the transformation instance from a template will parse the document for values between $(…) and change occurrences according to property_type names.

The resulting transformation is illustrated in Table 7. This transformation is ready to be applied to a XML document.

Table 7: Transformation example

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="CONTENTS">

<structure>
    <xsl:for-each select="CONTENT">
        <example>
            <ID_TEMP>
                <xsl:value-of select="ID_DIAG"/>
            </ID_TEMP>
            <VALUETYPE>varchar</VALUETYPE>
        </example>
    </xsl:for-each>
</structure>
```

```
</xsl:template>
</xsl:stylesheet>
```

5.4.2.5.2 Applying Transformations

Applying transformations to a XML formats generates a new XML document with new metadata. For example if we apply Table 7 transformation to Table 5 XML we obtain Table 8 XML.

Table 8: Transformed XML

```
<structure>
    <example>
        <ID_TEMP>
            4233
        </ID_TEMP>
        <VALUETYPE>varchar</VALUETYPE>
    </example>
    ...
</structure>
```

5.4.2.6 Document Generation

CIXE generates documents by converting from a XML format to Excel using Apache POI parser or Flat Files using a flat file parser.

5.4.2.7 XSLT SQL Generation

At its core, XSLT is a templating language. It takes XML as input, and then uses a set of templates to transform the XML into XML, HTML, or text.

XSLT SQL Generator was implemented for generating SQL select statements required when retrieving information from other databases. Also it is used for generating insert, update and delete statements scripts for when extracting Content information from the Content Repository to other databases. SQL Generator receives an XML document with parameters that then are converted to SQL statements.

Table 9: Example of Parameter XML for SQL Generator

```
<select>
    <entity name="DIAGNOSES">
        <attribute name="ID">
    </entity>
```

```
    <entity name="TRANSLATION">
        <attribute name="DESC">
    </entity>
    <join>
        <source entity="DIAGNOSES" attribute="CODE_DIAG">

         <target entity="TRANSLATION" attribute="CODE_T">
    </join>
</select>
```

After applying XSLT transformations for code generation this would result in a statement like the one in table 10.

Table 10: SQL Generator Example Result

```
select DIAGNOSES.ID, translation.DESC
from diagnoses, translation
where diagnoses.CODE_DIAG=translation.CODE_T
```

For DML statements it works the same way. With a parameterization file we get the final statement after applying transformations for code generation.

5.4.3 Engine

CIXE Engine module is responsible for the business logic necessary for controlling the workflow for the Import and Extraction process. At any time this module communicates with the other two main modules allowing it to use utilities module for handling Content information, and communication module for accessing Content Repository and CIXE data model information.

5.4.3.1 Import

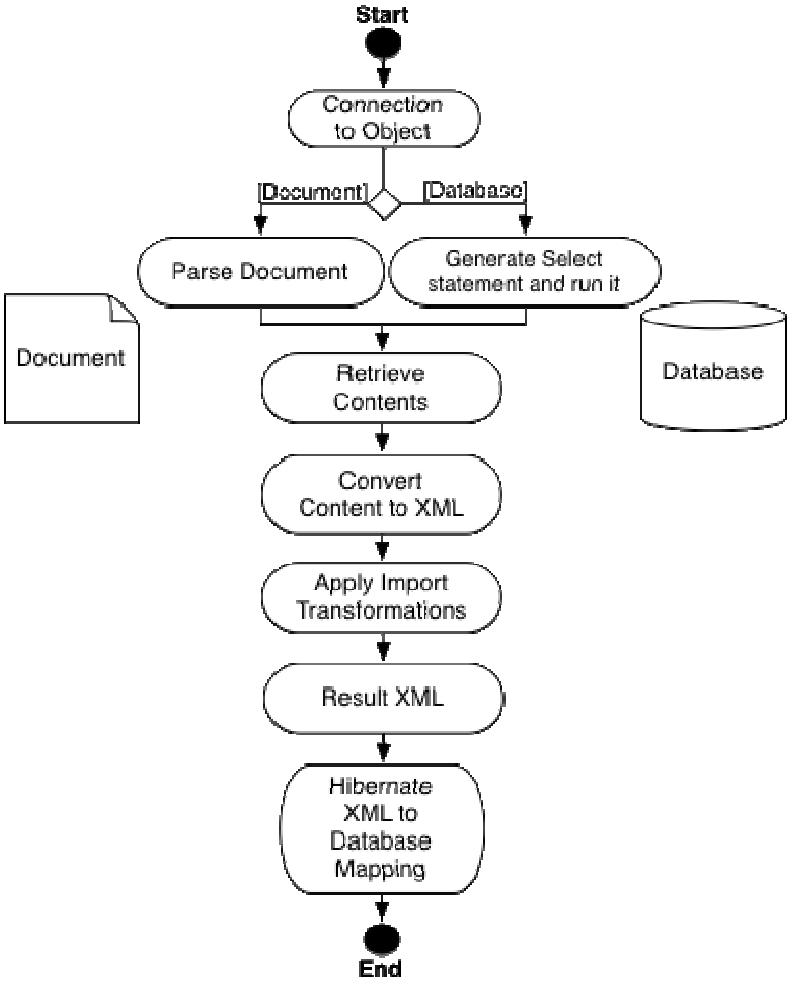


Figure 51: Import process workflow

Import process workflow is presented in Figure 51. First a connection to the Object to Import Content from is established. If the Object is a document it is parsed for retrieving the necessary information for Content Extraction, but if a database, it will generate a SQL select statement for the same effect. These parsing and SQL generations are made based on attributes and entities from the mappings defined for the Object. After this, Contents information are retrieved from the Object and converted to XML. To this XML, mappings Import transformations, related to that Object, are applied creating a new XML metadata. Finally, resulted XML will be persisted to database using Hibernate XML to Database mapping capability.

Dependencies

As it was mentioned in section 3.2.4, at the Core of the Content Repository is the basic structure of the Conceptual Network, which organizes all the existing Content. Because of this structure it is necessary to take into account, when importing Content, that Content information

mapped to Concepts are to be created first because of the dependency of relationships and descriptions.

5.4.3.2 Extraction

Extraction process workflow is presented in Figure 52.

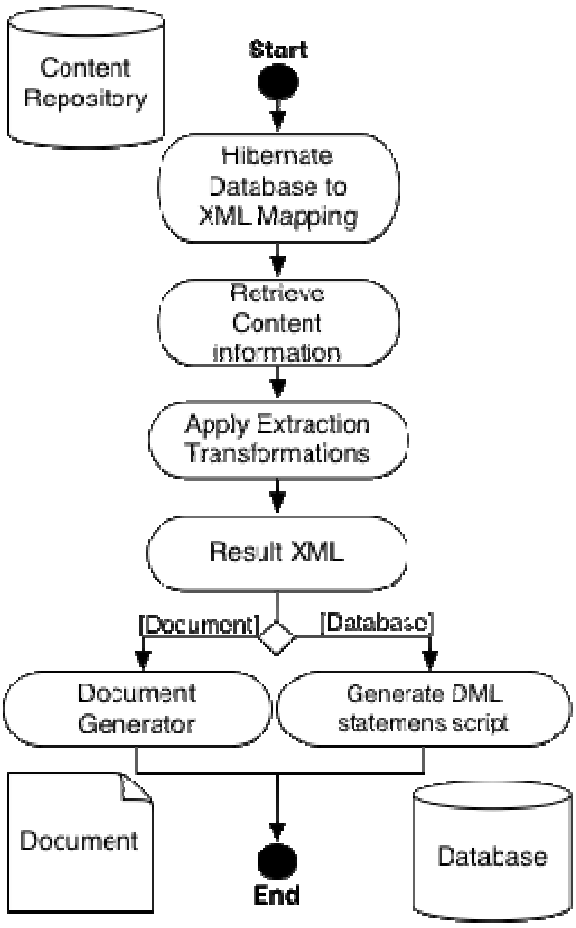


Figure 52: Extraction process workflow

First Content information is retrieved from Content Repository directly to XML using Hibernate's feature of XML to Database Mapping. This Content is represented in a metadata similar to Content Repository. To this XML, mappings Extraction transformations, related to that Object, are applied creating a new XML metadata. From this XML it is possible to generate Documents, such as CSV or Excel files, and SQL DML scripts for databases.

5.5 Database Layer

Using UML model presented in 4.3 as base for converting for the relational model, we obtain a data model as it is represented in Appendix B.

5.6 CIXE prototype Test Case

This chapter presents a Test Case made for evaluating performance of the application and also to show some screens of the application running.

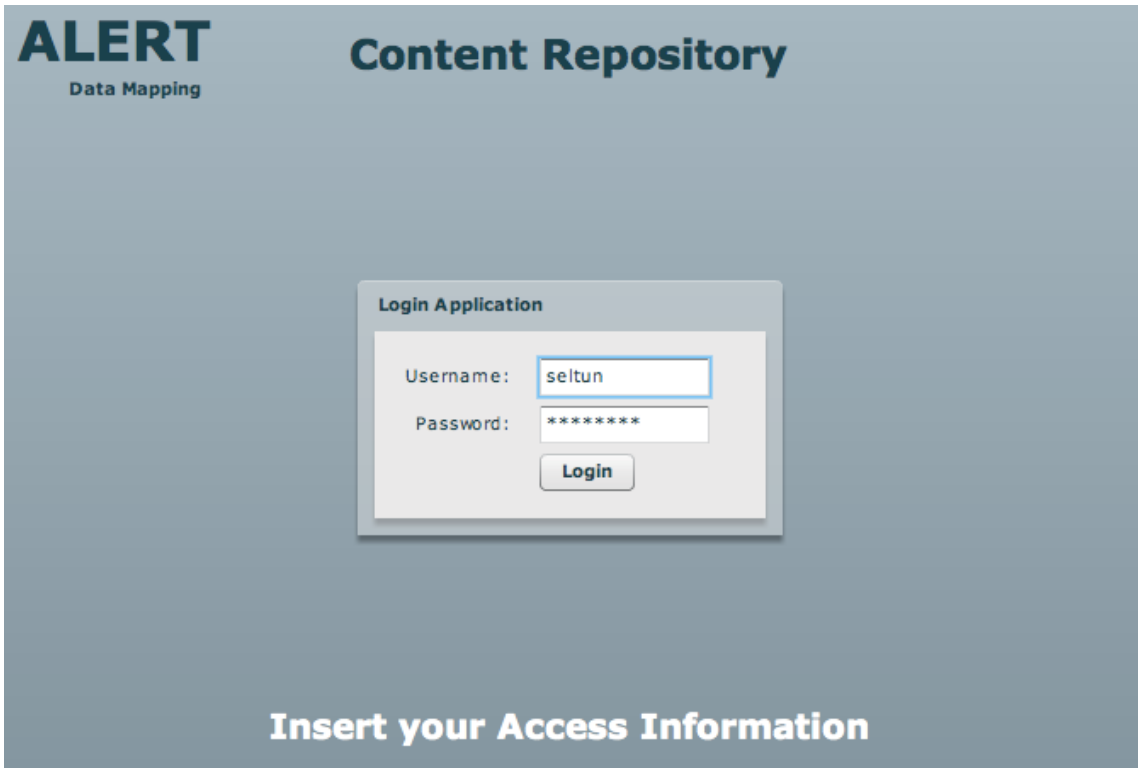


Figure 53: Login screen of the application

5.6.1 Purpose

This test has the purpose of creating two mappings configurations: one mapping configuration to a database, more specifically to a Content table for Diagnoses, and another one to an Excel document with Contents information. After these configurations, performance tests for Import Content information are executed.

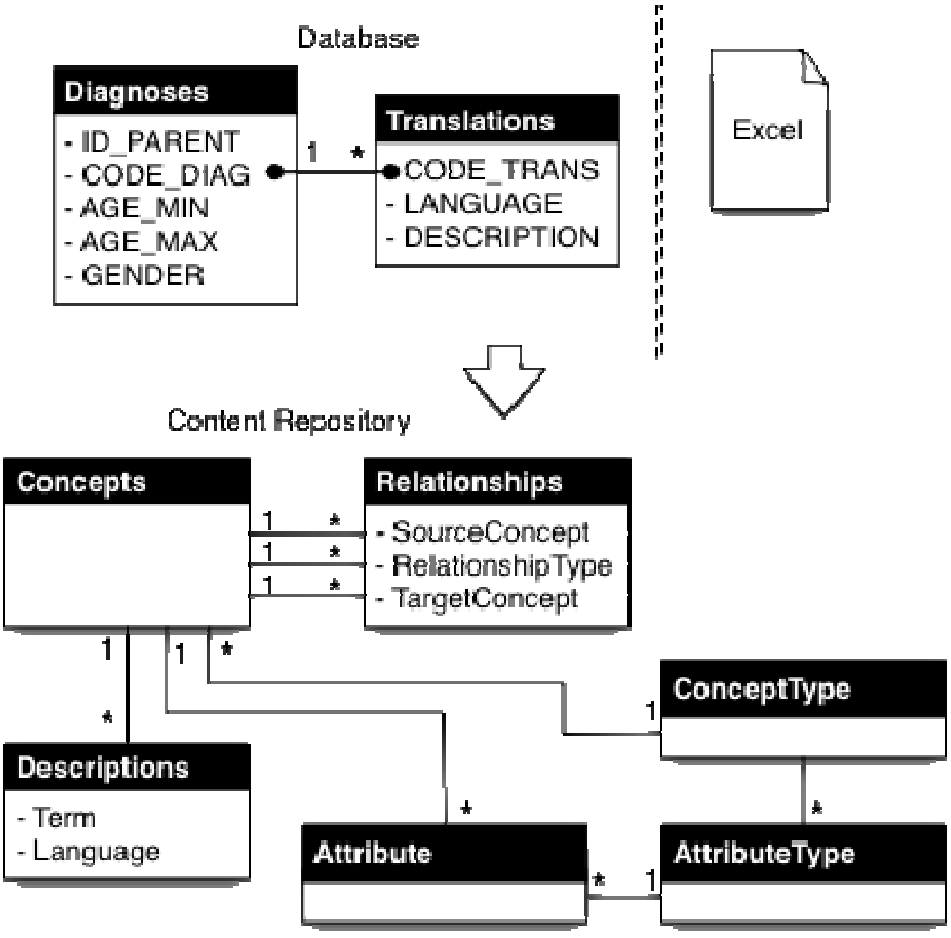


Figure 54: Different Structures

Figure 54 illustrates the difference from Content information metadata in a database and the metadata of the Content Repository. Excel document metadata is not illustrated in this picture but part of the document used is in Appendix C. Essentially this Excel document has the same structure as the Diagnoses table.

5.6.2 Database Access and Document Type Configuration

First we have to create a database access configuration to the database for this test. The application already has support for Excel document by default so there is no need to create a new Document Type.

Figure 55: Database Access Configuration Screen

5.6.3 Mapping Configurations

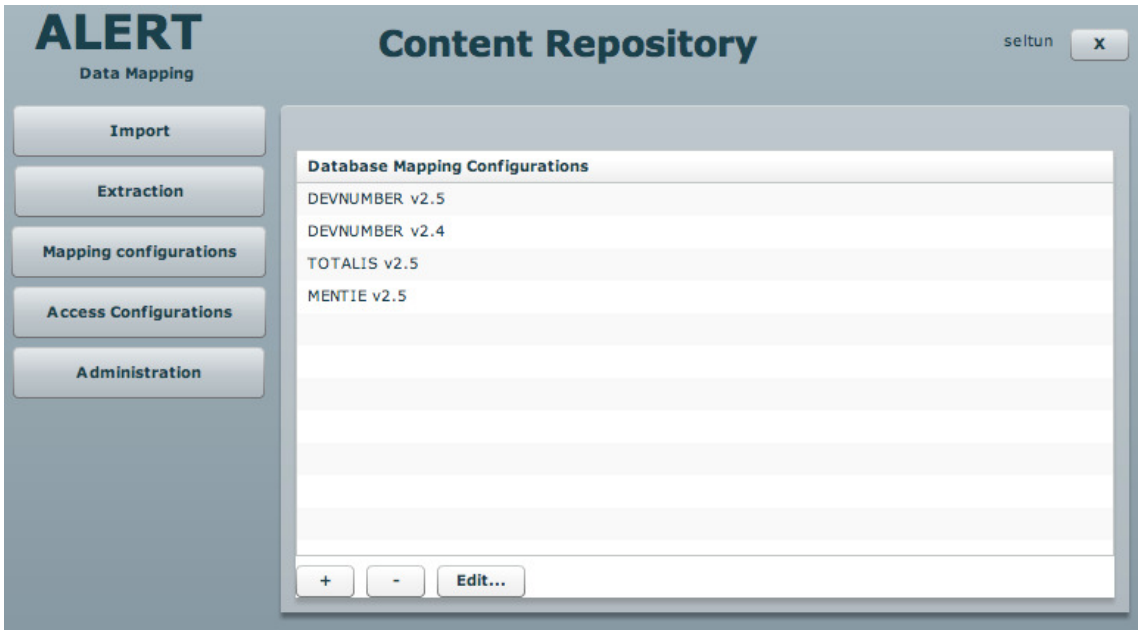


Figure 56: Mapping Configurations Screen

In Figure 56 we have a list of existent configuration mappings for databases and the possibility for creating new ones. The same happens for documents.

5.6.3.1 Mapping Types

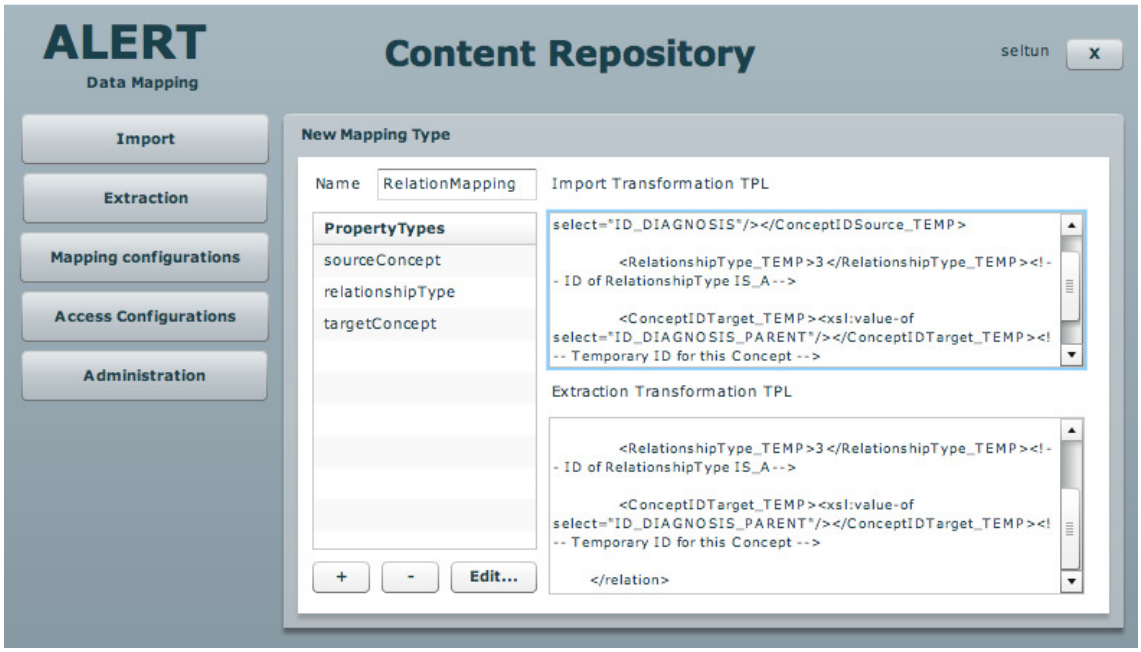


Figure 57: New Mapping Types Screen

This screen allows creating new mapping types for extending new transformations capabilities to the system. But by default application already has implemented mapping types that allow transforming information to metadata representation for Concept, Relationship, Attributes and Descriptions.

5.6.3.2 Mapping

The screen presented in this section shows metadata information from the database that is being mapped to Content Repository. This works the same way with documents. Metadata visual representation helps to better understand what attributes and entities are available for mapping.

Mapping types required for this case are:

- MappingContent (mapping to Diagnoses Entity as a concept)
- RelationMapping (mapping to parent and child relationship between id and id_parent as an IS_A relationship)
- AttributeMapping (mapping AGE_MIN, AGE_MAX and GENDER as attributes)
- DecriptionMapping (external mapping translations as relationships)

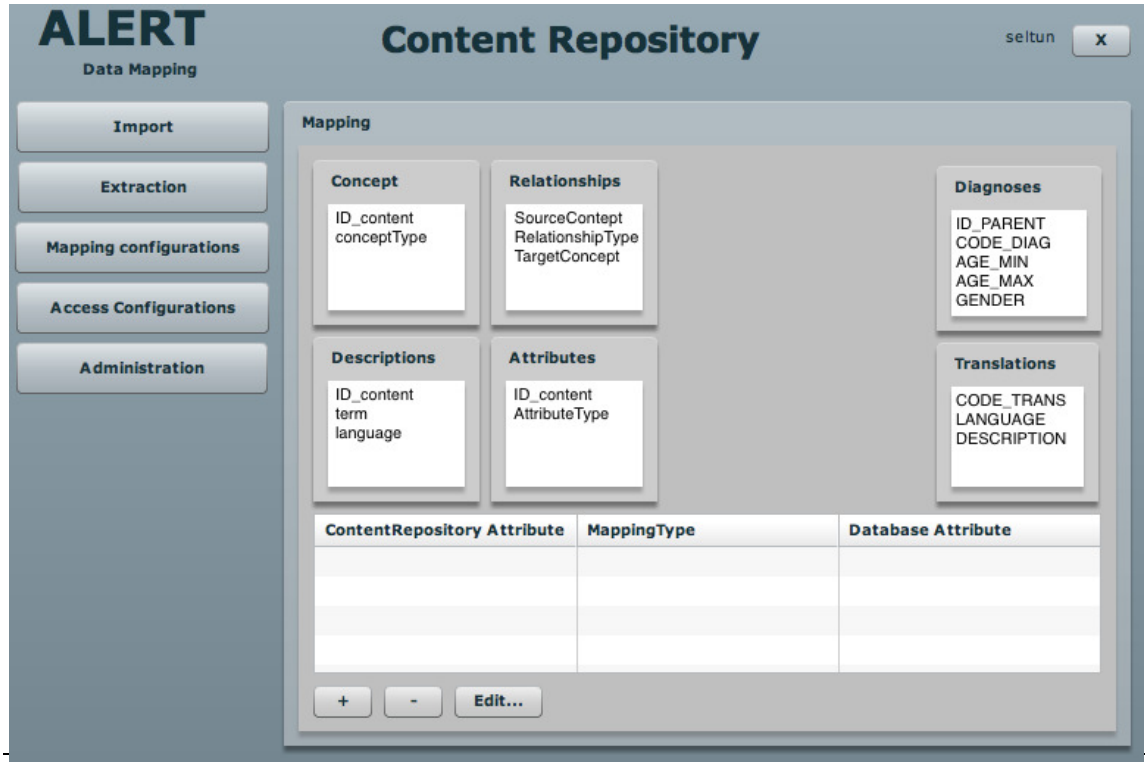


Figure 58: Mapping Screen

5.6.4 Performance Test Results

Performance Tests are important to evaluate average performance of products and avoid higher development costs. Usually, when a project fails performance tests, it's necessary to optimize and some times redefine architecture allowing better performance.

5.6.4.1 Import

After creating required mapping configurations we are ready to run Import tests using them as parameterization. Table 11 has the results for the import test.

Table 11: Import process – Results

	Number of Diagnoses	Time Taken	CPU Usage	Memory Usage	Performance
Excel Document	**21670**	**109 seconds**	**33%~42%**	**310 MBytes**	**Good**
Database	**29 734**	**112 seconds**	**24%~32%**	**300 MBytes**	**Good**

As we can see in the table, application has Good performance values for both excel documents and database Import. It has a good average CPU and Memory usage during the process. Time taken for the process is acceptable taking into account the number of Contents we

are importing. Although the document had almost minus 8100 than the database it took almost the same time for the process to end. This could be because parsing time for document being more exhaustive than querying time for database.

Note: For every one of the Contents in this Import process was created 1 record for Concepts, 1 for relationships and 3 records for attributes. As for descriptions, database had 7 different language translations and Excel document had only 1.

5.6.4.2 Extraction

Extraction process was tested after importing 29 734 Diagnoses from a database and with the purpose of extracting all those diagnoses. Mapping configurations were done earlier so we are ready to run extraction tests using them as parameterization. Table 12 has the results for the extraction test.

Table 12: Extraction process – Results

	Time Taken	CPU Usage	Memory Usage	Performance
Excel Document	**148 seconds**	**21%~39%**	**310 MBytes**	**Good**
SQL Scripts	**162 seconds**	**28%~35%**	**330 MBytes**	**Good**

As we can see in the table, application has Good performance values for extracting Contents to excel documents and SQL DML scripts. CPU, Memory Usage and Time taken for the process is acceptable taking into account the number of Contents we are extracting. SQL Scripts took a little more time than the excel documents. This could be because creating an Excel document and parsing it to write the values is a more direct extraction process than first creating XML parameterizations for XSLT SQL Generator and then applying transformations for code generation, necessary for this type of extraction.

5.7 Summary

CIXE prototype worked as proof of concept for the proposed architecture and model defined for the tool, allowing Content diffusion between ALERT® clinical Applications.

With this solution, users can specify database access configurations or document types and then create mappings configurations and specify mappings and transformations between the Content Repository's metadata and other databases or documents metadata. After having these configurations, the prototype can import and extract Content information based on those

mappings and transformations. This is possible by having a system data model, serving almost as an metadata mapping repository, and by representing Content in XML, during the Import and using XSLT for applying those mappings and transformations.

Chapter 6

Conclusions

6.1 Objectives' Satisfaction

The use of a conceptual network as a central repository of content, limits the dispersion of sources of content and at the same time allows representing semantic relations between the concepts.

A solution was proposed to solve the issue of Content diffusion, which main purpose is to establish mechanisms for:

- Importing Contents from existing databases and documents like excel files to the Content Repository
- Extracting and packaging of Content associated with ALERT® clinical applications or for creating documents with listing of Contents to present to customers, allowing them to choose the ones they want in their installation

The proposed solution is a web application that has an underlying Configurations module, with log and history, responsible for the management of metadata and mappings and transformations necessary for allowing interoperability between Content Repository and other databases or documents metadata.

To implement the proposed solution, a prototype was developed where users can specify database access configurations or document types, and then create mappings configurations and specify mappings and transformations between the Content Repository's metadata and other databases or documents metadata. After having these configurations, the prototype can import and extract Content information based on those mappings and transformations. This is possible by having a system data model, serving as an metadata and mapping repository for other databases and documents, and by representing Content in XML, during the Import and XSLT for applying those mappings and transformations.

Given the expectations of the project, objectives were met in the analysis, specification and implementation of a prototype of the solution.

6.2 Future Work

The impact of this work is expected gradually in the daily work of different actors in Content management, once the Content Repository starts to be used and benefit from the application developed to allow import and extraction of Contents.

For next steps to follow, the company's objective is progressively import the entire contents of their current primary Content database, into the repository, then extracted to SQL scripts and versioned for Data Quality Control.

There is always space for improvements. One of them is to improve the logging process for allowing even more detail and filtering information presented to the user. Other improvements would be investing in some important aspects of the application's interface. Improvements to the interface in general would help users to better understand what steps they need to do for executing an action. The area that needs more improving is Adjust the interface for mapping specification to be user-friendlier through the action of drag and drop between attributes from one point to another of the Import and Extraction process. Because ALERT wants to use this tool in other subsidiaries, it will be important to adapt the design of the interface to the look and feel of the ALERT® products, giving users some familiarity to the interface.

References

[SMCT09] IHTSDO, International Health Terminology Standards Development Organisation, SNOMED-CT, March 2009. http://www.ihtsdo.org/snomed-ct/

[ALE09a] ALERT Life Sciences Computing, S.A. ALERT Life Sciences Computing — Company presentation / March 2009, March 2009.

[FEU09a] FEUP, Proposta de projecto/dissertação: Difusão de conteúdos nas aplicações clínicas ALERT®, 2009.

[DJP05] Digital Repositories JISC Briefing Paper, 2005

[SCS08] Guide to Health Informatics 2nd Edition, SNOMED-CT Core Structure, Enrico Coiera, 2001.

[NLM08] National Institute of Health, National Library of Medicine, March 2009. http://www.nlm.nih.gov/

[UMLS08] National Institute of Health, National Library of Medicine, Unified Medical Language System, March 2009. http://www.nlm.nih.gov/research/umls/

[CAP09] College of American Pathologists Foundation, March 2009, http://www.cap.org/apps/cap.portal

[WCFE04] Wyss, Catharine M., George H.L. Fletcher, Fulya Erdinc, and Jeremy T. Engle, “MIQIS: Modular Integration of Queryable Information Systems”, 2004.

[SMUD05] Bilke, Alexander and Felix Naumann, “Schema Matching using Duplicates”, 2005.

[UDMR95] Jain, M., A. Mendhekar, and D. Van Gucht, “A Uniform Data Model for Relational Data and Meta-Data Query Processing”, 1995.

[XSLCG05] Code Generation Information for the Pragmatic Engineer, Code Generation Network, 2008. http://www.codegeneration.net/

[XML09] W3C - World Wide Web Consortium. W3C XML Specification, April 2009. http://www.w3.org/XML/.

[XSLT09] W3C - World Wide Web Consortium. W3C XSLT Specification, April 2009. http://www.w3.org/TR/xslt20/

[XPH09] W3C - World Wide Web Consortium. W3C XPATH Specification, April 2009. http://www.w3.org/TR/xpath/

[PSDEV09] Real solutions for Oracle Developers, allroundautomations, PL/SQL Developer, 2009. http://www.allroundautomations.com/plsqldev.html

[ECLP09] The Eclipse Foundation, Eclipse, 2009, http://www.eclipse.org/

[SVN09] Garrett Rooney; Practical Subversion; Apress; (1st Edition, paperback), 2004

[ORAC09] Oracle Corporation, Oracle Database 10.2, 2009, http://www.oracle.com/database

[FLS09] Gay, Jonathan. "The History of Flash", Adobe, 2009. http://www.adobe.com/macromedia/events/john_gay/

[FLEX09] Adobe Systems Incorporated, Adobe Flex 3, June 2009. http://www.adobe.com/products/flex/

[SAXN09] XSLT AND XQUERY PROCESSING, SAXONICA, Saxon, March 2009, http://saxon.sourceforge.net/

[HBNT09] Red Hat, Red Hat Middleware, JBoss, Hibernate, March 2009, https://www.hibernate.org/

[TCAT09] The Apache Software Foundation, Apache Tomcat, April 2009. http://tomcat.apache.org/

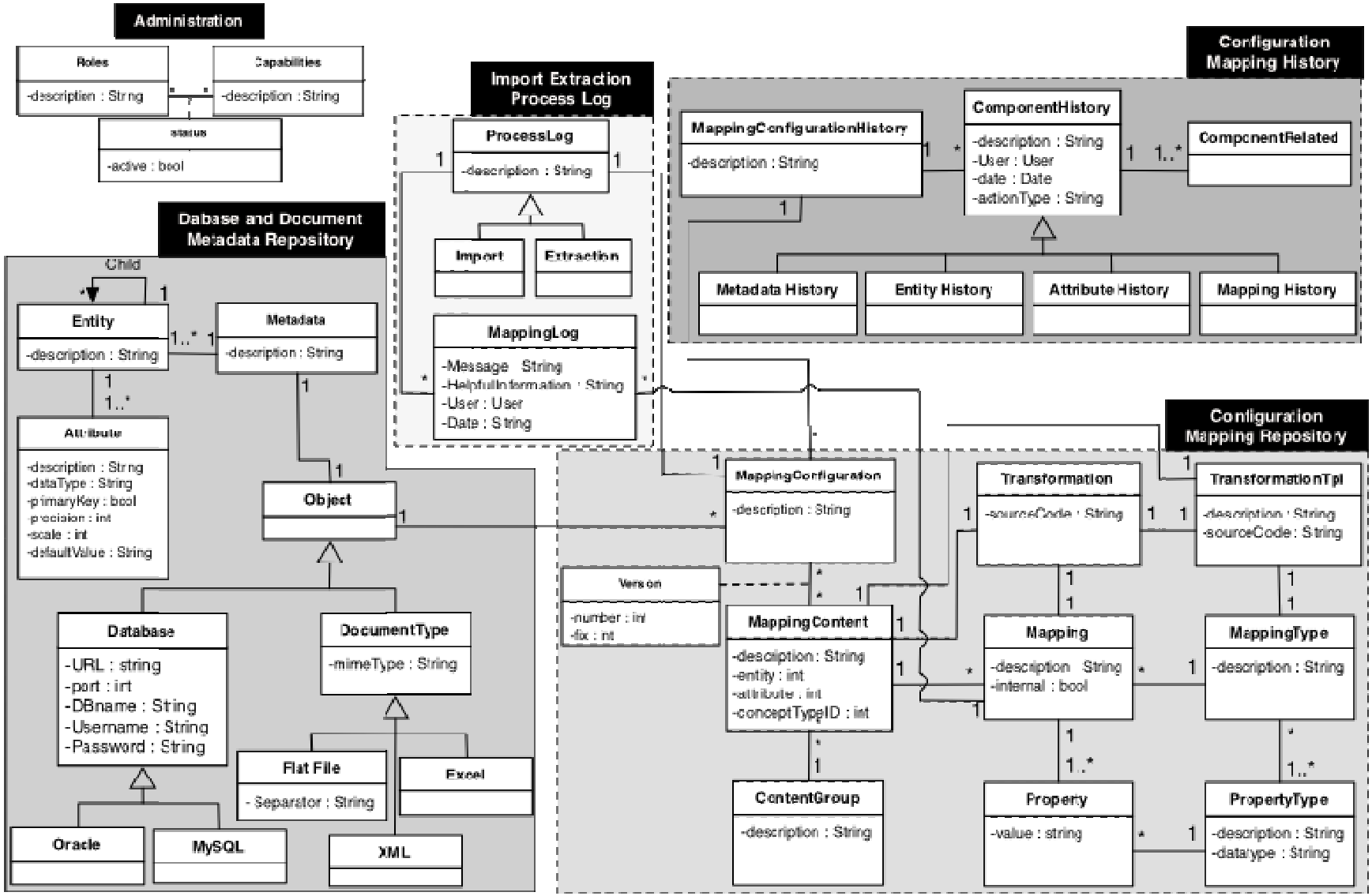
[DAOP09] Sun Microsystems, Inc. Core J2EE Patterns – Data Access Object, 2009. http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html

[TOP09] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. Pattern–Oriented Software Architecture – A System of Patterns. Wiley and Sons Ltd., 1996.

[APOI09] The Apache Software Foundation, The Apache POI Project, April 2009, http://poi.apache.org

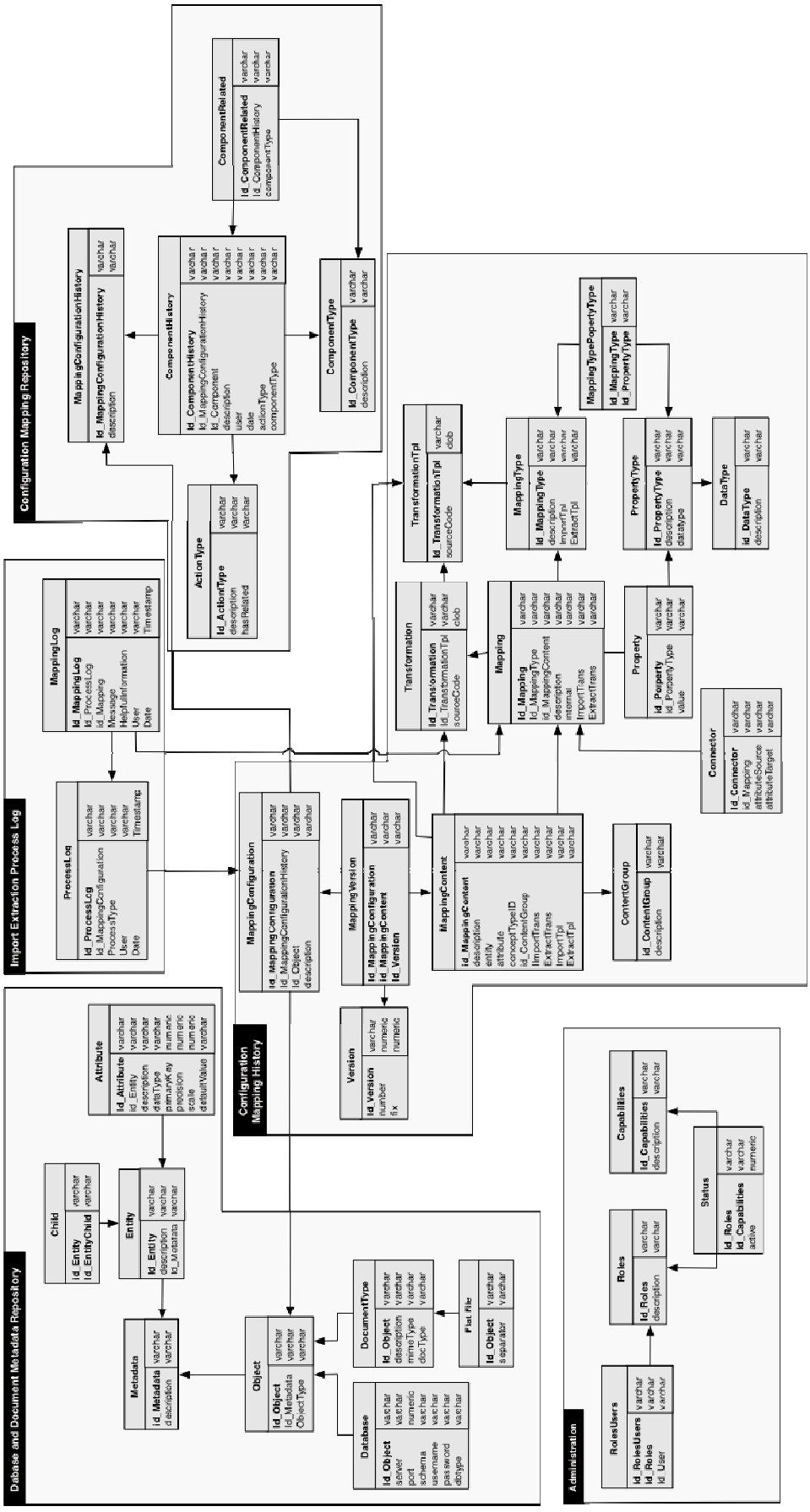
Appendix A

CIXE UML Model


Administration
Roles
-description : String
Capabilities
-description : String
status
-active : bool
Import Extraction Process Log
ProcessLog
-description : String
Import
Extraction
MappingLog
-Message String
-HelpfulInformation : String
-User : User
-Date : String
Configuration Mapping History
MappingConfigurationHistory
-description : String
ComponentHistory
-description : String
-User : User
-date : Date
-actionType : String
CompcnentRelated
Metadata History
Entity History
Attribute History
Mapping History
Dabase and Document Metadata Repository
Child
Entity
-description : String
Metadata
-description : String
Attribute
-description : String
-dataType : String
-primaryKey : bool
-precision : int
-scale : int
-defaultValue : String
Object
Database
-URL : string
-port : int
-DBname : String
-Username : String
-Password : String
DocumentType
-mimeType : String
Flat File
- Separator : String
Excel
Oracle
MySQL
XML
Configuration Mapping Repository
MappingConfiguration
-description : String
Version
-number : int
-fix : int
MappingContent
-description: String
-entity : int
-attribute : int
-conceptTypeID : int
ContentGroup
-description : String
Transformation
-sourceCode : String
TransformationTpl
-description : String
-sourceCode: String
Mapping
-description String
-internal : bool
MappingType
-description : String
Propery
-value : string
PropertyType
-description : String
-datatype : String

Appendix B

CIXE Relational Model


Configuration Mapping Repository
MappingConfigurationHistory
Id_MappingConfigurationHistory
description
ComponentHistory
Id_ComponentHistory
Id_MappingConfigurationItemHistory
Id_Component
description
user
date
actionType
componentType
ComponentRelated
Id_ComponentRelated
Id_ComponentHistory
componentType
ComponentType
Id_ComponentType
description
ActionType
Id_ActionType
description
hasRelated
TransformationTpl
Id_TransformationTpl
sourceCode
MappingTypePropertyType
Id_MappingType
Id_PropertyType
MappingType
Id_MappingType
description
ImportTpl
ExtractTpl
PropertyType
Id_PropertyType
description
datatype
DataType
Id_DataType
description
Import Extraction Process Log
MappingLog
Id_MappingLog
Id_ProcessLog
id_Mapping
Message
HelpfulInformation
User
Date
ProcessLog
Id_ProcessLog
Id_MappingConfiguration
ProcessType
User
Date
Transformation
Id_Transformation
Id_TransformationTpl
sourceCode
Mapping
Id_Mapping
Id_MappingType
id_MappingContent
description
internal
ImportTrans
ExtractTrans
Property
Id_Property
Id_PropertyType
value
Connector
Id_Connector
id_Mapping
attributeSource
attributeTarget
Configuration Mapping History
MappingConfiguration
Id_MappingConfiguration
Id_MappingConfigurationHistory
id_Object
description
MappingVersion
Id_MappingConfiguration
Id_MappingContent
Id_Version
MappingContent
Id_MappingContent
description
entity
attribute
conceptTypeID
id_ContentGroup
ImportTrans
ExtractTrans
ImportTpl
ExtractTpl
ContentGroup
Id_ContentGroup
description
Version
Id_Version
number
fix
Dabase and Document Metadata Repository
Attribute
Id_Attribute
id_Entity
description
dataType
primaryKey
precision
scale
defaultValue
Child
Id_Entity
Id_EntityChild
Entity
Id_Entity
description
Id_Metadata
Metadata
Id_Metadata
description
Object
Id_Object
Id_Metadata
ObjectType
DocumentType
Id_Object
description
mimeType
docType
Flat_File
Id_Object
separator
Database
Id_Object
server
port
schema
username
password
dbType
Administration
Capabilities
Id_Capabilities
description
Roles
Id_Roles
description
Status
Id_Roles
Id_Capabilities
active
RolesUsers
Id_RolesUsers
Id_Roles
Id_User

Appendix C

Sample Excel Document

DESC_TRANSLATION	CODE_ICD	CODE_ICD_PARENT	ID_DIAGNOSIS
PROCEDURES	00-99.99		409187
Procedures And Interventions, Not Elsewhere Classified	00	00-99.99	400248
Pharmaceuticals	00.1	00	400253
Implantation of chemotherapeutic agent	00.10	00.1	400254
Infusion of drotrecogin alfa (activated)	00.11	00.1	400255
Administration of inhaled nitric oxide	00.12	00.1	400256
Injection or infusion of nesiritide	00.13	00.1	400257
Injection or infusion of oxazolidinone class of antibiotics	00.14	00.1	400258
High-dose infusion interleukin-2 [IL-2]	00.15	00.1	400311
Pressurized treatment of venous bypass graft [conduit] with pharmaceutical substance	00.16	00.1	416194
Infusion of vasopressor agent	00.17	00.1	416195
Infusion of immunosuppressive antibody therapy	00.18	00.1	416286
	...		
Personal history of monoclonal drug therapy	V87.42	V87.4	479998
Personal history of antineoplastic chemotherapy	V87.41	V87.4	480113
Personal history of other drug therapy	V87.49	V87.4	480334
Contact with and (suspected) exposure to hazardous metals	V87.0	V87	480195
Contact with and (suspected) exposure to arsenic	V87.01	V87.0	479867
Contact with and (suspected) exposure to other hazardous metals	V87.09	V87.0	480138
Contact with and (suspected) exposure to other potentially hazardous substances	V87.3	V87	480225
Contact with and (suspected) exposure to other potentially hazardous substances	V87.39	V87.3	479822
Exposure to mold	V87.31	V87.3	480472