Reference Requirements Documents Manager

Denise Sofia Tavares Pedro
denisesofiatp@gmail.com
denise.pedro@tecnico.ulisboa.pt

1. INTRODUCTION

The stabilization of the knowledge in some domains motivates the creation of reference documents with requirements and good practices on those domains. An example are the reference requirements documents, fundamental to the behavior of a records management system, which contain policies and processes that guarantee a correct management of documents by organizations. Nowadays, not only one seeks to follow the guidelines presented in these documents, but also there is a search of solutions and/or components that already incorporate them. However these documents were developed by specialists in the field of records management and are essentially seek by system developers that, in most cases, are not familiar with records management concerns. This way emerges a need of creating a means to allow developers to take better advantage of the information present in these documents. The objective of this Project is to design a system that, supporting the storage of the information available in these documents, allows for an easier navigation and centered to the needs of each stakeholder. Also explore the use of goal-oriented requirements engineering techniques allow for a better understanding of the requirements present in the documents by abstracting general objectives.

2. Goal-Oriented Requirements Engineering

Requirements Engineering can be defined as a set of activities concerned with identifying and communicating the purpose of a software system, and the context in which it will run. It acts as the bridge between the real world needs of users, customers, and other constituencies affected by a software system, and the capabilities and opportunities afforded by software technologies. RE should be the way to answer the "**what, why and how**" questions. Why is some system needed? What is the purpose of this system and what unleashed is need. What features does this system need to have? What features are needed in order to satisfy the said context in which this system will run. Finally, how this said system is to be constructed. Goal Oriented Requirements Engineering makes use of goals for requirements elicitation, elaboration, structuring, specification, analysis, negotiation, documentation and evolution.

In order to understand the Goal-Oriented RE it is important to define what a *goal* is in this context. Requirements result from a set of desired specifications that a system must have and goals are abstractions of those specifications. A goal in the context of a software system is an objective that must be achieved by the system [1] and may be formulated at different levels of abstraction.

Goals are important in the RE process, and there are several reasons for it, some of them are presented next. First, they can help in achieving requirements completeness and relevance:

- **Completeness -** The requirements specification is complete with respect to a set of goals if all the goals can be proved to be achieved from the specification and the properties known about the domain considered; requirements are complete if they are sufficient to establish the goal they are refining.
- **Relevance -** A requirement is pertinent with respect to a set of goals in the domain considered if its specification is used in the proof of at least one goal.

There have been introduced several different approaches using the concept of goals as part of a Requirements Engineering technique. With GORE approaches it is possible to represent one or more stakeholder needs (goals), assign them to an agent (stakeholder or system), and relate it to other goals, frequently describing how can a goal be achieved [2]. Normally there are present in these approaches activities such as goal elicitation, goal refinement and analysis, and assignment of goals and agents relationships. There are four main GORE approaches: NFR Framework, i*/Tropos and GBRAM, and the KAOS approach which is the approach explored in this project.

3. KAOS

Standing for Knowledge Acquisition in Automated Specification (according to Dardenne and Lamsweerde) [3] or Keep All Objects Satisfied, this framework derives goal refinements, operationalizations, conflict management and risk analysis by reasoning (in a semi-formal or formal way) about behavioral goals; semi-formal when modeling and structuring goals, qualitative to select a goal among a set of alternatives and formal when in need of a more accurate reasoning. KAOS language combines semantic nets for conceptual modeling of goals, assumptions, agents, objects, and operations in the system, and linear-time temporal logic for specification of goals and objects, as well as state-base specifications for operations.

Summarizing, the KAOS approach thrives to:

- Focus on Goal elaboration:
  - Define initial set of high level goals and objects they refer to;
- Define initial set of agents and actions they are capable of.
- Then iteratively:
  - Refine goals using AND/OR decomposition links;
  - Identify obstacles to goals, and goals conflicts;
  - Operationalize goals into constraints (or software requirements) that can be assigned to individual agents;
  - Refine and Formalize definitions of objects and actions.

A KAOS specification is a collection of the following core models:

- Goal model and Responsibility model – representation of goals and assignment to agents;
- Object model – UML model derived from formal specifications of goals;
- Operation model – definition of various services to be provided by software agents.

4. Records Management Standards - MoReq2010 specification

MoReq2010, short for Modular Requirements for Records Systems**,** is a model that presents a set of requirements for reference, guidance and normalization [4]. A Records Management System must (totally or partially) comply with this model to manage records in an efficient manner and can then be denominated of MCRS (Moreq2010 Compliant Record System). To get certified according to MoReq2010 a system must fulfill all of the functional requirements specified in the standard.

As mentioned in [3] this specification thrives to avoid a "one size fits all" approach to implement a records management solution. Instead it defines a common set of core services that are shared by many different types of records systems, but which are also modular and flexible, allowing them to be incorporated into specialized and dedicated applications that might not previously have been acknowledged as records systems.

Created by the DLM-Forum community, its first version was published in 2001, then after reviews and evaluation MoReq2 was published in 2008 and in 2010 the most updated version was published, being this last update (MoReq2010) the document of interest in the scope of this work and the specification chosen to test the proposed solution.

MoReq2010 presents a new structure and content when compared to its predecessors, having independent modules, each likely to evolve independently of the other [3]. The document is divided into chapters, beginning by presenting "Important Information", the "Context," the "Purpose" and "Key Concepts". After the introductory chapters the functional requirements are then presented (eleven chapters) followed by nonfunctional requirements (fifteen chapters). Finally a number of series of requirements are shown, each with one or more modules. At least one of the modules in each series has to be fulfilled in order to make a system recognized as MCRS. Next table presents the description of each of the MoReq2010's chapters with functional requirements.

.

Chapter	**Description**
2. System Services	Expresses features that are common to all MoReq services; presents a choice of one of the modules from 100 Series - Types of interface
3.User and Group Service	Expresses the requirements related to users and user groups management
4. Model Role Service	Expresses the features related to how users are authorized to execute a function of the MCRS
5. Classification Service	Aggregates requirements related to the classification scheme, and the classification of records; presents the possibility of choosing one of the modules in the 200 series - Classification Series. It is related to chapters 6. Record Service and 8. Disposal Scheduling Service
6. Record Service	Aggregates requirements related to the aggregation of records, including hierarchies of aggregations, and its relations with the processes / business activities and classifications;
7. Model Metadata Service	Describes functionalities that facilitate the interoperability between different systems. The aim is to propose the implementation of a model which can be recognized by any MCRS, in order to export and import documents and their metadata easily
8. Disposal Scheduling Service	Aggregates requirements that define the functionalities for managing the life cycle of records in a MCRS
9. Disposal Holding Service	Aggregates requirements that define the features related to disposal hold, from the imposition of the suspension, the

	impact of the suspension and until its termination
10. Searching and Reporting Service	Aggregates requirements that define the features related to the searching of objects in a MCRS, expressing different methods, different ways of reporting, as well as features related to the generation and presentation of reports
11. Exporting Service	Aggregates requirements that define the features related to exporting content, metadata, historical events and access controls

Table 1 – MoReq2010 main important chapters with functional requirements

5. Problem Analysis

5.1 Black Box Analysis

The proposed solution can be conceptualized as a system, presented to users as a web application, where the documents in consideration should be available in a form that allows them to perform some pre-defined actions. These actions can be thought as functionalities available in the application interface, allowing a user to see and manipulate data from the reference documents. The most relevant functionalities of the application are presented in the next section with the help of use cases.

5.1.1 Proposed Actors and Use Cases

There are three types of actors which are also the expected users of the system: the **Administrator, Expert User** and the **Professional User**. The Administrator is in charge of the uploading of documents to the system, as well as any management action needed. Expert User is an actor capable of performing actions related to the management of associations between documents as well as relationships in a set of requirements. Also this user will be capable of creating possible views of these associations and relationships. This user must have a broad knowledge of records management in order to understand what correlations might exist between requirements; it also must have knowledge of Goal Oriented Requirements Engineering techniques to ensure that the created views comply with the used method. The Professional User, inheriting from the Expert User, will explore, manage, and possibly extract the created views. Proposed use cases are presented in figure 2 use case diagram.

5.2 White Box Analysis

The conceptualization of the system as a black box offers an abstraction to focus on its main functionality. A white box analysis will provide an insight on how this functionality is achieved. The next sections present the structure of the repository of the system and its different layers and also the exploration of KAOS goal models with MoReq2010 requirements.

5.2.1 Repository Structure

After analysis of the expected structure of the documents in general, in terms of pages, sections, paragraphs, among others that should be stored in a repository, the first step was to create a domain model for a database to support the storage of the structure. The Domain Model is presented in figure 3.

A Document can be any of the five discussed documents in the form of a PDF with several pages. These documents have one structure, which is unique for each one of them, despite the possible similarities in its components. The Structure is a set of sections, each of the later having Blocks. The blocks contain all of the components that can be found inside any section of the documents.

From this model, one can identify layers for abstraction and better understanding of the proposed system. First, a **Structure Layer** concerning precisely the above presented model of a repository to store the documents.

A user of the system can create certain associations between parts of different documents or inside the same document. The layer where all the services to interact with the documents' contents are made available to a user can be denominated **Business Layer**. The associations can be made combining components or objects, from the structure layer, where the documents are considered as a whole. The **Objects Layer**, containing the lowest level of granularity, is where single objects are considered. For example:

- Images;
- Texts that can be part of phrases, titles, tables or images subtitles;
- Table elements, etc.

The Components Layer, with an upper level of granularity is where compositions of objects from the objects layer are considered. Examples are:

- Sentences;
- Paragraph;
- Image and respective Subtitle, which is a Image + Text composition;
- Sections, etc.

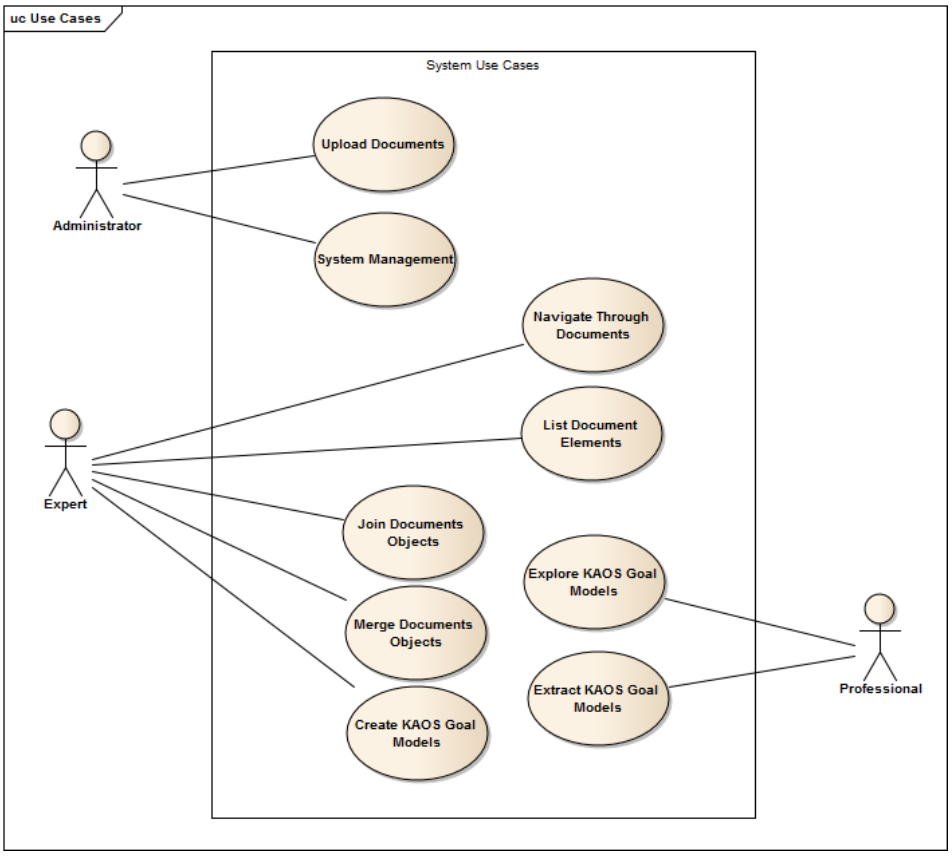

Figure 1 - Proposed System Use Cases

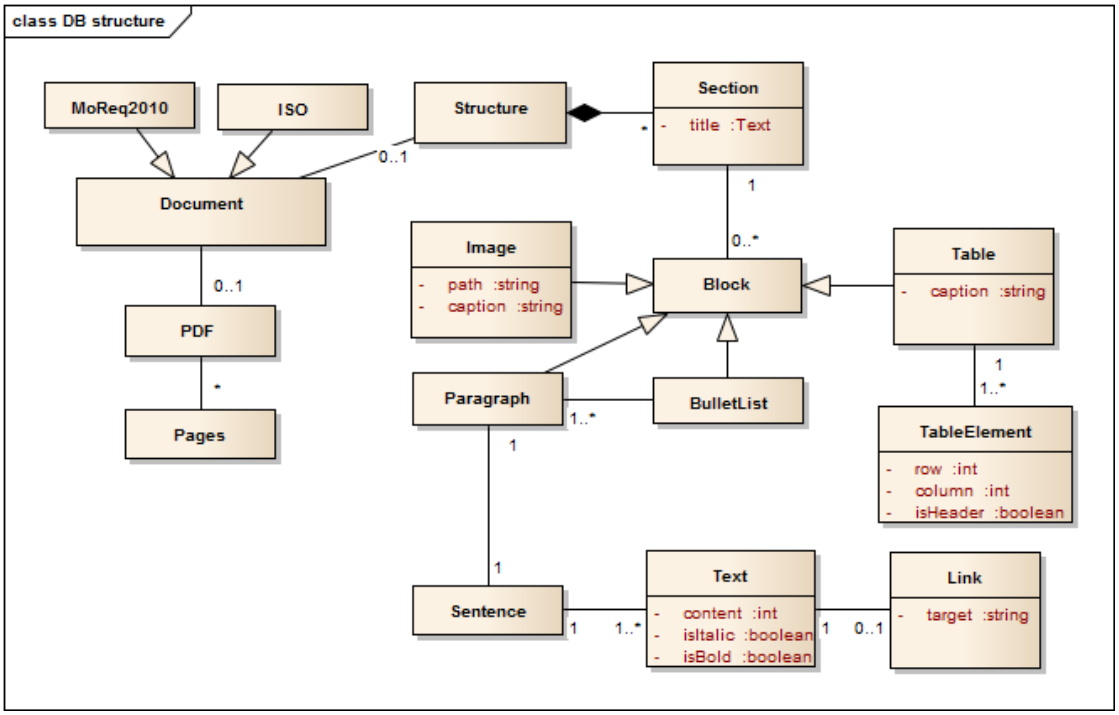

Figure 2 - Documents Structure Model

5.2.2 KAOS Models

The KAOS meta-model contains goals, requirements, expectations on the environment of the system, conflicts between goals, obstacles, entities, agents, etc. These components altogether allow for the creation of diagrams to Goal, Responsibility, Object and Operation modeling.

Figure 3 presents a model to support the creation of Goal Models. A Goal is considered the most relevant concept, which can be specialized in Parent Goals or Sub-Goals. Sub-Goals refine Parent Goals with refinement links - AND refinement or OR refinement. A Goal can be also refined by a requirement; this refinement is the last step for the Goal Model creation with requirements being the child leaves.

.

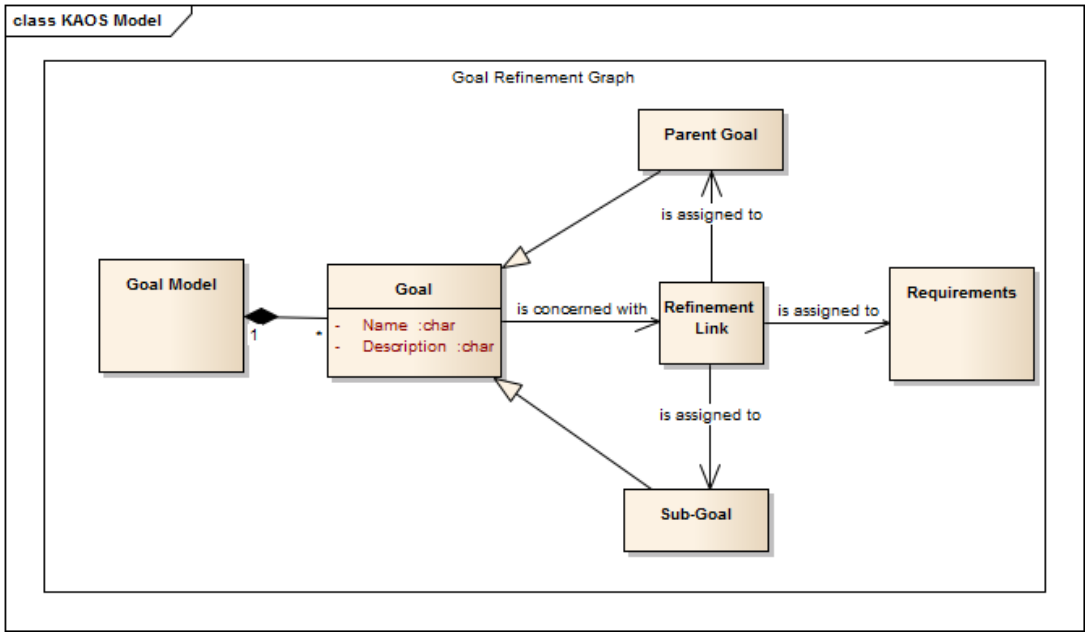

Figure 3 - KAOS Goal Model Support

In order to build a KAOS Goal model the next steps are to be followed:

- Identify goals;
- Refine each identified goal until achieving a set of requirements and/or expectations;

However in this project, we already have a set of requirements and the objective would be selecting and grouping these requirements by creating goals which can be achieved with that set. Therefore in order to build the Goal Model an exercise of analysis of a set of requirements must be the first step, then an attempt to group requirements that present a similar function or that can be thought has being part of the same purpose should be done. This refinement should be made in an upward movement until we reach a set of goals belonging to a major objective/function that a records system must have.

In order to test the model of figure 3, a goal model for each core service of the MoReq2010 specification was made. This exercise was carried after a detailed and careful analysis of each requirement of the core services; then a grouping of requirements having a similar objective was made, and the creation of a goal to join the found set. Next figures present the components used in the StarUML [9] program used to create the diagrams:

Also it was important to find a standard in how one should express the goals in the diagrams, i.e. the naming conventions to models creation.

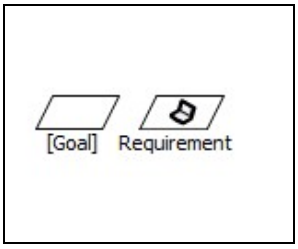

Figure 4 – Goal and Requirement Components

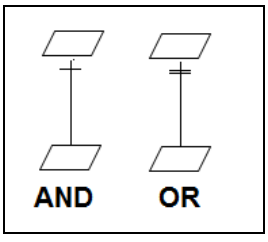

Figure 5 - Refinement Links

In [3] the words are preceded by a verb in its passive form, for example instead of writing "Satisfy Borrower Request" it is written as "Borrower Request Satisfied" and also [10] uses this convention, stating that the reason to this is to avoid confusion between goals and operations (agent behaviors). In the next

exercise this convention was followed being helpful in expressing a text in how to read each diagram.

5.2.3 KAOS Goal Models for MoReq2010 Core Services

In order to simplify this extended abstract, only two core services are presented.

User and Group Service

This service concerns with management of system Users and Groups in order to operations perform successfully. Despite not mandating protocols solutions should use for user authentication and user and group management, MoReq2010 presents a set of requirements that act as a wrapper to allow the use of either an external corporate directory service or a custom directory service built into the MCRS; these are simply basic concepts of a user and a group. Figure 6 presents its goal model.

.

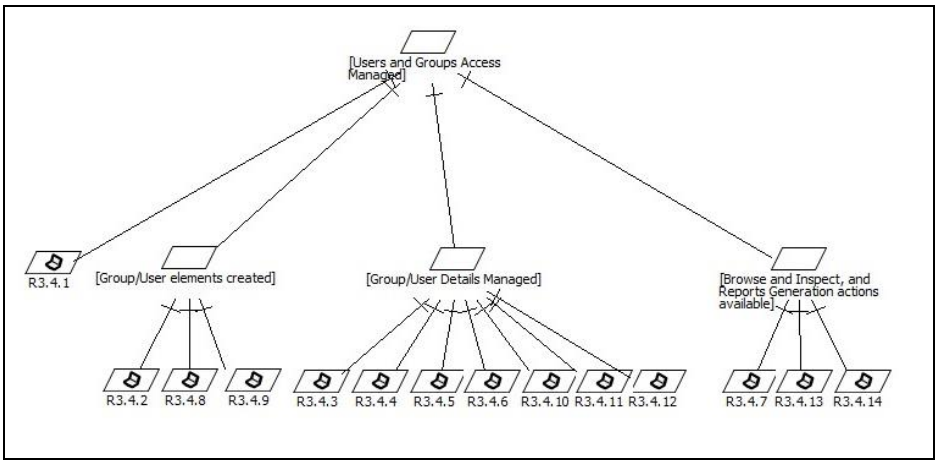

Figure 6 - Goal Model for User and Group Service

Requirement R3.4.1 concerns the access to the MCRS, stating that this must be only made by users that are authenticated and for whom there exists an active user entity, describing also the metadata that should be available for that entity. Group/User elements created contains requirements concerning with creation of groups and users. Group/User Details Managed has requirements updating, deleting, and destroying users or groups, and adding or removing users from groups. Browse and Inspect/Reports Generation actions available contains requirements concerned with reports generation for either listing users or active groups, and browse and inspect users and groups. The diagram reads as follows:

In order for the system to manage Users and Groups of the system, only authenticated users must be authorized to access it, elements for Group/User must be created, details from these elements must be managed and actions for Browsing, Inspecting and Generate Reports must be available.

Model Role Service

MoReq2010 authors claimed that at the time that this specification was made, there was no industry standard concerning how users are authorized to perform functions in the MCRS; for that reason a model role service was created in order to cover that matter. A role is granted to a user in order to give it permission to perform specific functions; this user is then defined throughout the specification as an "authorized user". Figure 7 presents its diagram.

Role/Access Control Lists Created contains requirements for both creations of a Role and ACL (short for Access Control List). Roles/ACL Details Managed has requirements concerning modifications of roles or ACL details, addition and removal of function definitions from active roles, deleting and destroying roles. Browse and Inspect for Access/Permissions Information and Reports Generation on Roles actions available contains requirements concerning the browsing and inspection of roles and function definitions and also an entity's ACL, discovery of authorized functions, and reports listing relationships between roles and function definitions. The diagram reads as follows:

In order for permissions of access to be managed, Role/Access Control Lists must be created, Roles/ACL details must be managed and actions to Browse and Inspect Access/Permission Information as well as Reports Generation must be available.

.

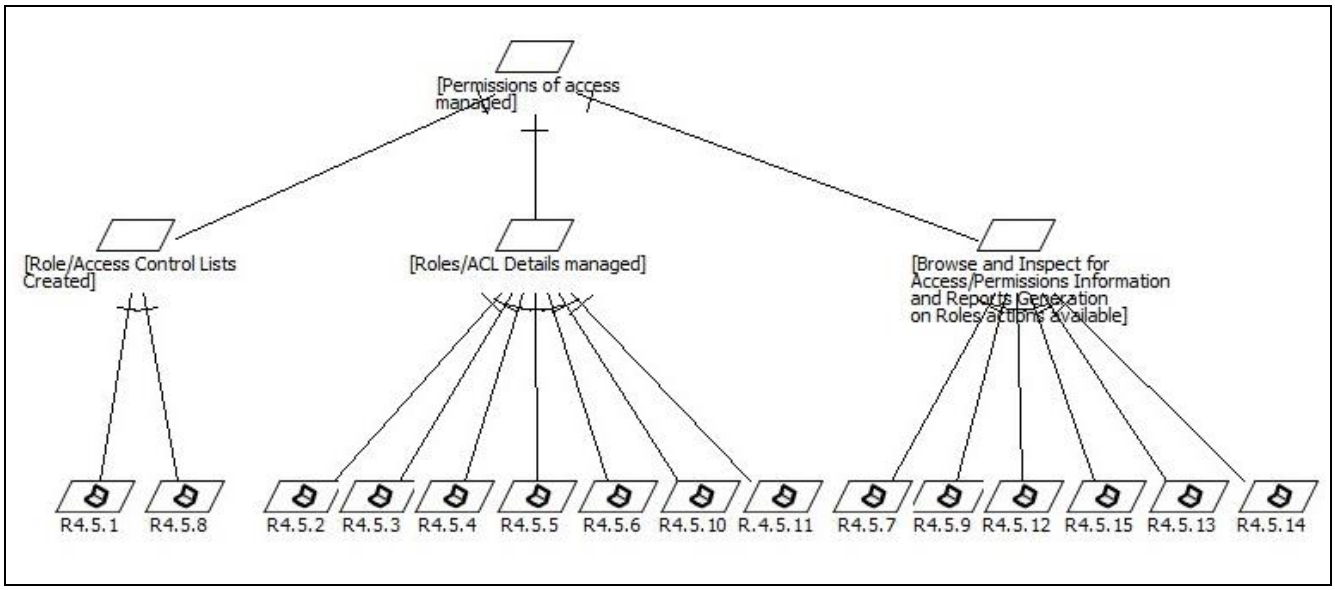

Figure 7 - Goal Model for Model Role Service

5.2.4 ISO 15489 and MoReq2010 alignment

After completing the exercise of creating goals to group the requirements inside MoReq2010 core services, one can attempt to align these results with ISO 15489 main goals to achieve a good records management system. ISO 15489 [8] describes processes of records management in chapter **4.3 Records Management Processes**: Capture, Registration, Classification, Access and security classification, Identification of disposition status, Storage, Use and tracking, and Implementation of disposition. These processes are the core features for records management according to ISO 15489, and with that in mind one can infer some relations between them and MoReq2010 core services features. According to these two standards a records management system should be able to present six basic capabilities:

- Capture of Records;
- Organization of Records;
- Protection/Security of Records;
- Disposal and Retention of disposal of records;
- Access to Records;
- Export/Import of records.

The results of this exercise can also be depicted by a KAOS goal model, present in figure 8:

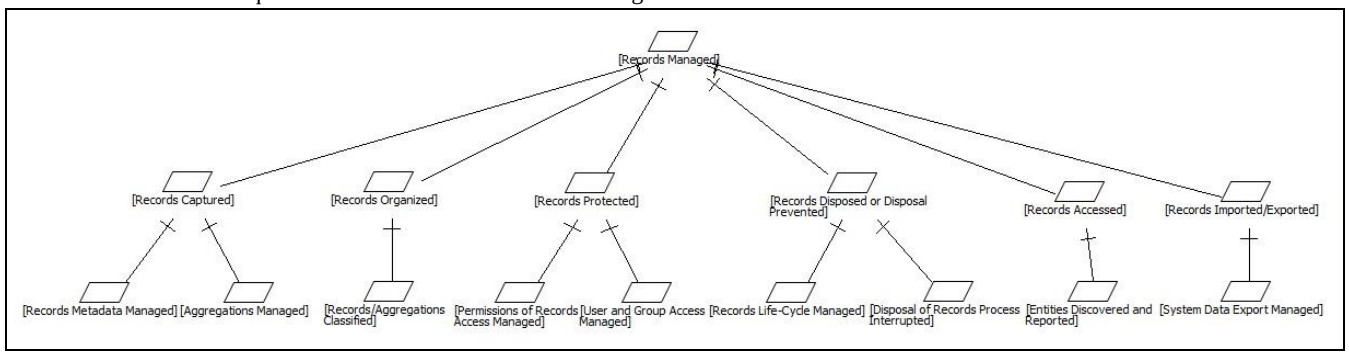

Figure 8 - Goal Model of Records Management Functions

This diagram can be read as follows:

In order for Records to be Managed, Records must be Captured, Organized, Protected, Disposed, Accessed and Exported. For Capture, records metadata and records aggregations must be managed. For organization Records and Records Aggregations must be classified. For Protection, permissions of access to records and user and groups of users of the records system must be managed. For Disposal or Disposal Prevention, records life-cycle must be managed and their disposal of records process interrupted. For Access, entities must be discovered and reported. Finally for Exporting, system data export must be managed.

6. Results

6.1 "Reqs Platform"

The web application that resulted from the proposed solution is denominated "Reqs". The application comprises functionalities to navigate through the MoReq2010 specification in a different and interesting way. It has three main important pages: **Home**, **Overview** and **MoreqDemo**. The **Home** page is for now a blank page, since the major concern was to first focus on having behavior to visualize MoReq2010 data. Figure 9 shows a screenshot of the **Overview** page.

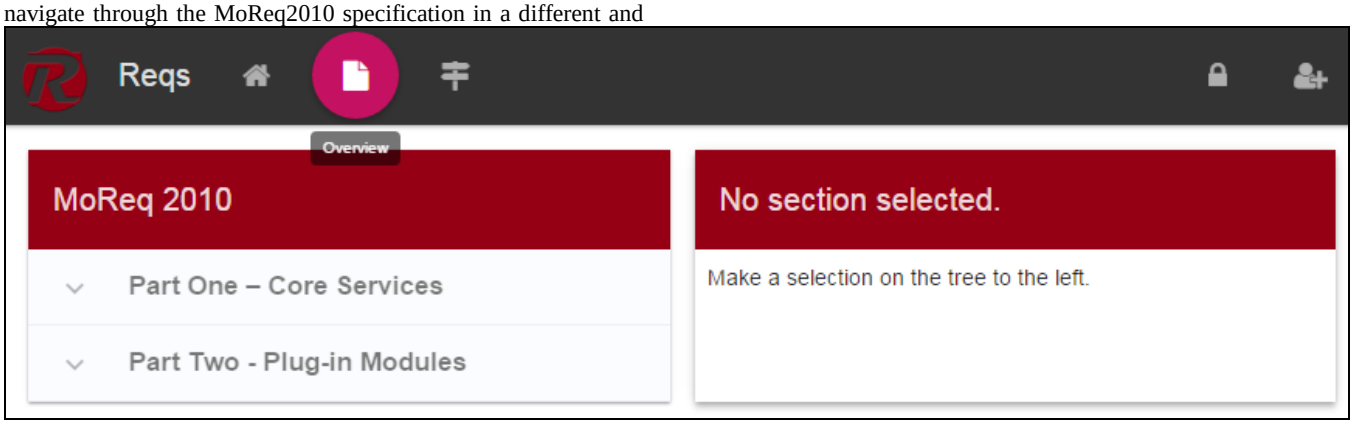

Figure 9 - Overview page for Reqs Application

The Overview page allows for a user to navigate through the document with the help of an index. The index appears on the left panel of the screen, listing all the chapters of the document.

The **MoreqDemo** page has options to list some of the documents most important components: functional requirements, non-functional requirements, and information model. A screenshot of this page can be seen in figure 10, which also shows the tab **Information Model**. The tabs for Functional and Non-Functional Requirements follow the exact same model as figure 9, with a list of the requirements appearing in the left panel and information about each one of them appearing in the right panel.

The Information Model tab also makes use of the two panel visualization. However, instead of having a tab for each component to be listed, the user has buttons available to choose from. There are four buttons as presented in figure 10: **Entity Types, System Metadata Element Definitions, Data Structures, and Function Definitions**. Figure 11 shows an example of listing metadata element definitions.

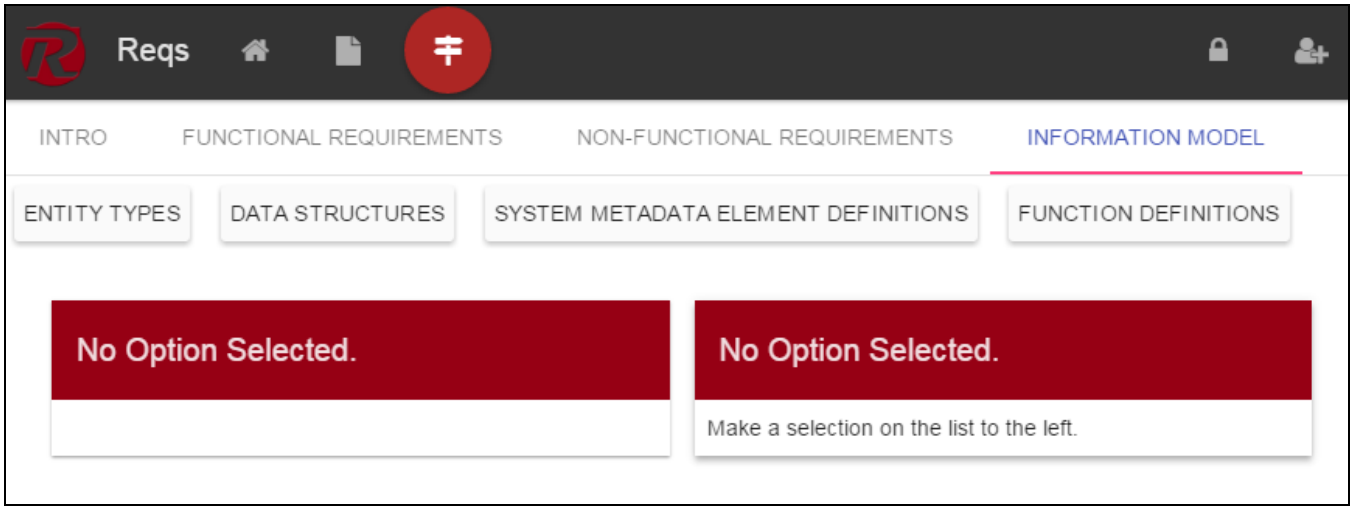

Figure 10 – MoReq Demo page for Reqs Application

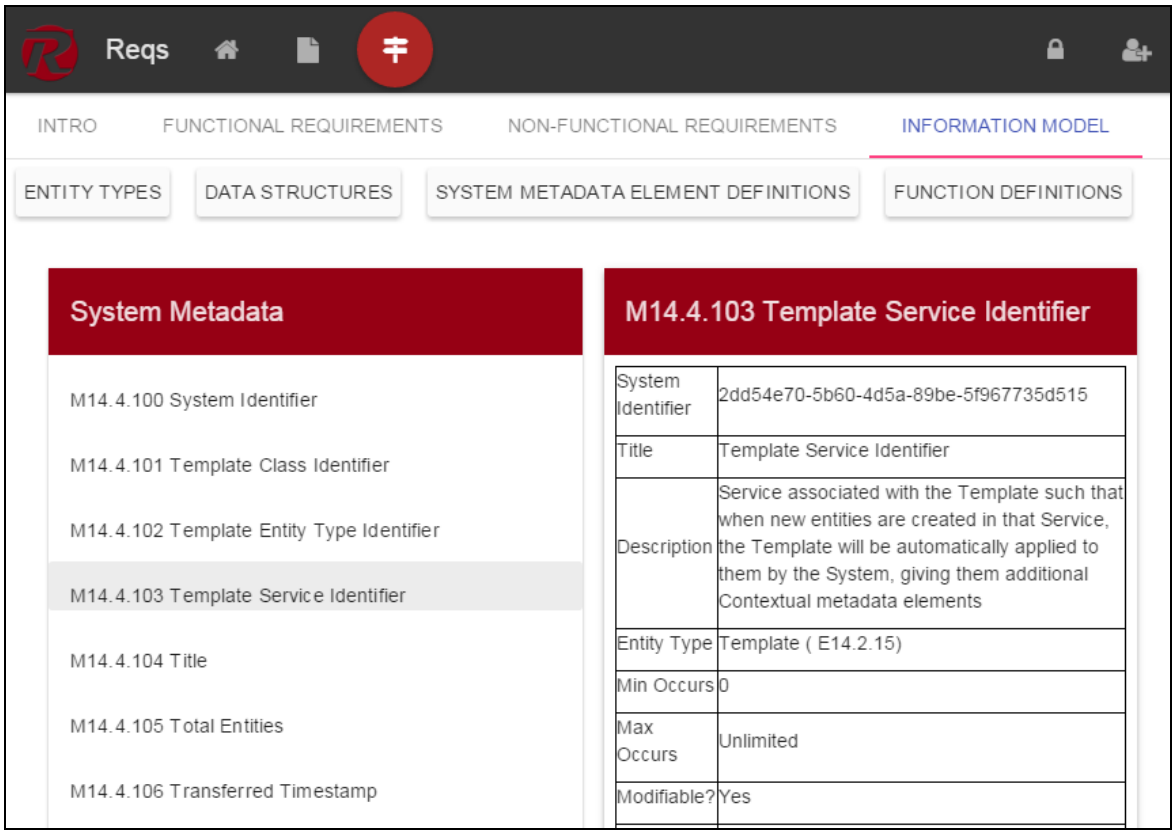

Figure 11 - Example of System metadata definition observation

6.2 Data Problems

MoReq2010 specification on its original PDF format already had some problems, more precisely in the chapters' numeration. When treating the information to be saved in the database it was decided to leave these problems as they were, and later add to the application functionality to correct them. This will ease the tracking of eventual changes to the original official document. These problems can be described as follows:

- Chapter 2.System Services has sub-chapters 2.1, 2.2 and 2.4. There is no sub-chapter 2.3. The same happens with every other chapter, except 1.Fundamentals, 14.Information Model, and the chapters mentioned in the next bullet.
- Chapter 6.Record Service has sub-chapters 6.1, 6.2, 6.3 and 6.5. There is no sub-chapter 6.4. Same situation found in chapter 7.Model Metadata Service.
- Chapter 14.Information Model doesn't present sub-chapter 14.5 Function Definitions in its bookmark index, however the chapter exists and can be viewed by navigating through the pages.
- Chapter 12.3 title has a typo in the word Functional (it appears as Fonctional), in the index, however the title on the document is correct.

7. Conclusion

7.1 Critical Analysis of the Work

The development of this work had several phases. The first phase was related to the definition of the problem and the objective for the whole project. The second and most important phase was the proposal of a system capable of store reference requirements documents for records management, not only structurally but also on its expected behavior. Then came the phase of building the system, starting from the repository were the documents were to be stored; the creation of a parser to read documents in a XML format and store them in the repository and lastly an user interface for data visualization and manipulation. Also, there was a phase were KAOS power was explored, by creating models for MoReq2010 core services to abstract higher goals from the vast set of requirements in the specification.

In this project only KAOS Goal Models where explored, however one could also explore the Object model and the Operation model. MoReq2010 has sufficient detail to objects identification as well as to operations that should be made on these objects. This exploration was not carried away, not only because of time constraints, but also for practical and convenience reasons since it was best to first attempt a simple iteration for testing the concepts before considering anything more complex. When it comes to Responsibility or Agent Model, MoReq2010 very general and generic concepts, which are thought as to fit many different records system as possible, hampers the task of finding specific stakeholders. One can think of a few that could be available in any

case: the developer of the system, the user, the buyer, among others.

Several use cases were proposed for the "Reqs" web application (presented in the analysis chapter), however, only those related to the visualization of MoReq2010 information were possible to implement in time to show some results of it, in the form of screenshots of the application, in this thesis.

7.2 Future Work

The presented system is still in a very initial stage and this work was made to validate that its purpose can be achieved, by exemplifying with MoReq2010 specification. Future steps should be made into adding more reference requirements documents into the system to enrich its repository, using the structure that was here presented.

It would be interesting if the creation of KAOS models could be done in a dynamic way, for example on how it is done in any program built to create diagrams (use case diagrams, UML diagrams, among others). Objects from KAOS models would be available and the user could drag and drop them and create links between them.

One aspect that may also be considered is the extraction of requirements from the system. ReqIF, short of Requirements Interchange Format, is an Organization Management Group standard that allows for the exchanging of requirements data between organizations that do not have a possibility to share the same repository. The exchanging information is stored in XML documents that comply to the ReqIF format, making them processable by different tools on different organizations, allowing for more interoperability. The requirements extraction should be done in this format to ensure that exchanging requirements from this system between a wide variety of tool implementations is tolerated.

8. References

[1] Lapouchnian, A. Goal-Oriented Requirements Engineering: An Overview of the Current Research. Department of Computer Science – University of Toronto, 2005.

[2] Jennifer Horkoff and Eric Yu. 2011. Analyzing goal models: different approaches and how to choose among them. In *Proceedings of the 2011 ACM Symposium on Applied Computing* (SAC '11). ACM, New York, NY, USA, 675-682. DOI=10.1145/1982185.1982334 http://doi.acm.org/10.1145/1982185.1982334.

[3] Dardenne, A., Lamsweerde, A., Fickas, S., Goal-directed requirements acquisitions. Science of Computer Programming 20, (1993) 3-30International Organization for Standardization, ISO 16175: Information and documentation – Records Management – Part 1: Overview and statement of principles., 2010

[4] DLM Forum Foundation, MoReq2010 - Modular Requirements for Records Systems: Core Services & Plug-in Modules., 2010 & 2011.Department of Defense of the United States of America, DoD 5015.02-STD: Electronic Records Management Software Applications Design Criteria Standard., 2007.

[5] Bashar Nuseibeh and Steve Easterbrook. 2000. Requirements engineering: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering* (ICSE '00). ACM, New York, NY, USA, 35-46. DOI=10.1145/336512.336523 http://doi.acm.org/10.1145/336512.336523.

[6] Zave, P. Classification of research efforts in requirements engineering. ACM Comput. Surv., 1997.

[7] Vieira, Ricardo. Requirements Engineering Approaches, Research Topics – PhD Program in Information Systems and Computer Engineering, IST - Technical University of Lisbon, Portugal.

[8] International Organization for Standardization, ISO 15489: Information and documentation – Records Management – Part 1: General., 2001

[9] Sam Supakkul. 2008. RE-Tools: A Multi-Notational Requirements Modelling Toolkit. [ONLINE] Available at: http://www.utdallas.edu/~supakkul/tools/RE-Tools/index.html. [Accessed 08 October 15].

[10] Respect-IT, KAOS Tutorial, V1.0, Oct 18, 2007