Abstract

Purpose - This paper describes an analysis of the MoReq2 and MoRe2010 specifications and points to best practices and techniques from the Requirements Engineering (RE) body of knowledge that should be applied to improve these specifications.

Design/methodology/approach – First, we describe the most relevant techniques and common practices from RE that we consider relevant for this scope. Next, we evaluate the potential impact of applying those to the two versions of the MoReq specifications under consideration.

Findings – Considering that the MoReq specifications are provided as textual documents, we discuss how the structure of these and the writing style of these documents can be improved by adopting a predefined structure template, standard based guidelines, and quality criteria as commonly recommended by the RE best practices. Also considering that these documents are the result of a process, we also suggest how the DLM-Forum can improve the management of that process by using traceability, prioritization, and change management techniques.

Research limitations/implications - We focused only on the MoReq documents made public, so we do not discuss the approaches eventually used to gather stakeholders’ needs prior to that. We also consider that as a very important issue, complementing this work.

Originality/value - We synthesize several recommendations that, if followed, will have a strong impact on the overall quality of MoReq. Furthermore, the proposals have an additional strength, as we believe they are not disruptive, as they can be directly applied against the actual specification.

Keywords Requirement Engineering, Recordkeeping, Records Management, MoReq

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% was managed by organizations (Gantz & Reinsel, 2010). Hence, nowadays records management is playing a crucial role to several organizations, motivating the management of these records as major concerns of any business process.

However, the growing importance of managing records is accompanied by an increasing offer of the market of Enterprise Record Management Systems (ERMS). That motivated the DLM-Forum to develop the Model Requirements for the Management of Electronic Records (MoReq), intended to define a set of baseline requirements for reference, guidance and normalization. MoReq is intended to be a specification model that any ERMS should comply to (even if only partially), in order to efficiently manage records. In MoReq 2010 we even can find the concept of MCRS as a “MoReq 2010 compliant records system”.

Since its first publication in 2001, the MoReq specification has gathered an increasing interest, making it an almost de facto standard for this area of concern (Piers, 2003). The attention it has been receiving also led to several reviews and evaluations from the DLM-Forum community, which pointed out inconsistencies, errors, and suggestions for improving (Cain, 2002). As the result of the insight gained, a second version of MoReq was developed in 2008 (MoReq2), and a third version was published in June 2010 (MoReq2010). All these versions of MoReq are accessible from <http://www.dlmforum.eu/>.

In spite of the fact that MoReq specifications have been with no doubt developed by skilled subject matter experts, we consider that the way the documents have been structured and the writing style of the requirements expressions are not the best, which we believe can be improved by aligning that with the fundamental Requirements Engineering (RE) best practices.

Broadly speaking, RE is a system engineering discipline concerned with discovering stakeholders, finding their business needs and goals, and documenting these concerns in a manner that is amenable for analysis, communication, and subsequent implementation (Young, 2004). Furthermore, documented requirements are the basis for systems’ testing and acceptance, as also to support the impact analysis of future systems’ maintenance activities. The motivation for this work came from the fact that we realized these are also concerns shared by the DLM-Forum.
The successive revisions of MoReq have been growing in terms of complexity in an uncontrolled manner. This fact hinders the effective management of the specifications and invalidates 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).

Following, we analyze these issues in the MoReq specifications, pointing suggestions to improve its quality by aligning it with the best practices of RE. We point out techniques and good practices of the requirement development process, more precisely on requirements documentation and on the process that can be used to improve the global quality of MoReq. Each of the following sections identifies a specific MoReq problem and, after describing its context, presents how it can be addressed through RE best practices and techniques, namely those covered in (Pohl K., 2010).

2. REQUIREMENTS ENGINEERING

A requirements specification by definition is a document that describes requirements in compliance with pre-defined rules and guidelines (Pohl, 2010).

The size and complexity of modern large-scale systems demand for high-levels of abstraction, conceptual reasoning, and validation upfront, where clear definitions of the real business-specific needs are required. However, regardless of the achievements in both industry and academia, the development of software systems is still a rather challenging process. From all systems engineering disciplines, RE is pointed as one of the most important, since it must support the work of several other disciplines throughout the entire product life-cycle (Hood, et al., 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. That process deals with the early activities related with the discovering of the purpose, scope, stakeholders and actors of the system being specified (Nuseibeh, 2000). These early activities can be classified as the requirements development process, whereas the activities of evolving accepted requirements (e.g., dealing with change requests, impact analysis, tracing, and status-tracking) can be regarded as the requirements management process (Wiegers, 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, often the importance of RE is underestimated, resulting in a large amount of rework: 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 artifact, usually called requirements document, which contains the detailed textual description of what the target system should do, also including constraints on that (Foster, et al., 2004). However, this form of specification, when based only on natural language, can be ambiguous and, in many cases, unverifiable, which can be addressed by the usage of an equivalent machine-executable representation (Foster, et al., 2004). However, the translation of natural language requirements into any formal or, at least, semi-formal computer model, is not a straight task, especially when it has to involve stakeholders with different backgrounds. Therefore, natural language textual specifications are still the most suitable, fast, and preferred manner by non-technical stakeholders to contribute and validate requirements specifications (Foster, et al., 2004). In the next sections we’ll analyze the MoReq specifications from this perspective, and propose how these can be improved.

3. REQUIREMENTS SPECIFICATIONS

Requirements documents also must contain other items than only requirement statements, such as (Hull, et al., 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 relevant information regarded as crucial, so all stakeholders can entirely understand the specification.

Combine additional information with requirements statements requires 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 arrangement must make it easier to understand the information stated in the document, the sources and issues related to the requirements and how different requirements fit together (especially when these are in large number, as in MoReq). 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, et al., 2005). Accordingly, templates for guidelines and properly documenting requirements specification are usually defined by national and international standardization bodies, business associations, or individual organizations (Pohl, 2010).

Two important references standards from the Institute of Electrical and Electronics Engineers (IEEE) for documenting requirements 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 the connection between the business and the technical stakeholders that will specify and build the envisioned system. A SRS addresses the specification of an only logical (software) system, which in turn may belong to a SyRS for a heterogeneous (logical and also physical) system. Independently of its outer context, a SRS must contain all the functionality of the software system, as well as all the assumptions and constraints of its environment. However, it also is pointed that a SRS should not describe, or unnecessarily constrain, any design or implementation details.

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. We must notice that the section “Specific Requirements”, unlike the others, does not have a fixed number for the sub-sections, as it depends on the project at hand. This section is usually the most extensive (and important) part of the document, as hence requirements need to be properly organized to facilitate stakeholders understanding. However, “there is no one optimal organization for all systems”, since “different classes of systems lend themselves to different organizations of requirements” (IEEE, 1998a). Thus, for supporting different organizations of SRS 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 use.

Therefore we propose MoReq could benefit from the application and enforcement of the guidelines proposed by IEEE Std. 830-1998, especially for these reasons (Pohl, 2010):

- This standard has already a proven structure defined by experts of Requirements Engineering;
- 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 could focus on the contents of the document, instead of dwelling on its organization;
- All the information 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.

4. ANALYSIS OF MOREQ AS A SOFTWARE REQUIREMENTS SPECIFICATION

The structures of the first version of MoReq and of MoReq2 do not differ much. They start with an introduction, an overview of broad ERMS requirements, a list of the functional requirement, followed by non-functional and the so-called metadata requirements, then a reference model that provides a summary of ERMS systems and finally several appendixes.

MoReq2010 has a quite different structure from its predecessors. It starts by presenting “important” and background information, the purpose of the specification, and the definition of key concepts. Then it follows with the definition of functional and non-functional requirements, a glossary of terms, a summary describing the relationship of MoReq with other specifications and related bibliography. It concludes by listing all the objects and functions that where mentioned within the requirements statements.

After comparing the structures of these MoReq specifications with the IEEE Std. 830-1998, we conclude that:

- All MoReq versions contain the information required for the section “Introduction”, at the first top-level part of the standard.
- Regarding the second part of the standard, for “Overall Description”, it is only possible to identify a partial match with the Section 2.2 of MoReq, which is devoted to the summary of all major functionalities of the software product.
- Considering the remaining sections proposed for a SRS structure, namely the “Specific Requirements”, is possible to identify the attempt of applying some constraints and assumptions. However, within the core of all the MoReq documents, i.e., the requirements specification, it is possible to find a mix of requirements, according to a
hierarchical structure organized by system features, and not by types as defined by IEEE Std. 830-1998. The MoReq2010 has a different structure from its previous versions, as it organizes requirements by objects and features. We are not aware of the reasons behind this disruptive structural change, but we found this new classification schema is not always consistent, and it even can promote confusion, since sections focusing features have several requirements related to objects of the system and vice-versa.

Table 1. Structure of a SRS recommended by the IEEE Std. 830-1998 (IEEE, 1998a)

<table>
<thead>
<tr>
<th>Table of Contents</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1. Introduction</strong></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>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>2. Overall Description</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>3. Specific Requirements</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<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**

5. REQUIREMENTS ATTRIBUTES

Requirements are usually complex entities, represented in structures with several attributes. Aside from a textual statement, requirement can be enriched with additional information for enabling their status control, help structuring, and further processing (e.g., traceability, filtering, and sorting) (Hull, et al., 2005) (Alexander & Stevens, 2002). Those information elements are commonly referred to as “requirements attributes”, and also might be structured in name of the attribute and the semantics of both the attribute itself and the values it allows (Pohl, 2010). The set of attributes defined for a particular requirement type is entitled the attribute schema, which must depend on the type of the requirements and of the management processes that need to be supported.

Despite some attributes are only relevant during the early RE activities (e.g., elicitation and analysis), it is usually considered that their majority is crucial throughout the whole software product life cycle (the focus of requirements management activities), and might even span to other System Engineering disciplines, such as Project Management. Furthermore, attributes might even be useful to facilitate understanding of the document to external or non-technical stakeholders (Hull, et al., 2005). For this reason it is important to establish the requirements attribute schema upfront, and document these attributes while writing the requirements textual descriptions.

Through the usage of requirements attributes we are better prepared to ensure that (Hull, et al., 2005):

- All the requirements have the necessary information related to them, since we have to clearly identify it ahead when defining or choosing the attributes schema;
- The detection of incomplete information is facilitated since, with a well-defined schema, one must fill out all the predefined attributes of the expected information. An empty attribute is normally a specification gap.
- The training and integration of new team members or stakeholders is facilitated, since the structured information can facilitate information search and its consultation.
- The information 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 of requirements attributes are those assembled by the Requirements Working Group\(^1\) (RWG) of the International Council on Systems Engineering (INCOSE), and Requirements Specification Model (RSM) proposed in (Pohl K., 1996). Also, requirements management tools usually provide pre-defined attributes and support to manage that (Pohl, 2010) (Hood, et al., 2007).

### Table 2. Attribute schema for MoReq based on the requirements attribute collection from (Pohl, 2010).

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</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 artifacts (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 artifact.</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 artifact. These are quite important to MoReq, since it already has several versions where requirements were changed.</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>

### 6. REQUIREMENTS ATTRIBUTES IN MOREQ

In the first version of MoReq the requirements are presented in a tabular form, with one requirement per row. Each requirement has 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.

MoReq2 uses the same structure, but a testability value was added, indicating whether the requirement can be tested through the test framework (a new concept also established in MoReq2). However, MoReq2010 abandoned this tabular form, while maintaining the type of content presented.

Using the collection of requirements attributes defined in (Pohl, 2010), which is partially based on the suggestions of the RWG, we propose in Table 2 propose a minimal requirements attributes scheme for MoReq. While deriving this scheme, we took into consideration that MoReq is only 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, we only included attributes that can be defined by the developers of the specification (without being too specific) and yet allowing some flexibility to further customization.

---

7. REQUIREMENTS IN NATURAL LANGUAGE

Apart from the structure of the document and the requirements attributes, there are also quality criteria that a requirement must meet, such as for example in (Davis, 1993) (Hull, et al., 2005) (Alexander & Stevens, 2002) and (Wiegers, 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 free natural language (instead of in a formal language) can limit the enforcement and verification of quality criteria. While natural language can allow flexibility and provide universality (in the sense that it doesn’t require any specialized training), it also brings the ambiguity inherent to the language.

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 structures (Hull, et al., 2005) (Pohl, 2010) (Ivy, 2002), as the example illustrated in Figure 1. With the usage of a pattern-language, writing a requirement is simplified to merely choosing the most correct pattern (according to the type of requirement) and providing the data to fill the placeholders of the chosen pattern.

In addition, the usage of syntactical requirement patterns also:

- Facilitate the change of requirements, since we only need to modify the pattern.
- Allow better processing, sorting, and filtering of information using the range of values of the placeholders.

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

8. NATURAL LANGUAGE IN MOREQ REQUIREMENTS

To analyze the natural language patterns or writing style used on MoReq, we selected a set of requirements from a short sub-section of MoReq2 and MoReq2010, and classified them according to the quality criteria previously described.

Table 3 presents the assessment results from MoReq2:

- R4.4.1 is considered precise, yet not atomic, since it describes at the same time an action of selecting files and records. Also, the requirement is unclear for the same reason: can an administrator select files and records at the same time, or only one type at a time?
- R4.4.2 cannot be considered unique because it also addresses two backup scenarios (thus, they should be two separated requirements, even if related). It is also unclear because it leaves the question: a “full backup” backs up the information of a “vital backup” or is that exclusive of the latter?
- R4.4.3 is a perfect example of the ambiguity present in natural language, as we have no definition for a “fully operational system”! How can that be verifiable?
- R4.4.4 raises the same issue at R4.4.3, as the concept “clean environment” is used without an unambiguous reference or description of what is the intended meaning of that kind of environment.
- RE4.4.5 should be in fact two different requirements. However, one of them is already addressed in another requirement of the same specification as “R4.28 - The ERMS must log in an audit trail all actions performed on records, volumes, sub-files, files, classes and retention and disposition schedules, regardless of whether the action affects one or more of them. “, thus making it a non-unique requirement.
Table 3. Natural language analysis of requirements in MoReq2.

<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><strong>R4.4.1</strong> - The ERMS must allow administrative roles to indicate that selected files or records contain, or are considered to be, “vital records”.</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>?</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>R4.4.2</strong> - The ERMS must provide two separate back-up operations:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• “full” backup, which backs up all (specified) ERMS data;</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>?</td>
<td>Yes</td>
</tr>
<tr>
<td>• “vital” backup, which backs up only the ERMS configuration and files and records identified as “vital”.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>R4.4.3</strong> - After recovering from a “vital” back-up the ERMS must be fully operational.</td>
<td>Yes</td>
<td>Yes</td>
<td>?</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>?</td>
<td>No</td>
</tr>
<tr>
<td><strong>R4.4.4</strong> - The ERMS should provide for two methods of restoring from a “full” back-up:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• restoration to a “clean” environment, in which the data from the “full” back-up overwrites and replaces the ERMS during the recovery operation;</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>• restoration over an existing environment, in which the data from the “full” back-up is merged back into an existing ERMS environment.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>R4.4.5</strong> - The ERMS must allow administrative roles to indicate that records are no longer considered vital. This action must be logged in the audit trail.</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>?</td>
<td>Yes</td>
</tr>
</tbody>
</table>

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><strong>R2.4.3</strong> - 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><strong>R4.5.08</strong> - 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><strong>R6.5.20</strong> - 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><strong>R8.4.14</strong> - 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><strong>R10.4.16</strong> - 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>

From the same requirements listed in Table 3 it is also possible to conclude that MoReq2 does not follow any predefined syntactic pattern. Different writing styles are used in the same section, although they could have been written with the same
style as others. For example, R4.4.3 could have been described as follows: “The ERMS must be fully operational after recovering from a ‘vital’ back-up”. The lack of such normalization leads to unnecessary confusion to readers and reveals a lack of consistency concerns regarding the specification. It is also noteworthy that the understandability problem, resulting from the lack of the definition of the key terms, is patent in several of the requirements. These problems could have been mitigated with the usage of a glossary or usage of a controlled natural language like was mentioned above.

Table 4 resumed analysis and validation of a small random sample of no well-formed requirements from MoReq2010.

- 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 express 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 extrapolate this analysis to the overall document, where the atomicity and clearness are not frequent, if we consider the rigorous metrics of RE. However it is important to notice that MoReq2010 is more normalized than the previous ones, as all the analyzed expressions begin with statements such as “The MCRS must …”, a structural worry not evident in the previous versions. This improvement offers a much more clear text, with much more comprehensiveness.

9. REQUIREMENTS MANAGEMENT
Requirements management encompasses all activities that deal the new and changed requirements and how these changes affect the overall specification as a 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 analyzing and obtaining a stable requirements baseline, one must define a rigorous change management process to allow the incorporation changes without compromising the consistency of the previous baseline, which in turn would certainly lead to a scenario out of control. This process must rely on impact analysis to support the decision of whether including the new or altered requirement.

Therefore, the goal of requirements management is to observe the system context in order to detect changes, monitor, and manage artifact related with RE concerns (Pohl, 2010). In the next sub-sections, we point classic activities used in requirements management that should be applied in MoReq.

9.1 Requirements Traceability
Traceability is the ability to describe a requirement life-cycle from its origin to its implementation and usage. Requirements traceability information must be able to support several activities, such as verifiability and acceptance tests, change management, and reuse of requirements (Hull, et al., 2005) (Pohl, 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 the traceability information are (Pohl, 2010) (Wiegers, 2006):

- **Traceability matrices**, where rows and columns represent 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. This ability can (and should) be extended to describe relations with all kinds of artifacts, such as project goals, use cases, design artifacts, and source code (Hull, et al., 2005) (Pohl, 2010).

MoReq 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. Traceability could have helped stakeholders to, for example:

- Understand the implications of the change of a single requirement.
• Understand how much will “cost” not fulfilling a requirement, in terms of requirements dependency.
• Easily picking a cluster-like set of strongly related requirements.
• Identify detailed sets of functionality (at technical level) by choosing a set of more abstract organizational goals (at business level).

Considering the requirements listed in Table 3, for example, we can identify that R4.4.2 has a constraint relation to R4.4.3 and to R4.4.4, since the last two can only be fulfilled if and only if R4.4.2 is already correctly implemented. Also, R4.4.1 establishes a pre-condition relationship to R4.4.5, since for a record to be declared as no longer “vital”, it is necessary that the function described in R4.4.1 is available.

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

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, 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 MOREQ
A requirement is always subject to change: either because of a project context change or because a problem was found (an overseen issue or a wrong assumption). Hence, it is necessary to clearly manage these changes.

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, 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:

“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. ACKNOWLEDGMENTS
This work was supported by FCT (INESC-ID multiannual funding) through the national PIDDAC Program funds.

12. REFERENCES
Ivy, H., 2002. Writing Good Requirements. s.l., s.n.
Pohl, K., 1996. PRO-ART - Enabling Requirements Pre-Traceability. Los Alamitos, s.n., pp. 76-84.