A New Metadata Language for Software Requirements

A. Chikh and F. Al-Qershi

Abstract—The main aim of this paper is to elucidate how we can provide a set of more powerful software requirements targeted to facilitate their retrieval. So a central challenge for this research work is to define a new concept of software requirement object as combining two kinds of knowledge: 1) requirements knowledge and 2) metadata knowledge. A new metadata language is proposed in order to qualify, retrieve and reuse requirements. This new language is organized as a hierarchy, including categories, aggregate data elements and simple data elements.

Index Terms—Metadata, semantic interoperability, software requirements.

I. INTRODUCTION

Software Requirements Engineering (SRE) is an integrated knowledge field with its own concepts, techniques, tools, methods, activities and actors. Its importance is due to the role of good software requirements in the success of any software product. In general, SRE process can be split into four activities: 1) elicitation; 2) analysis and negotiation; 3) specification/documentation; and 4) requirements validation. Several techniques for producing high quality requirements specifications are proposed in each of these four activities [1].

However, we are witnessing a lack of research work on semantic retrieving of software requirements. Specifying requirements starts from defining their attributes. This idea leads to propose the concept of software requirement object with its metadata such as the state (i.e. approved, rejected, satisfied, and tested). “Having stored reusable requirements elements, qualified and classified, in a repository might contribute to taking advantage of software development resources and to reducing the error probability in requirements specifications.” [2].

The rest of the paper is arranged as follows. Section II presents the problem statement. Section III is dedicated to the related work. Our main contribution is presented in section IV. Section V illustrates our proposal through an example of application. Finally, Section VI concludes the paper and focuses on future work.

II. PROBLEM STATEMENT

According the authors in [3] the expression metadata interoperability has a very broad meaning and entails a variety of problems to be resolved on: 1) machines must be able to communicate with each other in order to access and exchange metadata; 2) one machine must be able to process the metadata information objects received from another; 3) one must ensure that machines and humans correctly interpret the intended meaning of metadata.

We suppose that metadata could leverage the semantic interoperability among different software requirements thus helping stakeholders to find relevant information across them. The requirements repository becomes then a semantic web. This allows browsing requirements through semantic relations instead of just reading them linearly.

Benefits from semantic interoperability could be as follows: 1) Increasing productivity in SRE activities by reusing and exchanging existing requirements; 2) Integrating requirements in the software development life cycle.

III. RELATED WORKS

Currently there are many requirements metadata proposals, with quite varying potential of qualifying requirements. A Soft Wiki1 Ontology for Requirements Engineering (SWORE) was developed [4]. Stakeholders can tag requirements with freely chosen key words, resulting in an emerging tag space that represents the stakeholders’ vocabulary. The tagging process is also ontologically grounded2. The user can explore the requirements by using the tree navigation and additionally narrow down the shown requirements set by choosing tags from the tag cloud.

Authors in [5] propose to make higher-level requirements more granular by refining and breaking them down at various levels till a particular requirement can be assigned to a single component. Templates provide users with placeholders to provide input on the values of requirements attributes. It is assumed that almost all the requirements can be written in a primitive format <attribute, relation, and value>, e.g., <response time, <=, 10>. Some attributes might be unique identifier, description, priority and rationale, verification method, creator, template on which the requirement is based, and so on.

The authors in an INCOSE Requirements Working Group project [6] discuss the categorization of requirements and suggest definitions for different categories. A category set is used to organize requirements according to a classification scheme which is specific to that category set. A category is a class in a category set.

The author in [7] indicates the metadata importance due to the role that it plays in the completeness of requirements. The

1 Research project, funded by the German Federal Ministry of Education and Research
2 An ontology for tags-- www.holygoat.co.uk/owl/redwood/0.1/tags/
same author in [8] discusses the metadata as one quality measure to judge the degree of requirements goodness.

Unfortunately, these related works didn’t 1) give a clear definition of metadata; 2) make a difference between requirements specific metadata and general metadata; 3) organize metadata as a tree in order to facilitate their understanding and management.

IV. SOFTWARE REQUIREMENTS AS COMPOSITE OBJECTS

We aim to help stakeholders in accomplishing their tasks within a given software lifecycle phase, providing them a set of software requirement objects (SRO). We define SRO, as any entity, digital or non-digital, that may be exploited or reused by stakeholders in building another software requirement.

SRO is composed of two parts: 1) A requirement knowledge (RK), composed of the statement itself; 2) A metadata knowledge (MK), composed of metadata necessary to qualify RK. In this work we will focus more on MK proposing a requirements metadata language (RML).

A. Requirement Knowledge

RK represents the content of a software requirement which is the specification of what the system should do, the services that it provides or the constraints on its operation. RK may range from a high-level abstract statement of a service or constraint to a detailed mathematical functional specification [9]. RK could be organized as a hierarchy indexed by the main MK presented in section B. We show below different categorizations of RK.

Somerville distinguishes two families of RKs according if they are customer oriented (user requirements) or analyst oriented (system requirements). Requirements could be either functional or non-functional. Soren [10] distinguishes two other categories: data and managerial requirements. According Somerville [9] non-functional requirements might be classified as follows: product; organizational; and external requirements.

B. Metadata Knowledge

To facilitate a semantic retrieving of RK we propose to qualify them using domain specific metadata. We present in this section the architecture and structure of the new requirements metadata language RML.

RML corresponds to the metadata schema building block presented in [3]. RML structure is mainly inspired from LOM [11]. In fact RML data model is structured as a hierarchy, including categories, aggregate data elements and simple data elements, which represent the leaf nodes. Just simple data elements have individual values defined through their matching value space and data type. All data elements are optional. A data element could contain a list of ordered or unordered values, rather than a single value. A vocabulary, which is a recommended list of appropriate values, could be assigned to some data elements.

The category "Meta-Metadata" is domain independent, so its composition is similar to that of the corresponding category in LOM [11]. The categories "General"; "Lifecycle"; "Technical"; "Rights"; and "Annotations" are almost similar to their corresponding categories in LOM. Nevertheless their content was adapted to the field of requirements engineering. Finally the categories "Requirements specific"; "Traceability"; and "Taxonomy", which are domain dependent, were completely designed to fit the particularities of the SRE domain.

TABLE I: EXTRACT OF THE RML BASE SCHEMA ("REQUIREMENTS SPECIFIC" CATEGORY)

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Explanation</th>
<th>Value Space / Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Req. specific</td>
<td>This category describes the key semantic characteristics of the RK as a requirement.</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>Level</td>
<td>The level of RK according the goal level scale.</td>
<td>Business; Domain; Product; Design</td>
</tr>
<tr>
<td>5.2</td>
<td>Style</td>
<td>The style used to represent RK. The style might be formal (B language), semi-formal (such as UML diagrams) or informal (Natural language).</td>
<td>Data model; Data dictionary; Data expression; Virtual windows; Context diagrams; Event list and function list; …</td>
</tr>
<tr>
<td>5.3</td>
<td>Data nature</td>
<td>Whether this RK is quantitative or qualitative.</td>
<td>Quantitative; Qualitative</td>
</tr>
<tr>
<td>5.4</td>
<td>Output type</td>
<td>The output of this RK, typically a product or process: (1) Product: documents or data are built in response; (2) Process: procedures for using the things that are built.</td>
<td>Product; Process; Both</td>
</tr>
<tr>
<td>5.5</td>
<td>Derivation</td>
<td>How this RK was obtained</td>
<td>Primary; interpreted; derived</td>
</tr>
<tr>
<td>5.6</td>
<td>Scope</td>
<td>Whether RK covers the entire system or a part.</td>
<td>Global; Partial</td>
</tr>
<tr>
<td>5.7</td>
<td>Intended End Use Role</td>
<td>Principal actors(s) for which this RK was designed, most dominant first.</td>
<td>e.g. An end-user will benefit from this RK when it will be satisfied. A tester will use this RK to validate the final product.</td>
</tr>
<tr>
<td>5.8</td>
<td>Rationale</td>
<td>Motivation for this RK</td>
<td></td>
</tr>
<tr>
<td>5.9</td>
<td>Life Cycle Phase</td>
<td>The phase of the lifecycle this RK has reached</td>
<td>Pre-concept; Concept; Development; Manufacturing; Integration/Test; Deployment; Operation; Support; Disposal.</td>
</tr>
<tr>
<td>5.10</td>
<td>Importance</td>
<td>This RK is assessed as critical important if it is important to project success, particularly to the planning or design effort. Such RK is typically driver of cost, schedule, risk or performance.</td>
<td>Imperative; Critical; Trivial</td>
</tr>
<tr>
<td>5.11</td>
<td>Risk</td>
<td>The assessed risk level of RK in meeting needs.</td>
<td>High; Medium; Low</td>
</tr>
<tr>
<td>5.12</td>
<td>Stability</td>
<td>The assessment of the maturity or stability of RK.</td>
<td>High; Medium; Low</td>
</tr>
<tr>
<td>5.13</td>
<td>Validation Status</td>
<td>Whether, where or to what extent this RK has been validated against the needs.</td>
<td>Pending; Passed; Failed; Inconclusive…</td>
</tr>
<tr>
<td>5.14</td>
<td>Verification Status</td>
<td>Whether, where or to what extent the RK has been checked.</td>
<td>Pending; Passed; Failed; Inconclusive…</td>
</tr>
</tbody>
</table>
The Table I above shows an extract of the RML base schema identifying the categories and their embedded data elements, their explanation and their corresponding value space or examples. The extract is reserved to the “Requirements specific” category.

RML could be used by all the stakeholders during the SRE process.
1) Elicitation: RML is not useful as the requirements are not formalized yet.
2) Analysis and negotiation: Requirements should be qualified during this activity. RML will foster communication between the stakeholders, helping to solve their conflicts.
3) Documentation: RML will help in documenting requirements providing partial views of requirements.
4) Validation: RML will help by matching requirements each other.

V. EXAMPLE OF APPLICATION

This example of application [9] aims to demonstrate the use of RML in qualifying requirements related to the development of a Mental Health Care-Patient Management System (MHC-PMS). It is intended for use in clinics. Clinical staff access and read individual patient records and, for every consultation, update the patient record with details of the consultation and the treatment prescribed. Let’s consider the requirement sample: “The patient record includes fields about the diagnosis of the patient’s condition and the treatment prescribed. The MHC-PMS shall store the patient record according to the following data dictionary: …” Fig. 1 below illustrates how this requirement is qualified through an editor and using the “Requirements specific” category of RML.

VI. CONCLUSION

In this paper we presented an approach to introduce metadata into software requirements engineering building a new requirements metadata language RML. The main contributions are respectively: 1) the definition of the new SRO concept; 2) the design of RML; 3) the development of a prototype which allows the metadata authoring using an application profile.

Now, we are empirically assessing RML in real-world experimentation. As future work, we aim to automatically generate a part of metadata. The purpose of this automatic generation is to make RML use more efficient. Finally given the need of RML in retrieving requirements, it is important to have a design that can achieve the scalability needed to manage requirements in large scale repositories.

REFERENCES