Eliciting and Prioritizing Quality Requirements Supported by Ontologies: A Case Study Using the ElicitO Framework and Tool

Taiseera Hazeem Al Balushi¹, Pedro R. Falcone Sampaio², Pericles Loucopoulos³

¹Sultan Qaboos University, College of Commerce and Economics, P.O. Box 20, PC 123 AlKhod
taisira@squ.edu.om

²Centre for Services Research Group, Manchester Business School, University of Manchester, Booth Street East, Manchester M15 6PB, UK
P.Sampaio@manchester.ac.uk

³Management Science and Information Systems Group, Loughborough University, Loughborough LE11 3TU, UK
P.Loucopoulos@lboro.ac.uk

Abstract

As software complexity grows and clients demand higher quality software, quality requirements can no longer be considered to be of secondary importance. Thus eliciting, specifying, prioritising and validating quality requirements is a pre-requisite to the development of effective and efficient information systems. Despite the critical importance of quality requirements, there is a considerable gap in the breadth and depth of quality requirements engineering support in most Requirements Engineering (RE) approaches. In practice it is often the case to have quality requirements considered as an afterthought in the systems development process. Whilst there is a wealth of modelling techniques and tools for functional requirements, there is very limited support for quality requirements in requirements engineering. Support for quality requirements is usually ad-hoc, without clear guidelines on how to capture, specify and manage quality requirements and also without proper usage of standardised terminologies based on established quality models such as the ISO/IEC 9126 quality model. In this paper, we discuss a quality driven requirements engineering framework and tool that applies knowledge management techniques and quality ontologies to support RE activities. The ontology implements the quality characteristics and metrics prescribed by the ISO/9126 quality model, providing a common vocabulary to address quality concerns/aspects across RE activities. We empirically validate how the framework and tool can be used to effectively support the requirements elicitation and prioritization activities through a case study addressing the development of an intranet portal project at the University of Manchester.

Keywords: Non-Functional Requirements, Requirements Engineering, Quality requirements, Ontologies, Case Study

1. Introduction

Despite the importance of departing from a correct and complete set of requirements to attain high-quality software, there are statistics showing that almost 60% of errors identified in software project deliverables originate from incorrect/incomplete requirements obtained through requirements engineering activities (Weinberg, 1997). Incomplete, conflicting and inconsistent requirements are leading causes of software project failures according to the Standish Group (Standish group, 1994) and also according to the LASCAD and TAURAS project results (Lyytinen & Robey, 1999). Errors in requirements have a significant cost impact due to their propagation into final system deliverables.
Requirements Engineering (RE) is a success-critical factor in software projects (Hofmann & Lehner, 2001) as requirements reflect stakeholders’ needs. Thus, eliciting, specifying, validating and documenting these needs in a format that is amenable to analysis and communication, significantly impacts the effectiveness of software project delivery (Nuseibeh & Easterbrook, 2000). Requirements engineering also plays a pivotal role in identifying requirements for new value adding products and services helping to foster technology driven innovation (Kauppinen, Savolainen & Männistö, 2007).

There are three main types of requirements: (1) Functional Requirements, which are associated with specific functions, tasks, or behaviours that the system must support (Robertson & James, 2006), (2) Non-Functional Requirements (NFRs) also known as *quality requirements* which represent quality properties for a particular service, function, component or the entire system, and (3) Constraints, which impose explicitly defined limits on how products/services are to be constructed (Alexander & Beus-Dukic, 2009).

According to (Zave, 1997; Kotonya & Sommerville, 2001; Pohl, 2010), requirements engineering encompasses several core activities: requirements elicitation, requirements modelling/specification, requirements prioritization/negotiation, requirements validation and requirements management. The importance of RE activities has changed considerably with the capabilities arising from the Internet of Services and the Internet of Things (Pohl, 2010). Novel on demand service-oriented systems often have high levels of uncertainty and complexity originating from the unpredictable quality characteristics attached to the operational environments such as cloud computing and mobile computing platforms and high levels of systemic risk in connection with the overall qualities of the software composition. Eliciting, specifying, validating and agreeing on the key quality requirements associated with novel software intensive systems becomes a critical success factor in the overall systems development lifecycle (Azuma, 2004; Galster & Bucherer, 2008).

Early proposals of requirements engineering approaches (Kotonya & Sommerville, 1996; Loucopoulos & Karakostas, 1995; Maiden, 1998; Mullery, 1979) and associated tools such as DOORS (Dick, 2004; Jackson, 2002) and Requisite Pro (IBM, 2003) provide extensive support for tackling functional requirements, however, they have limited support for explicitly capturing, representing and integrating quality requirements into RE activities. The lack of explicit support for quality requirements is a major hindrance given that stakeholders are increasingly demanding higher quality products with explicit metrics for quality criteria (Doerr et al., 2005). Challenges relating to quality requirements support can be categorized into four major categories: product/functionality related, process related, standards related, and tool support related. The categorization of challenges can be described as follows:

1. **Product/functionality related.** Quality requirements often have a subjective nature, which makes it very hard to properly elicit and capture and as a result, they are often neglected in the process of RE (Ebert, 1997; Neto, Leite, & Cysneiros, 2000). In situations where they are adequately considered, they are usually addressed at late stages of system development or even after systems are deployed
Team members participating in the RE effort may also lack essential knowledge about key quality aspects relevant to a specific IT product or service.

2. **Process and method related.** RE approaches often overlook the need for specific process tasks, notations, and guidelines to address quality requirements while carrying out RE activities. RE approaches usually focus on the specification, modelling, and validation of functional requirements. This is mainly due to the vast majority of methods and techniques used by popular development approaches being specifically aimed at capturing behavioural, functional and data related aspects. When quality concerns are supported, coverage is often limited to a subset of RE activities (e.g., NFR goal graphs applied during elicitation) or has a narrow focus (addressing a single quality concern such as security, safety or reliability).

3. **Standards and metrics related.** Despite the availability of a wide range of quality models and standards to be used as a guidance to integrate quality requirements support into RE activities, the majority of the proposed RE approaches do not follow a quality model standard. There is a lack of a complete set of standard quality characteristics and metrics that can be applied during the process of systems development in order to achieve quality requirements compliance and a uniform terminology to address quality aspects in a project.

4. **Tool related.** Tool support available is often limited to functional requirements engineering, with limited support for eliciting, prioritizing and validating quality requirements.

The overall system construction should involve integrating quality concerns throughout several requirements engineering activities, including the proper elicitation and prioritization of the set of requirements, requirements specification, and also in the process of validation and inspection of the requirements with regard to the quality model. Therefore addressing quality requirements engineering by incorporating constructs tailored to address quality concerns into the RE approach is pivotal to assure that the application fulfils user’s functional and non-functional requirements.

In this paper, we discuss a quality driven requirements engineering framework and associated tool that apply knowledge management techniques including functional and non-functional domain ontologies to underpin effective integration of quality concerns in all RE activities. Ontologies (Gruber, 1993) provide a formal and uniform representation of functional domain and quality domain concerns helping to address requirements engineering issues such as problems of scope and problems of understanding (Christel & Kang, 1992) that are often encountered in requirements artefacts.

The quality ontology implements the quality characteristics and metrics prescribed by the ISO/9126 quality model, providing a common vocabulary to address quality concerns across RE activities. During RE activities, requirements analysts can leverage the functional domain ontologies and the quality ontology to elicit a comprehensive set of requirements specified according to the ISO/IEC 9126 quality
model standard, and also prioritize and validate the set of requirements using the metrics supported by the standard to guide the negotiation and prioritization tasks, facilitating the detection of consistency and completeness gaps in the set of requirements.

The use of knowledge management techniques to support requirements engineering activities also helps in augmenting the knowledge of team members participating in the RE project and facilitates the task of knowledge sharing between team members (White, 2010). We empirically validate how the framework and tool was used to effectively support the requirements elicitation and prioritization activities through a case study addressing an intranet portal development project at the University of Manchester. The ElicitO framework proposed in this paper builds on previous approaches to NFR support such as the works by (Chung, Nixon, Yu, & Mylopoulos, 2000; Cysneiros & Leite, 2004; Firesmith, 2003b) that regard quality requirements as non-functional requirements that can be elicited, specified, prioritized, validated and managed during RE.

The research method underpinning the requirements engineering framework, tool design and tool evaluation discussed in this paper is based on the design research method applied to the construction and evaluation of problem-specific artefacts as discussed in (Winter, 2008). The research outcomes have been derived following established steps in the process of design research: (1) Identify a need; (2) Design/Implement an artefact; (3) Evaluate the artefact; (4) Identify key lessons learnt (Winter, 2008). The research also uses the case study method (Yin, 2008) to underpin the storytelling and context description of the research undertaken throughout the paper. One of the main advantages of using the case study method stems from its capability to present information and consolidate research resulting from the use of mixed methods research designs into a cohesive report (Yin, 2008) (e.g., comparative literature review in section 3, artefact design and implementation in sections 4 and 5, and experiment design, empirical observation using focus groups and interviews in section 6).

The remainder of this paper is structured as follows: section 2 provides a background on quality requirements and the key challenges relating to supporting quality requirements. Section 3 presents related work comparing and contrasting key requirements engineering approaches aimed at supporting non-functional requirements. Section 4 discusses the ElicitO quality driven RE framework and the design of the ontologies underpinning the framework; Section 5 describes the architecture and core requirements supported by the ElicitO tool. Section 6 presents the case study addressing the ElicitO framework/tool evaluation in the context of a web engineering application and Section 7 summarizes the paper, discusses the key contributions, and future work.

2. Quality Requirements Background: Concepts and Support Issues

Despite the key importance of satisfying functional requirements, when the software system reaches the deployment stage, it is considered defective if users find it non-intuitive and hard to use, if it presents low reliability characteristics, if it is difficult to debug, or it is inefficient in resources usage. Low levels
of quality attained by the delivered software usually reflect poor support for quality requirements throughout systems engineering (Borderwisch, 1993). Quality requirements are viewed as system properties such as reliability, response time, safety, availability, etc., that should be enforced by the functional components of the system (Sommerville, 2004; Young, 2003). Quality requirements tend to be properties of a system as a whole, and are often difficult to be verified by assessing components in isolation (Nuseibeh & Easterbrook, 2000). Quality requirements engineering is regarded as a systematic and pragmatic approach to building quality into software systems (Chung et al., 2000).

Quality requirements are often referred to in the RE literature as non-functional requirements (NFRs). (Pohl, 2010) argues that not all non-functional requirements are quality requirements. This happens when the non-functional requirement conceals an underspecified functional requirement and/or other more specific quality requirements that need to be thoroughly refined and specified. To illustrate the notion of an underspecified non-functional requirement, consider a case where a software development effort involves the non-functional requirements “user accounts must be secure” and “accounts should be maintained with reasonable performance”. To illustrate the concept of a underspecified non-functional requirement, Figure 1 shows two examples of non-functional requirements specified using the NFR-goal modelling framework proposed by Chung, Nixon, & Yu (1995). In the example there are two underspecified non-functional requirements or soft goals at the root of the goal graph that need to be satisfied: Performance and Security. The Security non-functional requirement is refined and decomposed into the quality requirements (Accuracy, Confidentiality, and Availability) as illustrated in Figure 1.

The Performance requirement is refined into the quality requirements (Response Time and Space) as described in Figure 1. Following the NFR-goal framework modelling constructs, to satisfy the NFR soft goals (Performance and Security) each sub-goal connected by arcs to the parent nodes (super-goals) representing an {AND} goal decomposition will need to be satisfied in order to satisfy the super-goal. In the decomposition and refinement example each leaf of the goal graph can be further refined into a quality requirement with specific metrics attached to indicate how the quality objective will be attained.

Figure 1. NFR Goal Graph (Chung & Nixon, 1995)
Despite the fundamental relevance of quality requirements in the system development process their support across RE approaches has been ad-hoc and/or often neglected. The reasons for this can be summarised as follows:

- Quality requirements can be subject to different interpretations when not thoroughly specified and refined. For example the usability factor can have different meanings such as, (1) the system is easy to use and users can easily interact with it, or (2) the system is effective enough to help users in achieving the tasks needed.

- Quality requirements have numerous complex and nontrivial interdependencies. They conflict with each other when they make contradicting statements about a software attribute and they cooperate when they mutually enforce such attribute (Egyed & Grunbacher, 2004). For example, the design of a call centre operator’s user interface involves trade-offs between usability and response time. Quality requirements may also conflict with functional requirements (Moreira, Arajo, & Brito, 2002).

- Quality requirements are difficult to measure which in turn makes them harder to validate (Punter, Trendowicz, & Kaiser, 2002) and they are often evaluated subjectively (Thayer & Dorfman, 1990). Consequently it is important that quality requirements are stated in a way that also associates metrics to establish baselines, predict and assess likely quality gaps, and monitor improvement (Kitchenham & Pfleeger, 1996).

- Quality requirements are significantly more difficult to test when compared to functional requirements (Thayer & Dorfman, 1990). This is usually a cascading effect of underspecified non-functional requirements that have not been properly addressed in the RE activities and that have been carried forward to the implementation and testing stages.

To address the challenges described above quality concerns need to be at the centre stage of the overall RE approach.

3. Related Work

There is a wide consensus that successful software projects have usually invested a significant effort into software quality engineering (Kan, 2003). There is also a common understanding that the early stages of a software project, in particular, requirements analysis and elicitation, is a key factor for successful project delivery. Despite the cornerstone importance of quality engineering support in requirements engineering for a successful software project outcome, there is still a limited understanding of how to integrate quality concerns into requirements engineering.
In a broader perspective, all RE approaches provide some level of support for capturing quality requirements (e.g. include the textual description of the requirements after all functional requirements have been elicited). However, in the related work we focus mainly on RE approaches that have developed tailored structures to incorporate quality requirements support into RE activities (e.g., a specific modelling construct that can be used to represent NFRs or process tasks/guidelines that address non-functional requirements). The criteria used to compare quality requirements support among related approaches are described below and summarised in Table 1.

- **Adoption of a standard quality model**: this criterion examines if the approach adopts terminologies provided by standard quality models for guiding NFR engineering activities.

- **Integration with RE activities**: this criterion checks if the quality requirements engineering techniques are integrated into RE activities with the support of explicit techniques to address quality concerns. Some of the proposals are activity specific (narrow activity focus) where they address quality requirements in one/few RE activities using a tailored modelling construct in contrast to a more generic integration of quality concerns across all RE activities.

- **Specific notation for NFRs**: syntactical constructs to differentiate between functional and non-functional requirements.

- **Quality concerns/aspects discussed**: this criterion checks each proposal with regard to examples, case studies and discussions addressing quality model concerns/aspects.

One of the most influential approaches to systematically address quality concerns in software development was the NFR framework proposed by Chung et al, (1994). The framework developed process structures and tool support to enable the elicitation, negotiation and specification of quality requirements throughout the development process. Chung & Nixon (1995) used the NFR-goal framework (Mylopoulos, Chung, & Nixon, 1992) for quality requirements elicitation and negotiation which views quality requirements as goals that might conflict with each other during elicitation. Each goal is decomposed into sub-goals represented by a graph structure inspired by and/or trees used during problem solving. This process continues until the requirements engineer considers the goals/sub-goals refined into sufficient level of detail for the next stage of the development process. RE is carried out with the support of a tool: the NFR-assist.

Kotonya & Sommerville (1996) proposed the viewpoint-oriented requirements definition (VORD), which is a model used during the early stages of RE as a structuring mechanism for requirements elicitation and analysis. This approach can successfully elicit and document the quality requirements related to functional requirements in a viewpoint template, however, there is no standard quality model followed during elicitation. VORD does not address all RE activities (e.g., it does not address how conflict resolution is handled when quality requirements are contrasted vis-à-vis other types of requirements).
### Table 1. Quality Concern Support in RE Approaches

<table>
<thead>
<tr>
<th>RE Approach</th>
<th>Standard model characteristics adopted</th>
<th>Specific notations for requirements</th>
<th>Integration of quality concerns across RE activities</th>
<th>Quality Concerns/Aspects Discussed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chung et al, 94 Chung &amp; Nixon, 95</td>
<td>√</td>
<td>√</td>
<td></td>
<td>Functionality</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Usability</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Efficiency</td>
</tr>
<tr>
<td>Kotonya &amp; Sommerville, 96</td>
<td>√</td>
<td></td>
<td></td>
<td>Usability</td>
</tr>
<tr>
<td>Cysneiros et al, 2001</td>
<td>√</td>
<td>√</td>
<td></td>
<td>Functionality</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Usability</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reliability</td>
</tr>
<tr>
<td>Firesmith, 2002</td>
<td>√</td>
<td>√</td>
<td>ISO/IEC 9126 Characteristics</td>
<td>ISO/IEC 9126 Sub-characteristics</td>
</tr>
<tr>
<td>Brito &amp; Moreira, 2004</td>
<td>√</td>
<td>√</td>
<td>Functionality</td>
<td>Security</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Efficiency</td>
<td>Time behaviour</td>
</tr>
<tr>
<td>Sommerville, 2004</td>
<td>√</td>
<td></td>
<td>Functionality</td>
<td>Security</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Usability</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Efficiency</td>
<td>Time behaviour, resource utilization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Portability</td>
<td>Adaptability</td>
</tr>
<tr>
<td>Bhatti, 2005</td>
<td>√</td>
<td>√</td>
<td>ISO/IEC 9126 Characteristics</td>
<td>ISO/IEC 9126 Sub-characteristics</td>
</tr>
<tr>
<td>Mead et al., 2004, Mead et al., 2008 and SQUARE Tool, 2011</td>
<td></td>
<td>√</td>
<td>Functionality</td>
<td>Security</td>
</tr>
<tr>
<td>ElicitO</td>
<td>√</td>
<td>√ Ontology-Guided</td>
<td>√</td>
<td>ISO/IEC 9126 Characteristics</td>
</tr>
</tbody>
</table>

Cysneiros et al. (2001) integrated the quality requirements elicited from stakeholders into a conceptual model. They utilized the LEL and the NFR-goal framework (Mylopoulos et al., 1992) to capture quality requirements. The approach proposed a process to elicit quality requirements using the Language Extended Lexicon (LEL) to register the vocabulary of a given universe of discourse. LEL is based upon the following notion: understanding the problem’s language without paying excessive attention to a deeper investigation into the problem. The main objective of the LEL is to register signs (words or
phrases) peculiar to a specific field of application. Later, quality requirements and their possible implementation solutions are added to the LEL. The LEL will later drive the construction of the NFR model, which helps to decompose the quality requirements from a vague abstraction to more concrete decisions. Finally, the approach provides a systematic way on integrating the elicited quality requirements into UML use cases and scenarios as well as class, sequence, and collaboration diagrams. The quality requirements are elicited, prioritized, and specified without employing a standard quality model.

Firesmith (2002) addresses quality concerns by introducing a formal quality model (Firesmith, 2004) which first splits quality into its component quality factors (aspects, attributes, or characteristics) and sub-factors (i.e., parts of quality factors). It uses system-specific quality criteria (descriptions) and metrics (means of measurement) to turn these general high-level quality factors and sub-factors into detailed and system-specific measurable descriptions that can be used to specify a quality aspect (and to determine if that quality aspect actually exists at a level equal to or greater than the minimum amount specified in a requirements specification). The approach emphasised the importance of integrating quality concerns into all RE activities.

Brito & Moreira (2004) propose a model to identify and specify quality requirements that crosscuts requirements, including their systematic integration into the functional description at an early stage of the software development process. The first task in the approach consists of identifying all requirements of a system and selecting from those the quality attributes relevant to the application domain and stakeholders. The second task is divided into two parts: specify functional requirements using UML use cases and describing quality attributes using special templates, identifying those that cut across functional requirements. The third task proposes a set of models to present the integration of crosscutting quality requirements and functional requirements.

Sommerville (2004) introduces a range of quality attributes to be considered during the quality planning process. The quality plan should identify the most important quality attributes that developers should focus as well as proposing a standard mechanism for quality assessment. Standards are utilized at every stage of systems development to assure quality such as IEEE 1058.1-1987 Standard for Software Project Management Plans(IEEE 1058.1-1987 Standard for Software Project Management Plans) during project planning and IEEE 830-1993 Recommended Practice for Software Requirements Specifications(IEEE 830-1993 Recommended Practice for Software Requirements Specifications) during requirements specification. The approach does not propose a specific modelling notation for quality requirements.

Bhatti (2005) proposes an approach that complements (Cysneiros et al., 2001; Moreira et al., 2002) approaches where quality is integrated with the functional requirements. The proposal incorporates ISO/IEC 9126-2 external metrics as extensions to UML (Jacobson, Booch, & Rumbaugh, 1999) diagrams (use case diagram, sequence diagram, and class diagrams); it is referred to Quality with UML (QWUML). The approach is quite comprehensive in dealing with quality requirements, however, the
elicited requirements are mapped directly to design constructs without mentioning how quality concerns are addressed in other RE activities such as negotiation, prioritization and validation.

In (Mead et al., 2004; Mead et al., 2008) a method for eliciting and prioritizing security requirements named SQUARE is proposed. The SQUARE method was designed as a complementary set of RE practices that are orthogonal to existing organization’s software engineering practices. Steps are proposed to be used alongside existing lifecycle models to ensure adequate support for security requirements engineering for security-critical projects. The SQUARE method is supported by the SQUARE tool to guide process users in the adoption and use of the process steps (SQUARE Tool, 2011).

The Goal-oriented Requirement Language (GRL, 2011) provides a visual notation for supporting goal-oriented modelling and reasoning relating to requirements with guidelines and techniques especially developed for addressing non-functional requirements. Quality requirements are represented as softgoals which represent condition in the universe of discourse that actors would like to achieve. The GRL language is supported by a tool that enables drawing GRL models. Reasoning techniques are not currently supported by the GRL tool. GRL is the modelling language of the i* (i-Star) framework (Yu, 1995).

Table 1 also highlights how the ElicitO framework and tool discussed in the remainder of this paper compares and contrasts to other non-functional requirements engineering approaches covered in this section. ElicitO follows the same principle of adopting the ISO/IEC 9126-2 quality model to standardise the terminology used to specify non-functional requirements and quality metrics as in (Firesmith, 2002; Bhatti, 2005), however, in contrast to (Firesmith, 2002; Bhatti, 2005), ElicitO also codifies the quality model standard into automated ontologies providing knowledge-based system functionality to support the semi-automation of RE activities.

ElicitO also provides tool-based support to guide process users in the adoption and use of RE process activities as proposed in (Mead et al., 2004; Mead et al., 2008), however, in contrast to (Mead et al., 2004; Mead et al., 2008), ElicitO supports a standardised quality terminology and ontological constructs to capture non-functional requirements throughout the RE activities. Knowledge-based reasoning techniques are also supported by ElicitO enabling the semi-automation of RE activities; e.g., conflict identification during prioritization and realism checking during requirements validation.

Similar to the approaches discussed in (Chung & Nixon, 95; Cysneiros et al, 2001; Brito & Moreira, 2004; GRL, 2011), ElicitO also provides constructs that enable the explicit description of relationships between functional and non-functional requirements and the integration of quality concerns across the early software engineering stages, however, in contrast to (Chung & Nixon, 95; Cysneiros et al, 2001; Brito & Moreira, 2004; GRL, 2011), ElicitO follows a quality model standard to provide a systematic terminology to address quality concerns.
3.1 Quality-Driven Requirements Re-Engineering and Quality-Driven Architectural Approaches

Other key contribution in the area of quality-driven software engineering is the work by Tahvildari, Kontogiannis, & Mylopoulos (2003) which proposes a re-engineering framework that allows for specific quality requirements (e.g., performance and maintainability) to be modelled as soft-goal graphs for the selection of transformational steps that need to be applied at the architectural (design) level of the legacy systems being re-engineered. Niemelä, & Immonen (2007) proposed a systematic transformational process that maps quality requirements into architectural models enabling tracing quality requirements and variability across the models of a product family architecture. The paper by Kim et al. (2009) proposes a quality-driven approach to the development of software architectures that applies reusable design abstractions called architectural tactics that model general architectural solutions for common issues relating to systems quality.

The above proposals illustrate important contributions towards treating quality as an integral part of the RE approach tackling the issue from an architectural or re-engineering perspective, however some limitations still persist: Tahvildari et al. (2003) address the context of software reengineering which narrows the applicability to systems already in existence. The QRF approach proposed by Niemelä & Immonen (2007) has the advantage of addressing a wide range of quality concerns across a product family and also providing detailed guidelines for mapping quality requirements into architectural views. The limitations of the QRF approach by Niemelä & Immonen (2007) and the architectural tactics approach by Kim et al. (2009) arise from the use of constructs based on design patterns (Helm et al., 1994) to represent quality requirements. Design patterns often encode solution space design knowledge meaningful to system developers and IT domain experts. Problem domain stakeholders such as business analysts and end users often have limited knowledge of advanced modelling constructs such as design patterns and will not be able to fully participate in the process of negotiation, prioritization and validation of the requirements artefacts. Another limitation of design patterns is the limited availability of knowledge based techniques based on reasoning mechanisms to semi-automate requirements prioritization and/or validation activities.

Another important research dimension of information, knowledge management and decision support systems quality stems from developing quality criteria, metrics and mechanisms to assess, maintain and improve the quality contained in the data, information and knowledge-based system architectures. Related work in this area include the research conducted by Rao & Osei-Bryson (2007) proposing a comprehensive taxonomy of quality dimensions to define, organise and measure quality in knowledge management systems; the research conducted by Vassiliades, Bouzeghoup & Quix (2000) proposing a methodology for mapping high level user-defined quality goals for a data warehouse into specific quality factors and dimensions stored in the data warehouse metadata, facilitating the development of warehouse
maintenance and evolution processes taking into account quality concerns. The paper by Jarke et al. (1999) also discusses how to extend the data warehouse architecture model to support quality models, also providing mathematical techniques for measuring and optimizing data warehouse quality factors. The approach is based on extending the metadata management dimension of the data warehouse architecture to support quality analysis and quality-driven design.


The ElicitO quality driven RE framework described in this paper leverages the concepts and constructs of the RE approaches described previously towards addressing quality as a core concern of the RE effort. In addition, ontology-based systems and knowledge-based reasoning techniques are applied to support several RE activities. A standard quality model is also used throughout the RE activities. Non-functional requirements are represented as fully specified quality requirements (with quality metrics explicitly articulated in connection with each quality requirement). The ElicitO framework (illustrated in Figure 2) encompasses the following building blocks:

- Domain ontologies (functional ontology and quality ontology) that serve as a knowledge model to guide RE activities. The ISO/IEC 9126 quality model (ISO/IEC 9126-1:2001 Software engineering --Product quality -- Part 1: Quality model) is adopted for developing the quality ontology helping to standardise the terminology used in the entire systems development effort.

- Process guidelines concerning how the functional domain and quality ontologies support each specific RE activity and the overall RE approach.

- Specifications of restricting relationships between functional domain entities and quality concepts, helping to constrain the range of valid requirement alternatives in requirements engineering.

- Specifications of inter-relationships between quality characteristics, sub-characteristics and attributes available in the quality model to enable the identification of consistency issues in requirements, and facilitate negotiation/prioritization discussions.

- Reasoning techniques applied in connection with the domain knowledge and quality knowledge representations to support the semi-automation of RE activities.
The ElicitO framework supports the following RE activities (Elicitation, Negotiation & Prioritization, Specification, and Validation) and it provides a foundation for capturing quality requirements by applying the knowledge model supported by the quality ontology. The framework highlights important quality concerns that need to be taken into account in systems development for an application domain, however, it is not prescriptive with regard to the choice of lifecycle model, the sequence of activities to be carried out in a specific project, and the portfolio of techniques used to carry out each RE activity (e.g., elicitation can be performed through a combination of techniques such as interviewing and stakeholder workshops). The framework is therefore complementary to existing lifecycle models, existing RE processes and techniques aimed at addressing functional requirements.

For the purpose of the paper we will focus on the activities of Requirements Elicitation and Requirements Negotiation & Prioritization (shaded activities in Figure 2). The proposed approach helps in addressing common challenges relating to quality requirements elicitation, negotiation and prioritization such as the problem of scope and the problem of understanding as discussed by Christel & Kang (1992). Problems of scope result from the fact that requirements analysts often have limited knowledge of the application domain which in turn limits their effectiveness in performing RE activities. Thus, our framework provides requirements analysts with baseline knowledge about quality characteristics and their impact on functional aspects relating to an application domain, heuristics on how
to deal with quality concerns in RE, and reasoning techniques to support semi-automation of activities. Overall, the framework enables a more systematic approach to tackling quality requirements and supports requirements analysts in the process of acquiring problem solving knowledge relevant in early stages of a new development project. This argument is justified by the framework’s emphasis on the domain ontologies and other knowledge management techniques supported to help requirements analysts in the process of understanding the key concepts in the problem space, the capability of reusing past requirements as a baseline to build the new set of requirements and also the use of quality model features to structure and define precise metrics for quality requirements. This claim is also empirically assessed by the case study discussed in Section 6.

Problems of understanding typically result from the fact that requirements engineering involves a variety of people’s expressed needs across different communities in contrast to an individual’s perspective. Problems of understanding can also happen in connection to quality requirements due to their subjective nature as discussed in Section 2. Hence, our approach also helps to promote better communication standards across RE activities by ensuring that quality requirements follow the common terminology represented in the ontology enabling a uniform treatment across, for example, different elicitation interviews conducted by different requirements analysts, thus, reducing the chances of missing out on important requirements or not addressing requirements in a consistent manner.

The proposed framework also supports the identification of potential conflicts between quality requirements towards facilitating requirements prioritization discussions, trade-off analysis, consistency and realism checking. These features are also enabled by the use of knowledge management techniques and precise metrics attached to quality requirements combined with the semi-automation of requirements engineering activities supported by the ElicitO tool. For example, during an elicitation session using the framework and tool to specify the availability quality requirement for a CRM application hosted in a cloud computing platform (Buyya et al. 2011), if a stakeholder requests a 100% availability figure for the hosted application, the domain ontologies in conjunction with the rule-based engine supporting the framework will identify a potential source of unrealism in the set of requirements. Also, if different values for the availability requirement are elicited during a project, the tool will flag a potential conflict within the set of requirements. Reasoning about potential conflicts in the set of requirements is also enabled by knowledge describing mutual dependencies between quality model characteristics and sub-characteristics. In order to achieve this it is important to understand the inter-relationships between quality characteristics, sub-characteristics and attributes of a quality model that will be propagated to and reflected in quality requirements. Table 2 shows some relationships between quality attributes at the quality sub-characteristics level. The quality attributes may cooperate (+), conflict (-), or have no effect on each other (0). This model was adapted from a range of contributions specialized in analyzing quality attribute relationships (Egyed & Grunbacher, 2004; Franch & Carvallo, 2003; Wiegers, 2003).

The values for each cell of the table were determined as follows:
• Identifying and analyzing general relationships between pairs of quality attributes previously published in the literature (e.g., Egyed & Grunbacher, 2004; Franch & Carvallo, 2003; Wiegers, 2003) and populating the cells with the results proposed by the validated research references.

• For pairs of attributes without previously published relationship analysis, the values for a cell were determined by articulating what it means to achieve each individual attribute by identifying the metrics by which it is commonly measured and/or observed and by determining the cause and effect relationships between the pairs through dependency analysis using conceptual graphs (Cox et al., 2001).

For illustration purposes, Table 2 does not cover all quality sub-characteristics listed in the ISO/IEC 9126 model and is limited to the common ones used in software projects: usability (Abran, Khelifi, Sury, & Seffah, 2003; Cysneiros, Werneck, & Kushniruk, 2005; Moraga, Calero, Paz, Diaz, & Piattini, 2005; Nielsen, 1999), security (Firesmith, 2003a) and efficiency (Yuan, 2005). The ElicitO ontology encodes a comprehensive inter-relationship table based on the complete standard.

<table>
<thead>
<tr>
<th>Quality Characteristic</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
</tr>
<tr>
<td>Accuracy</td>
<td>+</td>
</tr>
<tr>
<td>Interoperability</td>
<td>0</td>
</tr>
<tr>
<td>Security</td>
<td>+</td>
</tr>
<tr>
<td>Recoverability</td>
<td>0</td>
</tr>
<tr>
<td>Fault Tolerance</td>
<td>0</td>
</tr>
<tr>
<td>Learnability</td>
<td>0</td>
</tr>
<tr>
<td>Understandability</td>
<td>+</td>
</tr>
<tr>
<td>Atractiveness</td>
<td>0</td>
</tr>
<tr>
<td>Operability</td>
<td>+</td>
</tr>
<tr>
<td>Time Behaviour</td>
<td>0</td>
</tr>
<tr>
<td>Resource Utilization</td>
<td>-</td>
</tr>
</tbody>
</table>

(*) represents a positive effect, (−) represents negative effect, (0) represents no effect.

The ElicitO framework was developed with the goal of supporting the following practices in systems development projects:

1. **Design for Reuse** (Sutcliffe, 2002): the framework departs from a knowledge acquisition effort aimed at discovering key functional requirements arising from a vertical application domain such as education, banking and telecommunications, and also quality requirements that are relevant horizontally across vertical domains.

2. **Design by Reuse** (Sutcliffe, 2002): the framework attempts to capitalise from knowledge acquisition and management efforts, preserving and reusing transient requirements knowledge generated during software development projects to reduce incremental development costs of new projects.
3. *Flexibility and Complementarity*: the framework/tool is flexible to be used alongside established software process models such as the Waterfall, the Spiral model, RUP and Agile Methods. The choice and order of activities are not prescribed and the approach to problem solving can be either top-down, bottom-up or hybrid, depending on the nature of the problem.

4. *Continuous Requirements Engineering* (Pohl, 2010): the activities and knowledge management assets supported by the framework span the entire system lifecycle and are also applicable across product lines and project activities.

### 4.1 The Design of the Ontologies Underpinning the ElicitO Framework

In this section, we briefly discuss the ontologies developed to support the ElicitO framework and the ontology engineering process followed in the development. To support the quality-driven RE framework we chose to adopt the ISO/IEC 9126 as the baseline quality model for developing the foundations for integrating quality concerns into the RE approach. Other alternative quality models and quality management approaches assessed in the development of the ElicitO framework were the Boehm quality model (Boehm et al., 1978), and the McCall quality model (McCall et al., 1977). Both alternative models analysed have a similar hierarchical structure to the ISO/IEC 9126 quality model, however, prescribe a different set of first and second level quality characteristics and sub-characteristics when compared to ISO/IEC 9126. Al-Qutaish (2010) discusses and compares the key software quality models assessing the differences in capturing quality concerns at different layers of the model. The key benefit of building the ElicitO ontology using the ISO/IEC 9126 as the baseline quality model hierarchy stems from the terminological consensus and agreement reached by the ISO organisation and the continuous support and development of the quality model standard by ISO. It is important to acknowledge that the characteristics and sub-characteristics available from other quality models that are not supported by the ISO/IEC 9126 can also be incorporated into the ElicitO framework therefore we view other quality models as complementary from a quality ontology perspective. The detailed rationale for adopting ISO/IEC 9126 (*ISO/IEC 9126-1:2001 Software engineering --Product quality -- Part 1: Quality model*) as the baseline quality model was as follows:

- Covers a comprehensive range of quality attributes relating to relevant quality categories found in information technology projects (quality in use, external qualities, and internal qualities).
- The characteristics and sub-characteristics prescribed by the model provide a standardised terminology for specifying, measuring and evaluating system and software product quality. The set of quality characteristics also serve as a yardstick against which stated quality requirements can be compared for consistency and completeness (ISO, 2011).
- Is a widely recognized standard that has been employed across a range of software engineering applications/projects (Abran et al., 2003; Azuma, 2004; Bhatti, 2005; Carvallo, Franch, Grau, & Quer, 2004; Cote, Surya, Laporte, & Martin, 2005). The model has been continuously revised and updated since its original release in 1991 with a new revised standard (ISO/IEC 25010) released in March 2011 to succeed ISO/IEC 9126 (ISO, 2011).
Commercial and research tools are available to support the model (Carvallo, Franch, & Quer, 2004; QM tool, 2010).

To validate the applicability of the ElicitO framework we developed a case study based on the University Helpdesk domain ontology to support requirements engineering activities for a web portal development project. The knowledge captured is represented by two core ontologies: the quality ontology, which is based on software quality models representing reusable knowledge about different quality characteristics and sub-characteristics, quality metrics related to application domains, and the inter-relationships between quality characteristics. For example, the quality characteristic reliability encompasses the sub-characteristics maturity that can be measured by the level of accessibility anytime anywhere; fault tolerance measured by mean time between failure, and recoverability which can be measured by the mean time to repair. These quality characteristics are generic and can be applied to any vertical application domain. The functional ontology defines the key system context objects and relationships pertinent to the application’s domain of discourse including domain entities such as students, centres, services, labs, faculty, etc. and the properties or attributes attached to each entity (e.g., student-ID, name, address, etc.). Relationships between these entities are also specified in addition to the functions and tasks related to the University Helpdesk domain.

The knowledge representing the domain functions and explicitly defined quality characteristics and metrics associated to the domain was acquired via the ontology engineering process discussed in (Gomez-Perez et al., 2004) aimed at capturing and codifying the quality knowledge contained within the ISO/IEC 9126 standard, the functional domain knowledge relating to the key concepts and business functions for the university portal domain and defining the relationships and constraints relevant to the domain of discourse. This process involved researching technical reports describing the ISO/IEC 9126 quality standard, identifying key functions, quality factors and constraints that underpin educational portals published in the literature, e.g., (Raol et al. 2002; Masrek, 2007).

To acquire and structure the knowledge contained in the ontology we also used textbooks, standards, IT manuals of previously developed Helpdesk systems and interviews with domain experts (e.g. head of information services and five help desk operators with more than 5 years of experience each). Some classes and properties defined in other ontologies were also reused, for example, the knowledge encoded in the SUMO knowledge base (Group, 2003). Finally, the ontologies were implemented in Protégé (Protégé, 2011). Figure 3 shows a screenshot of the quality ontology in Protégé: The quality model is decomposed into three top level taxonomy classes as shown in Figure 3: (1) Representation of the relationships between quality characteristics; (2) representation of the ISO/9126 quality model including characteristics and quality sub-characteristics, and (3) representation of the quality metrics. An expanded snapshot of the quality ontology hierarchy is described in Appendix A1.
Another important feature of Protégé is the built-in reasoning capabilities allowing the development of constraints on how the quality and functional domain ontologies are linked and used. This is achieved through OWL expressions denoting domain restrictions/constraints (McGuinness & Harmelen, 2004). Restrictions are defined between the classes of the top level taxonomies described above to determine which metrics are associated with quality characteristics and sub-characteristics. In addition constraints are defined to represent the metrics related to the functional domain activities (top level taxonomy 4 in Figure 3). The restrictions specified below were used to constrain an individual that belongs to a class (e.g. helpdesk has-metric page_downloads_speed). There are two main sets of asserted conditions:

1. Metrics identification, in which all general metrics of the Quality Ontology are reused and other domain-specific metrics are identified. For example:

   - has_a_QualityMetric Num_of_links_per_page
   - has_a_QualityMetric Max_num_of_links_in_an_index_page
   - has_a_QualityMetric Avg_num_of_words_per_page
   - has_a_QualityMetric num_of_images_per_page
   - has_a_QualityMetric page_download_speed
   - has_a_QualityMetric Avg_num_of_colours_per_page

2. Assertions between quality metrics and domain functions. The example below shows the most important quality metrics associated to the activity (FAQ)

   - has_a_QualityMetric page_download_speed
   - has_a_QualityMetric Avg_num_of_colours_per_page
   - has_a_QualityMetric Num_of_links_per_page

The above metrics are defined once and are reusable across any other functional domain activity (e.g. navigation, search, help, etc.). The knowledge codified is also reusable across elicitation sessions enabling requirements analysts to configure a new set of requirements for a specific systems development scenario. In order to conduct conflict identification actions we incorporated conflict identification reasoning into the system by applying the rule-based reasoning framework supported by the Java Expert System Shell-JessTab (Eriksson, 2003). The production rules guide the user interface of the ElicitO tool that semi-automates the requirements engineering activities supported by the ElicitO framework.
Knowledge management techniques are applied to support requirements engineering activities helping to facilitate group collaboration, alleviate learning curve difficulties, guide requirements engineering tasks, and address interdisciplinary communication problems improving the effectiveness and efficiency of the requirements engineering process. The domain ontologies provide the capability to codify relevant knowledge that requirements analysts can leverage in the process of understanding the problem domain and facilitating agreement on the requirements artefacts during systems development.

Having knowledge templates of functional and non-functional requirements for a specific application domain helps to decrease the occurrence of problems of scope (e.g., finding all relevant requirements) and understanding (e.g., enabling that all non-functional requirements are uniformly and properly treated across different RE activities conducted by different requirements analysts), thus, reducing the chances of missing out important requirements or not treating requirements uniformly. In addition, the explicitly
defined quality characteristics and metrics associated to a domain help during requirements negotiation and prioritization since precise and non-ambiguous artefacts are developed. The requirements elicitation tool user interface was built using the Protégé API, which interacts with the objects stored in the knowledge base (domain and quality ontologies).

Figures 4 and 5 describe process templates specified using the SPEM notation (SPEM, 2010) for carrying out an RE activity supported by the ElicitO tool. The templates identify several tasks, techniques, guidelines and heuristics that can be used by requirements engineers to execute an activity and represent process suggestions that can be used by requirements engineers when tackling project tasks. The overall framework and tool are designed to be extensible with regard to process templates, enabling RE participants to incorporate new process templates that codify best practices in RE activity processes for specific problem/situation categories that are acquired upon successful organizational project experience. The activity process model and guidelines implemented by the tool to support requirements elicitation is illustrated in Figure 4.

![Figure 4. Requirements Elicitation Activity Process Template](image)

The requirements engineer performs requirements elicitation by enquiring stakeholders about their user/system requirements, the tool guides the elicitation activity by providing heuristics regarding functional and quality requirements relevant to a domain of discourse and offering stakeholders choices for metrics for a given quality requirement. The elicitation session execution supported by the framework and tool originates an initial set of requirements. This set can be further refined through iterations until the requirements have been completely captured. The tool will always suggest that a quality metric and a relevant domain are associated with quality requirements, helping to avoid underspecified non-functional requirements. Figure 5 illustrates the activity process model and guidelines to support the requirements prioritization task.
Figure 5. Requirements Prioritization Activity Process Template

Figure 6 shows the architecture of the ElicitO Tool. The tool has two main layers: the ontology layer where the quality and domain ontologies are encoded as OWL constructs in the Protégé database. It is also where rules are executed; the user interaction layer, which is decomposed into application and GUI components. The application component communicates with the ontology layer when querying for domain knowledge and the related quality attributes via the Protégé API. All query results and information displayed to the user is done via the graphical user interface. The underlying database for the storage of a requirements session is MySQL.

Table 3. Core Requirements Supported by the ElicitO Tool

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement Description</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR 1</td>
<td>Elicited requirements shall be represented as statements of intention with regard to the objectives, properties or use of the system</td>
<td>Essential(5)</td>
</tr>
<tr>
<td>FR 2</td>
<td>Elicited requirements shall be subject to input, storage, update, search, retrieval and reuse from ElicitO's main user interface</td>
<td>Essential(5)</td>
</tr>
<tr>
<td>FR 3</td>
<td>Elicited requirements stored and/or manipulated by the tool shall have the following attributes:</td>
<td>Essential(5)</td>
</tr>
</tbody>
</table>
A project identifier indicating the project the requirement belongs to
An elicitation session identifier indicating the elicitation session where the requirement was identified
A type descriptor indicating if the requirement is a functional, quality, constraint or other type of requirement (e.g., underspecified NFRs)
A priority attribute that can either be a qualitative field or a numeric field
A quality requirement shall have a metric field chosen from the range of values corresponding to the type of the attribute in the quality ontology
An owner or group of owners may be assigned to a requirement
A version number may be assigned to a requirement
An update date and time may also be assigned to a requirement

FR 4 The tool shall incorporate formal and extensible functional domain ontologies encoded in a knowledge base that will support the following features:

- Ontological descriptions of key entities and concepts in domains such as banking, insurance, education, etc.
- Ontological descriptions of key system functions arising in enterprise applications such as CRM, ERP and SCM supporting the key domains
- Ontological descriptions shall form a domain vocabulary and axioms that express knowledge that can be used to define new functional requirements
- Restrictions (constraints) should be defined to represent the metrics related to the functional domain activities

Essential(5)

FR 5 The tool shall incorporate a formal and extensible quality ontology encoded in a knowledge base that will support the following features:

- A complete quality taxonomy as described in the ISO/IEC 9126 quality model standard including quality characteristics, sub-characteristics and attributes available in the quality model
- Specifications of inter-relationships between quality characteristics, sub-characteristics and attributes available in the quality model to enable the identification of consistency issues in requirements, and facilitate negotiation/prioritization discussions
- Specifications of restricting relationships between functional domain entities and quality concepts, helping to constrain the range of valid requirement alternatives in requirements engineering
- Ontological descriptions shall form a domain vocabulary and axioms that express knowledge that can be used to define new quality requirements
- The quality attributes at the quality sub-characteristics level of the quality model may cooperate (+), conflict (-), or have no effect on each other (0). Knowledge should be encoded in the ontology to represent the interdependency relationships between quality sub-characteristics
- Restrictions (constraints) should be defined between the classes of the top level taxonomies of the quality model to determine which metrics are associated with quality characteristics and sub-characteristics
- Metrics associated with quality characteristics and sub-characteristics should have a name, unit of measurement (e.g., seconds) and description defining the metric

Essential(5)

FR 6 The tool shall incorporate the notion of an extensible Requirements Engineering Templates that will support the following features:

- Knowledge regarding how to perform a requirements engineering activity: steps, techniques and best practices shall be made available via the user interface during an elicitation session
- Detailed information describing the execution of elicitation techniques (e.g., interviews, workshops, focus groups) shall be made available via the user interface during an elicitation session
- Detailed information describing the execution of prioritization techniques (e.g., nominal group voting, weighted multi-votes, and pair wise comparison) shall be made available via the user interface during a prioritization session
- Automation of the process template steps enabling the tool to guide the RE activity, trigger reasoning techniques activating rules stored in the rule-based engine to support RE activities
- Instances of past requirements stored in the knowledge base relevant to the current problem domain shall be made available for reuse (input, storage, search, retrieval) via the user interface during an elicitation session

Essential(5)

The core functional requirements supported by the ElicitO tool are summarised in Table 3. Appendix A2 describes a selection of the key quality requirements underpinning ElicitO. We have omitted non-
essential requirements from the table to simplify the description and also omitted the rationale for each functional requirement to avoid repetition of the content described in Sections 4 and 5 of the paper which discuss the motivation and rationale underpinning the development of the ElicitO tool. Further details regarding the features supported by the ElicitO tool are described in (AlBalushi, 2008).


The ElicitO tool was used to support requirements engineering activities in connection with the Manchester Unity Web Project. The objective of this project was to enhance the current website of the university by adding extra features specified by different stakeholders’ views. The requirements were elicited and negotiated through focus group sessions. The participants from different university departments that attended the focus group sessions and provided feedback for what they would like to have available/implemented in the new system and what sort of problems they had encountered with the old system. An email was sent to all the participants of the focus group session to brief them with the objectives of the project. The requirements were gathered in three different focus group sessions with 12 members in each session. The project manager and one information systems analyst led the requirements engineering efforts. The first two sessions were conducted without the framework/tool support and the third session was conducted with ElicitO framework and tool support. The duration of each focus group session was two hours, the first hour was dedicated to requirements elicitation and the second hour was for requirements prioritization. The first focus group session was divided into two parts. In the first part, the project manager used the card sorting technique during the requirements elicitation exercise where participants were assigned 5 cards each and asked for what they wanted to have available on the website and what problems they came across using the website. However, the participants could only write down one entity/object in each card. The cards were collected and requirements relating to similar functional aspects were grouped (i.e., email, webmail, etc.). Each of the entity/object categories was dealt with individually and participants were asked to elaborate more on them. The three main categories which were the topics of main concern by the majority of the participants were (email, student’s timetable, reporting problems to helpdesk). For focus group session (1) the key requirements elicited during the elicitation phase are presented in Table 4. As can be observed in the table, the type of requirements elicited without tool support are very general, and vary from functional to underspecified non-functional requirements with very little attention to quality requirements and quality metrics (R2, R3, and R5). The second stage of the session was the requirements prioritization on which the requirements engineer asked the participants to rank the above requirements with either essential or nice to have. This triggered conflicts as every participant would attribute high voting importance for what they wanted disregarding any potential conflicts that his/her set of priorities could generate when taking into account the priorities of other participants. The participants were asked to vote on the importance of the requirements and the result of the voting was as illustrated in Table 5. Focus group 2 and the elicitation and prioritization sessions were conducted using a similar approach to focus group 1. Table 6 presents the key requirements elicited and their prioritization from focus group 2.
Table 4. Requirements captured without the tool support (Focus Group 1)

<table>
<thead>
<tr>
<th>User Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1 Provide information/pathway onto how to access web services (i.e. web mail,</td>
</tr>
<tr>
<td>network drive, etc.)</td>
</tr>
<tr>
<td>R2 FAQ should be clear and simple in answering users technical problems</td>
</tr>
<tr>
<td>R3 Make the websites among different schools consistent</td>
</tr>
<tr>
<td>R4 Provide campus map when required</td>
</tr>
<tr>
<td>R5 Make the university regulations and policies easy to access</td>
</tr>
<tr>
<td>R6 Make students’ user names accessible to faculty when using Blackboard (e-</td>
</tr>
<tr>
<td>learning) to register students</td>
</tr>
<tr>
<td>R7 Provide information on how to report a problem and to whom</td>
</tr>
<tr>
<td>R8 Provide information about exam timetables and venues</td>
</tr>
<tr>
<td>R9 Provide links to the outside world</td>
</tr>
<tr>
<td>R10 Highlight important events or alerts</td>
</tr>
<tr>
<td>R11 Update the staff directory frequently</td>
</tr>
</tbody>
</table>

Table 5. Requirements prioritized without the tool support (Focus Group 1)

<table>
<thead>
<tr>
<th>User Requirements</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1 Provide information/pathway onto how to access web services (i.e. web mail,</td>
<td></td>
</tr>
<tr>
<td>network drive, etc.)</td>
<td>Nice to have</td>
</tr>
<tr>
<td>R2 FAQ should be clear and simple in answering users technical problems</td>
<td>Essential</td>
</tr>
<tr>
<td>R3 Make the websites among different schools consistent</td>
<td>Nice to have</td>
</tr>
<tr>
<td>R4 Provide campus map when required</td>
<td>Essential</td>
</tr>
<tr>
<td>R5 Make the university regulations and policies easy to access</td>
<td>Essential</td>
</tr>
<tr>
<td>R6 Make students’ user names accessible to faculty when using Blackboard (e-</td>
<td></td>
</tr>
<tr>
<td>learning) to register students</td>
<td>Essential</td>
</tr>
<tr>
<td>R7 Provide information on how to report a problem and to whom</td>
<td>Nice to have</td>
</tr>
<tr>
<td>R8 Provide information about exam timetables and venues</td>
<td>Essential</td>
</tr>
<tr>
<td>R9 Provide links to the outside world</td>
<td>Nice to have</td>
</tr>
<tr>
<td>R10 Highlight important events or alerts</td>
<td>Nice to have</td>
</tr>
<tr>
<td>R11 Update the staff directory frequently</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.  Requirements elicitation and prioritization without tool support (Focus Group 2)

<table>
<thead>
<tr>
<th>User Requirements</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1 Make the website more appealing to international students.</td>
<td>Nice to have</td>
</tr>
<tr>
<td>R2 Unique and identified desktop for everyone</td>
<td>Nice to have</td>
</tr>
<tr>
<td>R3 Speed up faculty’s leave application requests</td>
<td>Essential</td>
</tr>
<tr>
<td>R4 Provide local news and weather forecast upon log in</td>
<td>Nice to have</td>
</tr>
<tr>
<td>R5 Update the users of their preference books which just arrived into the library</td>
<td>Essential</td>
</tr>
<tr>
<td>R6 Provide links to BBC or other local radio stations</td>
<td>Nice to have</td>
</tr>
<tr>
<td>R7 Provide a better way for group interaction on a project through the university</td>
<td></td>
</tr>
<tr>
<td>website</td>
<td>Essential</td>
</tr>
<tr>
<td>R8 Provide information about bus and train services</td>
<td>Nice to have</td>
</tr>
<tr>
<td>R9 Less downtimes on university webmail</td>
<td>Essential</td>
</tr>
</tbody>
</table>

In contrast to the unstructured and ad-hoc approach conducted during elicitation and prioritization sessions for focus groups 1 and 2, another focus group exercise was conducted, this time using the ElicitO framework and tool to underpin the requirements elicitation and prioritization activities. 12 randomly chosen participants of focus groups 1 and 2 were asked to join a third focus group. Participants in focus group 3 took part in this semi-automated session.

The rationale for randomly choosing participants that were involved in previous focus group sessions to also join focus group 3 was that they would have the experience of trying to elicit and prioritize non-
functional requirements without the ElicitO framework and tool support (non-automated approach). After participating in focus group session 3 (semi-automated elicitation and prioritization) the participants would also be able to compare the outcome of requirements elicitation and prioritization with and without framework/tool support, therefore being able to provide feedback on the effectiveness of ElicitO.

ElicitO supports the elicitation activity by retrieving from the ontology knowledge base relevant quality characteristics attached to the domain that will underpin potential questions that the requirements analyst will need to ask when eliciting quality aspects relevant to a particular functional characteristic of the domain of discourse. ElicitO highlights all the functional activities and tasks of the domain and their attached quality characteristics. Once a certain domain activity/task is selected in the tool (e.g. FAQ) relevant quality characteristics that can be discussed with stakeholders towards developing NFR specifications are presented, as illustrated in Figure 7(a). The add requirements button allows stakeholders to detail a quality requirement, in the given example, the quality characteristics (efficiency) and the associated sub-characteristics (time behaviour) related to the functional activity FAQ. ElicitO also allows the requirements analyst to ask more specific questions about the quality requirements through metrics such as (page download speed) as shown in Figure 7(b). Once all the requirements are identified, a list of detailed quality requirements is generated to be further discussed by the stakeholders.

For the discussion and prioritization activity, ElicitO offers a number of prioritization methods such as nominal group voting, weighted multi-votes, and pairwise comparison, and it is left to the requirements analyst to decide which method to use during the elicitation activities. For the discussion and prioritization activity conducted in the case study, the nominal group voting prioritization method was selected due to its simplicity, the familiarity of the project manager with the method and the non-hierarchical nature of the focus group session aimed at allowing all participants’ opinions to be taken into
account. Figures 8 (a), 8 (b) and 8(c) provide snapshots of the requirements elicited and prioritized by focus group 3. The following process steps were enacted:

1. All participants assessed the perceived return on value of each quality requirement in the set by using a Likert scale from (1-5) with the scale numbers having the following semantic meanings: 1-no value, 2-little value, 3-some value, 4-high value, 5-very high value.

2. For each requirement the mean value of all participants’ assessment was calculated and a priority was specified. The participants were also able to write a short justification for choosing a certain quality requirement over the other to facilitate prioritization and conflict resolution discussions.

3. The overall set of prioritized requirements was analyzed towards identifying and resolving potential conflicts through facilitated group discussions led by the project manager based on the justifications provided by the participants, nominal group voting on pairs of conflicting requirements and consensus building.

- Automated reasoning techniques were applied in the investigation of the set of prioritized requirements for potential conflicts through pressing the analyze requirements button illustrated at the bottom of Figure 8(a). This triggered the production rules specified using the Jess Rule system (Eriksson, 2003) checking the set of requirements for potential conflicts. A list of potential conflicting requirements was generated as highlighted in Figure 8(b).

- Pairs of conflicting requirements in the list were selected by the project manager to allow further discussion, nominal group voting and the assignment of a final consensual priority as illustrated in Figure 8(c).

The requirements prioritization task in the ElicitO framework can also be used in combination with other approaches and frameworks that support consensus building and group decision support (French et al., 2009; Bryson, 1996; Bryson, 1997; Ziemer, Sampaio & Stalhane, 2006). For example, the decision making procedure discussed in (Bryson, 1996) which proposes the generation of a numeric preference vector providing pairwise comparison information reflecting the group's belief in the relative importance of each object/alternative when compared to other objects/alternatives in the set. According to Bryson (1997) the individual and group preference vectors can be constructed as follows:

\[ x^t = (x^t_1, \ldots, x^t_N) \]

In the scenario of applying the decision support procedure in connection to the prioritization of a set of requirements, each object can be regarded as an individual requirement and pairwise comparisons in the vector will reflect relative preferences stated by each individual stakeholder between requirements in the document. The decision making procedure then iterates with the objective of reaching group consensus, including steps that support group discussions concerning the problem situation with group members taking into account their opinions along with supporting arguments for different points of view. The iteration stage of the decision support procedure stops after the calculated individual and group consensus indicators reach an acceptable level of consensus.
Overall the ElicitO framework and tool facilitated the requirements engineering activities by providing the functional domain concepts, quality attributes and precise metrics relevant to requirements analysts about a specific application domain via the knowledge encoded in the ontology which in turn decreased the problems of scope and problems of understanding during the requirements elicitation process. ElicitO also helped with the identification of potential conflicts among desired quality attributes and facilitated the process of assessing and negotiating trade-offs towards balancing the weights attached to quality requirements across the project increasing overall stakeholder satisfaction.
The summary of findings obtained from focus group sessions and other experiences of RE activities conducted with ElicitO support are described below:

- The knowledge encoded in the ontology helps to discover and formalize quality requirements and also prevents the propagation of underspecified non-functional requirements throughout the requirements engineering process helping to tackle problems of understanding caused by different interpretations of quality requirements. The outcome of focus group 3 provided evidence that despite having gained domain knowledge by participating in a previous focus group session, the knowledge management support system provided by ElicitO was still capable to support participants in eliciting and prioritizing previously unidentified non-functional requirements, and also helped in enhancing the quality of the specified requirements.
- The knowledge encoded in the ontology based on the ISO/IEC 9169 quality model supported requirements analysts in capturing a comprehensive set of non-functional requirements (breadth) with specific metrics associated with each quality requirement (depth). The terminology also enabled a uniform treatment of quality aspects across RE activities and elicitation sessions.
- The framework and tool tackled conflicting requirements early in the process enabling stakeholders to identify root causes of conflicts, negotiate and prioritize the requirements. As a result, shorter interaction cycles were associated with each RE activity.

It is also important to highlight that the gains in the number of non-functional requirements elicited and also in the quality of each individual non-functional requirement elicited (precision with regard to metrics) observed empirically through the experiment can also lead to some insights about the perceived impact of the tool and the process in supporting the requirements engineering effort:

- Tool Impact: The capability of eliciting and prioritizing fully specified non-functional requirements with clear associated metrics combined with the rule-based reasoning techniques incorporated in the tool are especially useful when conducting complex tasks relevant to the RE effort such as validating large sets of requirements. For example, if “stakeholder A” would like to set a maximal threshold for the “Page_download_speed” attribute value to 1 second combined with a simple command line user interface for a call centre application while “stakeholder B” would be willing to accept a page to download in 3 seconds, provided that a graphical web-based user interface is developed for the call centre application. The two goals have potential conflicts relating to the “Page_download_speed” attribute that can result in inconsistency problems relating to the overall project. When the requirements set for a project scales beyond the hundreds of requirements mark, a manual approach to detect inconsistency will struggle to identify conflicting requirements. The same argument is applicable to detecting unrealism issues and applying other validation checks on the requirements set.
Process Impact: the ElicitO framework aims to be complementary and orthogonal to existing RE processes, providing process templates that serve as guidelines to tackle requirements engineering tasks. In RE practice, the authors have experienced several real world project situations where requirements engineering efforts are undertaken by project team members with limited exposure to software engineering processes and/or formal training in requirements/software engineering. In these scenarios, the process templates supported by ElicitO provide valuable guidelines on how to approach the problem and ensure systematic treatment of quality concerns across the RE stages. The design philosophy underpinning ElicitO was to avoid prescribing which processes should be adopted by requirements engineers when there is already an established approach adopted by the organization to conduct RE. In the experiment described in this paper, however, the elicitation and prioritization processes supported by ElicitO helped in widening the portfolio of steps and techniques available to the requirements analysts that were driving the RE effort enabling the systematic treatment of quality requirements on an equal level of importance to functional requirements across the RE effort with a positive impact on the depth and breadth of quality requirements discovered. Future experiments can also be developed by tackling similar RE problem scenarios following different processes in combination with the ElicitO tool to attempt to isolate the process contribution in the RE results.

Another important finding was the speed up noticed by requirements engineers in the process of facing the initial learning curve challenges intrinsic to the domain of discourse. The knowledge encoded in the ontology provided a rich background about the key aspects involved in developing requirements for the new domain and also encoded tried and tested best practices relating to functional constraints and qualities relating to the domain of discourse. The following quotes obtained from participants of RE activities conducted using the ElicitO framework and tool help to highlight the benefits of the knowledge-guided and quality-driven RE approach:

"Using ElicitO, I was able to quickly identify key quality factors relevant to the application that I would not have considered during elicitation interviews…"

"The tool helped in highlighting the metrics relevant to a successful system…"

"Nice to find out that non-functional requirements can have a significant impact on systems quality…"

"Understanding that other project participants also have a stake in the system and that their priorities are not always aligned with my set of priorities helped me to think about what would be more important for the system as a whole…"

"Asking questions about how long a user should wait for a page download should be at the core of every web development project and yet, this was the first time I was actually asked to provide a precise figure to the page download speed issue…"

"Assigning numerical weights to requirements facilitates negotiating priorities more effectively than in situations where we have vague attributes such as essential and nice to have…"
ElicitO supports a requirements development framework that fosters reusability of requirements and also the possibility of developing the application either bottom-up, top-down or via a hybrid process lifecycle approach. This flexibility helps requirements engineers in configuring the process activities that need to be undertaken in each system development stage. The ontologies stored in ElicitO also benefit from every new project conducted by the organization. With time, the wealth of knowledge and best practices encoded help requirements analysts to be more effective and efficient when facing a new project and their tacit knowledge can be later codified helping to create a virtuous application engineering cycle that can be leveraged by the organization.

7. Conclusions and Further Work

Knowledge management techniques and theories are increasingly being applied to support requirements engineering activities towards addressing collaboration, learning curve difficulties and interdisciplinary communication problems arising in systems engineering, helping to improve the effectiveness and efficiency of the requirements engineering process (White, 2010). Ontology-based systems provide the capability to codify relevant knowledge that requirements analyst can use in the process of understanding the problem domain and facilitating agreement on the requirements artefacts during systems development. Within this context, this paper investigates the use of quality ontologies to support quality requirements elicitation and prioritization. The ontology implements the quality characteristics and metrics prescribed by the ISO/9126 quality model, providing a common vocabulary to address quality concerns across RE activities. The framework and the tool are aimed at empowering the requirements analysts with a knowledge repository that helps in the process of capturing and prioritizing precise non-functional requirements and establishing a uniform terminology for dealing with quality concerns across RE activities. We empirically validate how the ElicitO framework and tool was used to effectively support requirements elicitation and prioritization through a case study addressing an intranet portal development project at the University of Manchester. The results of the case study evaluation indicate that quality requirements can be better understood, negotiated about, precisely captured and documented.

The framework and tool are particularly suitable to software development projects where there is extensive opportunity for reusing requirements such as in enterprise application development (e.g., Customer Relationship Management, Enterprise Resource Planning, Supply Chain Management and Corporate Enterprise Portals) and where the set of non-functional requirements are large, with complex inter relationships, and with significant variations in perception of importance and technical feasibility within the stakeholders’ community. In this paper we have focussed on the elicitation and prioritization/negotiation activities for a University Intranet Portal case study. Other activities of RE can also be addressed using the ElicitO framework and tool as discussed in (AlBalushi, 2008; Sampaio et al., 2010). Ongoing work is investigating the merging and extension of the quality ontology with taxonomies for specifying NFRs in service-oriented environments (Galster & Buccherer, 2008), the integration of the ElicitO tool and framework with RE frameworks aimed at enhancing business and information
technology alignment (Lee-Klenz, Sampaio & Wood-Harper, 2010), the incorporation of reasoning mechanisms to perform automated requirements prioritization and triage (Duan, Laurent, Cleland-Huang & Kwiatkowski, 2009), and extensive validation checking across very large sets of requirements.

References


ISO (2011) URL:
Appendix A1: Expanded Snapshot of Quality Ontology Hierarchy
## Appendix A2: Key Quality Requirements Underpinning the ElicitO Tool

<table>
<thead>
<tr>
<th>Quality Characteristic</th>
<th>Requirement Description</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Functionality</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td>Each user must login into the system with a unique user name and a password at least 8 characters long involving at least one letter, at least one number and at least one symbol that is neither a letter nor a number. Password change requests must provide the old password and ensure that the new password is at least 8 characters long involving at least one letter, at least one number and at least one symbol that is neither a letter nor a number. The new password must be entered twice. After three unsuccessful login attempts the user account will be blocked and only a system administrator can reset the password. All interactions with the ElicitO tool should be stored in a system log.</td>
<td>ElicitO should provide security mechanisms that will enable its use across different projects by a variety of stakeholders, safeguarding confidential information and ensuring that only authorized stakeholders have access to project related information. Actions performed by users of the system should be logged to support auditing functions.</td>
</tr>
<tr>
<td><strong>Compliance</strong></td>
<td>The system shall observe privacy laws, data protection laws and adhere to organizational rules and regulations.</td>
<td>ElicitO should adhere to application related standards or conventions or regulations in laws and similar prescriptions.</td>
</tr>
<tr>
<td><strong>Usability</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Understandability</td>
<td>Density of the shown information should not be seen by the user as excessively disordered (i.e., more than three conceptual themes displayed). Users should be able to manipulate with the functions offered by the tool by leveraging common UI interaction patterns encountered in productivity tools. If codification colours are used, no more than six colours should be used, besides the addition of the black and white colours.</td>
<td>The user interface must be intuitive and users can easily interact with the functions offered by the tool.</td>
</tr>
<tr>
<td>Learnability</td>
<td>Context-sensitive help should be provided when the tasks have specific steps or contextual information.</td>
<td>The capability to offer help associated with the tool functions and features eases the learning curve effort required to use the tool.</td>
</tr>
<tr>
<td>Compliance</td>
<td>Guidelines provided by the ISO 9241 for user interface development should be followed.</td>
<td>Guidelines provided by the ISO 9241 standard allow the development of effective user interfaces that can be inspected and validated.</td>
</tr>
<tr>
<td><strong>Efficiency</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time behaviour</td>
<td>System should respond to basic users’ requests and functions in less than 2 seconds. When advanced functions are used, users should be warned of longer processing waiting times.</td>
<td>The capability of software product to provide appropriate performance in the execution of functions is essential to the uptake of the tool.</td>
</tr>
<tr>
<td>Resource utilization</td>
<td>System main memory utilization should be kept below 50 megabytes.</td>
<td>Tool can be used in low spec PCs and laptops.</td>
</tr>
</tbody>
</table>