A Requirements Engineering analysis of MoReq
Ricardo Vieira, David Ferreira, José Borbinha, Guilherme Gaspar
INESC-ID, Information Systems Group; IST, Computer Science and Engineering Department
Lisbon, Portugal
{rjcv,dferreira,jlb,j.guilherme.gaspar}@ist.utl.pt

Abstract
Purpose - This paper provides an analysis of the MoRe2010 specification while identifying the best practices and techniques of Requirements Engineering’s (RE) body of knowledge that could be applied to improve the quality of this specification.

Design/methodology/approach – Considering the scope of MoReq, we describe the most relevant related RE techniques and common practices. Next, we evaluate the potential impact of applying these techniques to MoReq2010.

Findings – Bearing in mind that MoReq2010 is embodied as a textual document, we discuss how the structure and the writing style of this document could be improved by adopting: (1) a pre-defined structure template, (2) standard-based guidelines, and (3) well-defined requirements quality criteria. Also considering that MoReq2010 is the result of a collaborative process, we also suggest how the authoring process could be improved by using traceability, prioritization, and other requirements management techniques.

Research limitations/implications - We only assessed the MoReq2010 specification as it was made public. Thus, although we consider that requirements discovery approaches are very important for the quality of any final result, we do not discuss the techniques that were eventually used to elicit original stakeholders’ needs.

Originality/value - We synthesize several recommendations that, if followed, will certainly have a strong impact on the overall quality of MoReq2010. Furthermore, as we believe that these recommendations are not disruptive, they can be directly applied against the current version of MoReq2010.

Keywords Requirement Engineering, Recordkeeping, Records Management, MoReq2010

Paper type Research paper

1. INTRODUCTION

It was estimated that in 2009 nearly eight hundred thousand petabytes of information were generated in the world, of which nearly 75% were managed by organizations (Gantz & Reinsel, 2010). Hence, nowadays records management is playing a crucial role to several organizations, thus the management of these records is one of the major concerns of any business.

However, the growing importance of managing records is followed by an increasing market offer of Enterprise Record Management Systems (ERMS). This fact motivated the DLM-Forum to develop the Model Requirements for the Management of Electronic Records (MoReq), which is intended to define a set of baseline requirements for guidance and normalization. MoReq is intended to be a reference model that any ERMS should comply to (even if only partially), in order to effectively manage records, so therefore emerged the term MCRS, standing for “MoReq2010 Compliant Records System”.

Since its first publication in 2001, the MoReq specification has gathered an increasing interest. The attention it has been receiving, since then also led to several reviews and evaluations from the community, which pointed out inconsistencies, errors, and suggestions for improving (Cain, 2002). With the gained insight, a second version of MoReq was released in 2008 (MoReq2), and a third version in June 2010 (MoReq2010). All these versions of MoReq are publicly available at <http://www.dlmforum.eu/>.

Despite being developed by skilled domain experts, we consider that MoReq2010 can be improved, both in terms of organization and content, by better aligning its structure and writing style with Requirements Engineering’s (RE) best practices.

Broadly speaking, RE is a system engineering discipline concerned with discovering the stakeholders of a system of concern, finding their business needs and goals, and documenting these concerns in a manner that is amenable for analysis, communication, and subsequent implementation (Young, 2004). Also, documented requirements provide the basis for acceptance testing, and for supporting the impact analysis of future maintenance activities. Therefore, although the concerns of the DLM-Forum are not the development of a specific ERMS, but instead produce a reference model for normalizing the required features of these software-intensive systems, the purpose of this research work is to assess how RE can support the DLM-Forum to improve future developments on the MoReq specification.
The adoption of RE best practices is of paramount importance because following the trend of the last revisions, the complexity of MoReq might be growing in terms of complexity in an uncontrolled manner. This hinders the effective management of MoReq and limits its practical usage to address real-world problems. Therefore, the understandability of MoReq could be largely improved and the communication barriers could be significantly lessened if more attention is given to requirements development (e.g., specification process, document structure, and writing style) and requirements management (e.g., traceability and change management activities).

The remaining of this paper is structured by nine sections, each identifying a specific MoReq2010 problem and suggestions about how it could be addressed through best practices and techniques of RE, namely by those synthetized in (Pohl K., 2010).

2. REQUIREMENTS ENGINEERING
Regardless of the achievements of both industry and academia, the development of software systems is still a rather challenging process. The size and complexity of modern large-scale systems demand for high-levels of abstraction, conceptual reasoning, and validation upfront. However, none of this can be achieved without clear definitions of the real business-specific needs, i.e., without the real requirements of business stakeholders (Young, 2004). Thus, RE is regarded as one of the most important systems engineering disciplines, since it must support the work of several other disciplines throughout the entire product life-cycle (Hood, Wiedermann, Fichtinger, & Pautz, 2007), providing useful scope and status information to project management upstream and a stable basis for development downstream, including design and testing.

Typically, the development of a system begins with a RE process which deals with the early activities related with the discovering of the purpose, and scope of the system being specified, as also of its associated stakeholders and actors (Nuseibeh, 2000). These early activities belong to the requirements development process, whereas the activities of evolving accepted requirements (e.g., dealing with change requests, impact analysis, tracing, and status-tracking) are regarded as belonging to the requirements management process (Wiegers, SoftwareRequirements, 2003). In short, RE is concerned with real-world goals for the systems functionality, and also how those can be precisely specified and maintained throughout the system’s development process (Zave, 1997).

However, the importance of RE is often underestimated, thus resulting in a large amount of rework to fix misunderstandings that cause a ripple effect toward implementation; one must not forget that any attempt to start technical work beforehand, without a deep understanding of the target system’s purpose, will certainly jeopardize the project outcome (e.g., over budget, behind schedule, lack of quality) (Young, 2004).

The main deliverable of RE is an artefact, usually named requirements document, which contains the detailed textual description of what the target system should do, and eventual constraints on that (Foster, Krolnik, & Lacey, 2004). However, when only natural language descriptions are provided, this form of specification can be ambiguous and, in many cases, inconsistent. The problems can be partially addressed by the adoption of more formal representations (Foster, Krolnik, & Lacey, 2004). However, the translation of natural language requirements into any formal or, at least, semi-formal computer model, is not a straightforward task, especially when it has to involve stakeholders with different backgrounds. Therefore, textual specifications are still the most suitable, fast, and preferred manner by which business stakeholders can contribute and validate requirements (Foster, Krolnik, & Lacey, 2004). In the next sections we’ll analyse MoReq2010 from this perspective, and propose how its specification can be improved.

3. REQUIREMENTS SPECIFICATIONS
By definition, a requirements specification is a document that describes requirements in compliance with pre-defined rules and guidelines (Pohl K., 2010). Besides individual requirements statements, requirements specifications must also contain other contextual information, such as (Hull, Jackson, & Dick, 2005):

- Background information that place the requirements in context (e.g., business and user needs, current system or situation, and the rationale for the new system);
- Domain knowledge necessary to understand the concerning system, including glossary of terms and references to other documents;
- Definition of the system’s goals and scope of the requirements;
- Descriptive text connecting different sections of the document;
- Descriptions of the stakeholders involved;
- Other project-specific topics which are crucial so that stakeholders can entirely understand the specification.

However this additional information cannot be combined with individual requirements statements without a well-defined, coherent, and clear document structure; otherwise it might be troublesome for stakeholders to fully understand its content,
i.e., the system being specified. A good document organization must help readers to quick understand the information stated in the document, the sources and issues related to the requirements and how different requirements fit together (especially when dealing with large and complex specifications, as MoReq2010). These considerations are also important while trying to reduce the number of requirements, detecting errors and omissions, eliminating conflicts between requirements, evaluating new and modified requirements, and reusing requirements across projects (Hull, Jackson, & Dick, 2005). For this reason, templates and guidelines for properly documenting requirements specification have been defined by national and international standardization bodies, business associations, or individual organizations (Pohl K., 2010).

Two relevant references for this kind of work are the IEEE 830-1998 - Software Requirements Specifications (SRS) (IEEE, 1998a) and the IEEE 1233-1998 - System Requirements Specifications (SyRS) (IEEE, 1998b).

These documents establish best practices, providing the basis for conflict resolution between stakeholders. While a SRS solely addresses the specification of software a SyRS has a wider scope, covering the whole needs for a system, independently of its technology (hardware and software).

The IEEE Std. 830-1998 advocates that a SRS should be divided into three top-level parts, as detailed in the Table 1: the introduction, the overall description, and the specific requirements. Unlike the other sections, the section entitled “Specific Requirements” does not have a fixed number of sub-sections as it depends on the project at hand. This section is usually the most extensive part of the document, as hence requirements need to be properly organized to facilitate stakeholders understanding. However, “there is no optimal organization for all systems”, since “different classes of systems lend themselves to different organizations of requirements” (IEEE, 1998a). Thus, for supporting different organizations according to project-specific needs, the IEEE Std. 830-1998 proposes alternatives for structuring a SRS: system mode, user class, objects, features, stimulus or response. If none of the above presents additional benefits for properly organizing the SRS when compared with the other structuring schemas, a functional hierarchy grouped by common inputs, outputs or internal data access should be used.

**Table 1. Recommend Structure for a SRS by the IEEE Std. 830-1998 (IEEE, 1998a)**

<table>
<thead>
<tr>
<th>Table of Contents</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
<td></td>
</tr>
<tr>
<td>1.1. Purpose</td>
<td>The motivation for the specification and identification of the intended audience.</td>
</tr>
<tr>
<td>1.2. Scope</td>
<td>Name of the software product, describing its benefits, objectives, and goals.</td>
</tr>
<tr>
<td>1.3. Definitions, acronyms, and abbreviations</td>
<td>Terms required to fully understanding the specification.</td>
</tr>
<tr>
<td>1.4. References</td>
<td>List of references used in the specification.</td>
</tr>
<tr>
<td>1.5. Overview</td>
<td>Overview of the contents and structure of the requirements specification.</td>
</tr>
<tr>
<td>2. Overall Description</td>
<td></td>
</tr>
<tr>
<td>2.1. Product perspective</td>
<td>Describes the dependencies and relations with other products. If the software is part of a larger system, then this section should make explicit the relation to the broader system by identifying interfaces between them.</td>
</tr>
<tr>
<td>2.2. Product functions</td>
<td>A summary of the major functionalities that the system will perform.</td>
</tr>
<tr>
<td>2.3. User characteristics</td>
<td>Describes the general characteristics of the users of the product.</td>
</tr>
<tr>
<td>2.4. Constraints</td>
<td>Define general constraints that limit the developer’s options, such as regulatory policies, hardware limitations, and interfaces to other applications.</td>
</tr>
<tr>
<td>2.5. Assumptions and dependencies</td>
<td>List of all the factors on which the document content’s rely, since changes in these factors can have a strong impact on the requirements stated in the SRS.</td>
</tr>
<tr>
<td>3. Specific Requirements</td>
<td></td>
</tr>
<tr>
<td>- External interfaces</td>
<td>Details the interfaces described in the section “Overall Description”.</td>
</tr>
<tr>
<td>- Functions</td>
<td>Defines all the functions that the system must perform.</td>
</tr>
<tr>
<td>- Performance requirements</td>
<td>Defines all the performance requirements for the software.</td>
</tr>
<tr>
<td>- Logical database requirements</td>
<td>Defines the logical requirements for any information that is stored in a database.</td>
</tr>
<tr>
<td>- Design constraints</td>
<td>Defines design constraints imposed by hardware limitations or by other standards.</td>
</tr>
<tr>
<td>- Software system attributes</td>
<td>Defines attributes of the system that can serve as requirements.</td>
</tr>
</tbody>
</table>

Appendixes

Index

Therefore we propose MoReq could benefit from the application and enforcement of the guidelines proposed by IEEE Std. 830-1998, especially for the following reasons (Pohl K., 2010):
• This standard has already a proven structure defined by experts of RE;
• A reference structure makes it easy to check, by comparison, if some information is missing;
• By adopting a reference structure, the developers of this specification can focus more objectively on the contents of the document, instead of dwelling on its own subjective organization;
• It’s easier to lookup for information since it can always be found in the same place, even across different versions;
• There are already several requirements management tools that aid users to create a requirement document based upon templates from these reference structures (so others further reusing MoReq will be able to import it properly, even if the DLM-Forum decides not to make use of any of these tools).

4. ANALYSIS OF MOREQ2010 AS A REQUIREMENTS SPECIFICATION
MoReq2010 has a quite different structure from its predecessors. It is divided in two main parts: Core Services and Plug-in Modules. The former starts by presenting modular context and background information and the definition of key concepts. Then the specification follows by describing modular services presenting, for each service, the main concepts and the definition of its functional and non-functional requirements. A service in MoReq2010 is “a subset of the total functionality of an MCRS that focuses on managing only one or a small group of type entities”. The part Core Services ends with a glossary of terms and an information model that specifies all the existing entities, functions and metadata along with the relations between them. The second part - Plug-in Modules – describes modules as extensions to core services, by defining their requirements. Plug-in modules are organized into series and for an ERMS to be compliant with MoReq2010 it must implement at least one module from each series.

The organization of the requirements in MoReq2010 is complex, making it all hard to understand for most readers (Fresko, 2012). In order to identify its deviation from the best practices of RE we compared the structure of MoReq2010 with the IEEE Std. 830-1998 (see Table 1). The result of this comparison allows us to draw the following remarks:

• MoReq2010 contains all the information required for the section “Introduction” in the IEEE Std. 830-1998.
• Regarding the second part of the standard - “Overall Description” - it is only possible to identify a partial match which is devoted to the summary of all major functionalities of the software product. This can be regarded as a disadvantage - as a specification of modular services that can be integrated in other business systems it would be helpful to have a chapter similar to IEEE’s “2.1 Product perspective” for describing how these services can be integrated.
• Considering the remaining section required by IEEE Std. 830-1998, namely the “Specific Requirements” MoReq2010 organizes requirements by modular services. As it was previously mentioned “there is no optimal organization for all systems” since it depends on the type of system being described. However it is important that each section is cohesive thus avoiding cross-section references that might weaken the categorization, difficult understanding, and increases complexity. This principle is especially important in MoReq2010 since it describes modules that, in theory, could be implemented in a standalone manner, without requiring other services. Table 2 summarizes all the existing relations between requirements in MoReq2010. Although the majority of references refer to requirements within the same chapter, there are several cross-sections references. It is also noteworthy the strong dependency between chapter four and eight. The percentage of dependency relations was calculated by taking into account the amount of existing requirements in a given chapter divided by the number of requirements that this chapter relates to. As a final remark, although a reference does not always maps into a dependency relation it would be desirable to reduce or completely remove cross-section references within MoReq2010 in order to reduce its complexity.

5. REQUIREMENTS ATTRIBUTES
Requirements are usually complex entities, represented in structures with several attributes. Aside from a textual statement, requirements can be enriched with additional information for enabling their status control, help structuring; and further processing (e.g., traceability, filtering, and sorting) (Hull, Jackson, & Dick, 2005) (Alexander & Stevens, 2002). These commonly are referred to as “requirements attributes”. While defining a template for a SRS, it is important to properly define these attributes by giving them a meaningful name, and stating how they can be structured, their value types, and range of accepted values (Pohl K., 2010). The set of attributes defined for a particular requirement type is entitled the attribute schema, which depends on the requirement type and the management processes that need to be supported.
Some of these attributes are only relevant during the early RE activities (e.g., elicitation and analysis). However, the majority is crucial throughout the whole software product life cycle (the focus of requirements management activities), and might even be useful to other System Engineering disciplines, such as Project Management. Additionally, attributes might help external or non-technical stakeholders to quickly understand the specification (Hull, Jackson, & Dick, 2005). For this reason it is important to establish the requirements attribute schema upfront, and document these attributes while writing the requirements textual descriptions.

Requirements attributes allows us to be better prepared to ensure that (Hull, Jackson, & Dick, 2005):

- All the requirements have the necessary information related to them, since we have to clearly identify ahead this information when defining or choosing the attributes schema;
- Incomplete information is easily detected since, with a well-defined schema, one must fill out all the pre-defined attributes. An empty attribute is normally a specification gap.
- The training and integration of new stakeholders is facilitated, since the normalized information supports faster searches and helps navigation while consulting the specification.
- Requirements metadata can be ordered in a consistent manner, and made local to the requirement itself, thus facilitating requirements analysis and evolution.
- The comparison of information is facilitated, since requirements of the same type must share the same schema.
- Stakeholders can filter and select the information that is more relevant to them, according to their specific viewpoints.

Two of the most well-known references for requirements attributes are those assembled by the Requirements Working Group\(^1\) (RWG) of the International Council on Systems Engineering (INCOSE), and the Requirements Specification Model (RSM) proposed in (Pohl K., 1996). Additionally, requirements management tools usually provide a set of pre-defined attributes, which can be extended by custom-made attributes, to support requirements management activities (Pohl K., 2010) (Hood, Wiedermann, Fichtinger, & Pautz, 2007).

---

\(^1\) [http://www.incose.org/practice/techactivities/wg/rqmts/]
6. REQUIREMENTS ATTRIBUTES IN MOREQ2010

In MoReq2010 each requirement has the following attributes: a unique identifier (formed by the concatenation of the chapter number, the section number, and the row number), a description of the requirement, and a textual note. Along the text it is also possible to find references to other requirements, functions or metadata.

Using the collection of requirements attributes defined in (Pohl K., 2010), which is partially based on the suggestions of the RWG, we propose in Table 3 a minimal requirements attribute scheme for MoReq. While devising this scheme, we took into consideration that MoReq was developed with the purpose of being a reference model that can (should) be adapted and customized to better fit the specific needs of organizations that are going to acquire or build an ERMS. Thus, our proposal only includes attributes that can be defined by the developers of the specification (without being too specific) yet allowing some flexibility toward further customization. Using these attributes can help MoReq to:

- Improve communication, through identification attributes. Although MoReq already uses unique identifiers, in a specification with more than 250 requirements it is difficult to remember requirements. Therefore we propose to add an attribute to provide a unique name to the requirements.
- Improve understandability, by using context relationship attributes that will help readers to understand the rationale underneath each requirement.
- Improve consistency, by using more structured cross-reference attributes. In turn, this can help reducing complexity.
- Reduce complexity, by using management attributes. These are quite important regarding MoReq’s evolution that already has several versions (MoReq, MoReq2 and MoReq2010) where requirements significantly changed.

Table 3. Attribute schema for MoReq based on the requirements attribute collection from (Pohl K., 2010).

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification Attributes</td>
<td>Attributes that ensure the unique identification of the requirement.</td>
</tr>
<tr>
<td>Identifier</td>
<td>Unique identifier.</td>
</tr>
<tr>
<td>Name</td>
<td>Unique name.</td>
</tr>
<tr>
<td>Context Relationships</td>
<td>Attributes for documenting relationships with the context of the requirement.</td>
</tr>
<tr>
<td>Source</td>
<td>Describes the source(s) of the requirement.</td>
</tr>
<tr>
<td>Reason</td>
<td>Rationale, describing the reason why the requirement is present in the specification.</td>
</tr>
<tr>
<td>Content Aspects</td>
<td>Attributes related to the content of the requirement.</td>
</tr>
<tr>
<td>Requirement Type</td>
<td>Describes the type of the requirement (as identified in Table 1 as “Specific Requirements”).</td>
</tr>
<tr>
<td>Short Description</td>
<td>Summary of the contents of the requirement.</td>
</tr>
<tr>
<td>Additional information</td>
<td>Complementary information.</td>
</tr>
<tr>
<td>Cross References</td>
<td>Describe relations with other analysis, design or development artefacts (e.g., with one or more tests in the framework test or even with another requirements in other documents).</td>
</tr>
<tr>
<td>Status Attributes</td>
<td>Attributes that document the status of the requirement artefact.</td>
</tr>
<tr>
<td>Priority</td>
<td>Describes the importance of the requirement in the context of the whole system.</td>
</tr>
<tr>
<td>Management Attributes</td>
<td>Attributes that document management information about the requirement artefact.</td>
</tr>
<tr>
<td>Version</td>
<td>Version of the requirement.</td>
</tr>
<tr>
<td>Change History</td>
<td>Change log of requirement’s revisions.</td>
</tr>
</tbody>
</table>

7. REQUIREMENTS IN NATURAL LANGUAGE

Aside from the structure of the document and the requirements attributes, there are also quality criteria that an individual requirement must meet, such as those presented in (Davis, 1993) (Hull, Jackson, & Dick, 2005) (Alexander & Stevens, 2002) and (Wiegers, Writing Quality Requirements, 1999):

- …only address one concern (atomicity).
- …be only in one place without missing information (uniqueness).
- …allow implementation within cost, schedule, and technical limitations (feasibility).
- …be clearly understandable and express objective facts with one and only one interpretation (clearness).
- …be exact and succinct (conciseness).
- …not address solution details or any design constraint (abstractness), unless it is really necessary and stated as that.
- …not contradict any other requirement (consistency).
- …allow verification by inspection, demonstration, test or analysis (verifiability).
Expressing requirements in unconstrained natural language (instead of in a formal language) can limit the enforcement and verification of requirements quality criteria. Despite being familiar to all stakeholders (in the sense that it doesn’t require any specialized training) and being universal (in terms of expressiveness), the inherent ambiguity of natural language often introduces requirements defects.

One of the most well-known techniques to reduce the ambiguity of requirements expressed in natural languages is the usage of syntactical patterns, as sentence-level templates (Hull, Jackson, & Dick, 2005) (Pohl K., 2010) (Ivy, 2002), as illustrated in Figure 1. By using a pattern-language, the complexity of writing a requirement is significantly reduced: one must merely choose the most suitable pattern (according to the type of requirement) and providing the data to fill the placeholders of the chosen pattern.

![Figure 1](example.png)

**Figure 1.** Example of a syntactical requirement pattern (Hull, Jackson, & Dick, 2005).

In addition, the usage of syntactical requirement patterns also:

- Facilitates the change of requirements, since we only need to modify the pattern.
- Better supports the tasks of processing, sorting, and filtering the information based on the range of values of each placeholder.

Other relevant technique to avoid ambiguity is the usage of glossaries or controlled languages, i.e., defining a restricted grammar and a limited set of terms for a specific domain (Pohl K., 2010).

Table 4. Natural language analysis of requirements in MoReq2010.

<table>
<thead>
<tr>
<th>Requirements</th>
<th>atomic</th>
<th>unique</th>
<th>feasible</th>
<th>clear</th>
<th>concise</th>
<th>abstract</th>
<th>consistent</th>
<th>verifiable</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2.4.3 - The MCRS must allow an Authorised user to Browse across its services, or bundles of services under R2.4.1, and Inspect the Metadata of each as listed under R2.4.2.</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>?</td>
<td>Yes</td>
</tr>
<tr>
<td>R4.5.08 - The MCRS must automatically Create an Access control list (D14.3.2) for each Service, or bundle of services under R2.4.1, and for each Entity in the MCRS where so specified, with the following Metadata: Include Inherited Roles Flag (M14.4.43). Each Access control list also has: Access control entries for that Entity.</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>?</td>
<td>Yes</td>
</tr>
<tr>
<td>R6.5.20 - The MCRS must allow an Authorised user to Modify the Title and Description of an active Component, and any of its Contextual metadata.</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>?</td>
<td>Yes</td>
</tr>
<tr>
<td>R8.4.14 - The MCRS must update the disposal status of any Record when requested by an Authorised user and, either immediately or periodically, and at least daily, the MCRS must update the disposal status of all active records.</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>?</td>
<td>Yes</td>
</tr>
<tr>
<td>R10.4.16 - The MCRS must allow a User to combine, chain, or join, the results of several Search queries so as to answer Complex search enquiries.</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>?</td>
<td>No</td>
</tr>
</tbody>
</table>

8. NATURAL LANGUAGE IN MOREQ2010 REQUIREMENTS

For analysing the natural language patterns and writing style used within MoReq, we selected a set of requirements and classified them according to the quality criteria previously described. Table 4 presents the analysis results.

- R2.4.3 is considered not atomic, as it expresses two needs: “browsing services (…)” and “inspect metadata”. It also is not clear, as we must avoid referring to any subject without expressing that subject explicitly: “of each” may be about the services, but can’t it be about the users, or the system itself?
R4.5.08 expresses three different needs, which makes it non-atomic and unclear. It is hard for a requirement to be clear if it is too long, as this one.

R6.5.20 is also non-atomic and unclear: the subject in “its Contextual metadata (…)” is probably the component, but it can also be the user or the system.

R8.4.14 is non-atomic, as it expresses more than one need, and unclear, as it offers different alternatives, in a confused way. How must we understand “immediately”? And if the “update” is requested to be done periodically, which periods must be considered, besides “daily”?

R10.4.16 is unclear, as it makes use of the term “several” in a way that it cannot be verified.

We can generalize the results of this analysis to the remaining of MoReq2010, namely in terms of atomicity and clearness. However it is noteworthy that MoReq2010 is more normalized than its previous predecessors, as all the analysed expressions begin with statements such as “The MCRS must …”, a concern that was not clear in the previous versions. This change improves the clearness and understandability of individual requirements.

To better understand the impact of these requirement defects within MoReq2010, we refer to the work of Guilherme Gaspar (Gaspar, 2012). Gaspar analysed 174 functional requirements of MoReq2010’s core services regarding completeness, ambiguity, clearness and atomicity. As the result of this analysis (summarized in Table 5) we derived 200 recommendations that, after further processing, were reduced to 141 (59 were marked as false positives). Each recommendation indicates that a requirement does not follow one of the checked quality criteria and therefore needs to be clarified or changed.

Table 5. Requirement analysis of MoReq2010 requirements (Gaspar, 2012).

<table>
<thead>
<tr>
<th>Quality</th>
<th>Check Criteria</th>
<th># Rec</th>
<th>% FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete</td>
<td>Search for words that indicate incompleteness (e.g. “not determined”, “not defined”, composed words)</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Unambiguous</td>
<td>Search for ambiguous words (e.g. “significant”, “greater”, “easy”, etc.)</td>
<td>21</td>
<td>14%</td>
</tr>
<tr>
<td>Clear</td>
<td>Search for word repetitions</td>
<td>14</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>Search for implicit references (“that”, “this”, “its”, “theirs”)</td>
<td>102</td>
<td>39%</td>
</tr>
<tr>
<td>Atomic</td>
<td>Count words (up to 50 words)</td>
<td>63</td>
<td>25%</td>
</tr>
<tr>
<td>Number (#) of Recommendations (Rec) and Percentage (%) of False Positives (FP)</td>
<td>200</td>
<td>30%</td>
<td></td>
</tr>
</tbody>
</table>

9. REQUIREMENTS MANAGEMENT

Requirements management encompasses all activities that deal with new and changed requirements and how these changes affect the overall specification as a cohesive whole. Requirements are not static; they evolve over time, while one gains a deeper insight on the problem. Often the originally stated requirements are not necessarily the real requirements. Hence, after analysing and obtaining a stable requirements baseline, one must define a rigorous change management process to assist the incorporation of change requests without compromising the consistency of the previous baseline. Otherwise, it would certainly lead to a scenario out of control. This process must rely on impact analysis to support the decision making process of whether including the new or altered requirement.

Therefore, the goal of requirements management is to systematically assess the benefit of change requests that arise from an evolving environment and a deeper insight on the business needs and modify the RE-related artefacts accordingly (Pohl K., 2010). In the next sub-sections, we identify classic activities performed during requirements management that should also applied to MoReq.

9.1 Requirements Traceability

Traceability provides the means to describe a requirement life-cycle from its origin to its implementation. Requirements traceability information must be able to support several activities, such as verifiability and acceptance tests, change management, and reuse of requirements (Hull, Jackson, & Dick, 2005) (Pohl K., 2010).
Normally the traceability information is described in the form of trace relations, where the origin and source of the relation is stated along with the type of the traceability relation. In natural language, the most common approach to document traceability is to use requirement identifiers to express a relationship (e.g. “R1.17 is based on R2.18”).

Examples of common techniques to overview trace relations are (Pohl K., 2010) (Wiegers, More About Software Requirements, 2006):

- **Traceability matrices**, where rows and columns represent respectively the source and destination of the relation and the intersection between them describes the type of relation.
- **Traceability graphs**, where nodes represent the requirements and edges represent the relationship types.

In fact, traceability is not restricted to the definition of relations between requirements. These relations can (and should) be extended to describe relations between textual requirements in all other kinds of artefacts, such as project goals, use cases, design artefacts, and source code (Hull, Jackson, & Dick, 2005) (Pohl K., 2010).

MoReq2010 is part of a continued project with already three versions published. Thus, it would be important to define traceability relationships between the requirements already established in these versions. For instance traceability could have helped stakeholders to:

- Understand the implications of a given change on a single requirement.
- Understand the impact of not fulfilling a given requirement, in terms of requirements dependency.
- Easily picking a cluster-like set of cohesively related requirements.
- Identify detailed sets of functionality (at technical level) by choosing a set of more abstract organizational goals (at business level).

Finally, traceability relationships to other artefacts related with an ERMS (e.g., standards, specifications, reports) can also be defined in MoReq, in order to provide a better understanding of the content of its requirements.

### 9.2 Prioritizing Requirements

Defining requirements priority is a matter of selecting the requirements that will be subject of evaluation, selecting the prioritization criteria that will be systematically used, selecting an appropriate prioritization technique, and ask responsible stakeholder(s) assess the selected requirements accordingly.

Defining requirements’ priority might be considered a subjective task, since it strongly depends not only on the project nature (size and complexity), but also on its specific stakeholders. Each project is unique and one must take into consideration the so-called Project Management Triangle² (also known as the “Iron Triangle”): the challenges to balance budget, schedule, and scope. For instance, having a tight schedule to complete the project might lead to a scope reduction decision, by picking a set of requirements that will not be implemented. Also, requirement triage (based on priorities) facilitates the decision of the implementation order (requirements with high priority are implemented first), resolves potential conflicts, provides a requirements ordering rationale within a specification, the requirements validation order, etc. (Pohl K., 2010).

The requirement priority within MoReq is a value that ultimately needs to be defined by the organizations that are using and adapting this requirements specification. However, this does not imply that the developers of MoReq cannot provide their own default recommended prioritization, so that it can be reused or adapted according to stakeholders’ needs. In fact, it is already defined in the two first versions of MoReq that the use of the word “must” in a requirement means that it is a mandatory requirement and that the word “should” means it is optional. Also, MoReq2010 previews separate chapters to describe functional requirements for “optional modules”.

Probably without noticing it, MoReq is already using one of the classifications defined in (IEEE, 1998b), which suggests three main classes for requirements prioritization:

- **Essential**: the system will not be valid unless the requirement is implemented (equivalent to a mandatory requirement in MoReq).
- **Conditional**: the requirement will probably improve the system, but not implementing it is still regarded as a satisfactory implementation (equivalent to the “optional modules” defined in MoReq).
- **Optional**: the requirement can be valuable (or not), depending on the stakeholders’ needs (equivalent to an optional requirement in MoReq).

---

10. REQUIREMENTS MANAGEMENT IN MOREQ2010
A requirement is always subject to change: either because of a project context change or because a problem was found (an oversight issue or a wrong assumption). Hence, it is necessary to clearly manage these changes. In fact one complain in previous specifications of MoReq was that once costumers had compliant systems implemented and installed it became complex to make modifications for local variations resulted from a change in the specification (Wetherall, 2011). MoReq2010 intends to mitigate that problem with its new modular approach.

To properly manage changes, it is essential to define a change control board composed of at least one representative of each stakeholder category and decision-makers. This board will then be responsible for accepting and reviewing change requests, evaluating whether the change is acceptable and deciding whether it is going to be included in the project (Pohl K. , 2010). Also, the board will ensure that all stakeholder categories are properly represented and are able to actively take part in the decisions related to their concerns.

DLM-Forum reveals that it is already aware of the benefits of involving all the stakeholders on the MoReq development. For example, there is a consultation portal where stakeholders could review the draft versions of MoReq2010, and can now comment and suggest changes. Also, there is a “MoReq Governance Board” (MGB), with representatives of the main decision-makers group associated with this specification, that is responsible for:

“All ensure MoReq brand name protection, ensure the consistency and quality of MoReq Chapter zero content and translations of the MoReq specification, instigate and promote MoReq awareness, use and adoption in the EU through publications, training programmes and other communication channels and partners, and provide an ongoing programme for the maintenance and testing of MoReq”.

So, according to the best practices of requirements management, we leave the suggestion to DLM-Forum to expand the MGB in order to also include a wider basis of stakeholder representatives, and even create a portal which allows non-member stakeholders to submit change requests at all time. Ultimately, the DLM-Forum could adopt an open, or at least a controlled approach, to allow a wide range of relevant stakeholders outside DLM-Forum to collaboratively contribute and harness the potential of the MoReq specification.

11. CONCLUSIONS
MoReq2010 is a third version of a specification that has been widely used around the world. However the increasing interest on MoReq also led to several reviews and evaluations, which pointed out inconsistencies, errors, and suggestions for improving (Cain, 2002). In general terms MoReq is accused of being too complex and hard to understand uniformly by its main targeted communities (ranging from information managers, or other in similar roles, to software specialists) invalidating its practical usage to address real-world problems. In this research paper we reported the results of an analysis of the document embodying the MoReq2010 specification according to the best practices of RE, and as a consequence we also suggest how these best practices could be used to achieve better understandability and less complexity. It is important to stress that we only assessed the MoReq2010 specification as it was made public, ignoring the techniques that were eventually used to elicit the original needs of the stakeholders. We recognize that to recommend more effective and coherent best practices that also should be considered. Anyway, we also believe that the proposals here presented have an additional strength, as we believe they are not disruptive, as they can be directly applied against the actual specification.

12. ACKNOWLEDGMENTS
This work was supported by national funds through FCT – Fundação para a Ciência e a Tecnologia, under project PEst-OE/EEI/LA0021/2011.

13. REFERENCES


