An Ontological Interpretation of Non-Functional Requirements

Renata Guizzardi [a,1], Feng-Lin Li[b], Alexander Borgida[c],
Giancarlo Guizzardi[a], Jennifer Horkoff[b], and John Mylopoulos[b]
[a] *Federal University of Espírito Santo* (*UFES*), *Vitória, Brazil*
[b] *University of Trento, Trento, Italy*
[c] *Rutgers University, New Brunswick, USA*

Abstract. Non-functional requirements (NFRs) have been the focus of research in Requirements Engineering (RE) for more than 20 years. Despite this attention, their ontological nature is still an open question, thereby hampering efforts to develop concepts, tools and techniques for eliciting, modeling, and analyzing them, in order to produce a specification for a system-to-be. In this paper, we propose to treat NFRs as *qualities*, based on definitions of the UFO foundational ontology. Furthermore, based on these ontological definitions, we provide guidelines for distinguishing between non-functional and functional requirements, and sketch a syntax of a specification language that can be used for capturing NFRs.

Keywords. Non-functional requirements, foundational ontology, UFO, qualities.

Introduction

Requirements Engineering (RE) is the field of Software Engineering (SE) concerned with the elicitation, modeling and analysis of stakeholder needs and wants, for purposes of deriving a specification for a system-to-be. In much of RE research and practice, these needs and wants are captured in terms of *functional requirements* (*FRs*) and *non-functional requirements* (NFRs). While functional requirements specify *what* the software system must do, non-functional requirements specify, among others, *how well* the system shall perform its functions [1]. For example, "Users shall be able to *withdraw money* from their accounts" is a functional requirement for an ATM machine. On the other hand, issues concerning *how long it takes* for a user to withdraw money and *how well* the information of the account holder is protected are non-functional requirements. It is well documented in the RE literature that NFRs are a frequent cause of software development failure or malfunction; see, for instance the woes of the new US HealthCare ("Obamacare") website, most of which related to an inability to handle the heavy workload experienced after its launch[2].

Non-functional requirements have been the focus of research in Requirements Engineering (RE) for decades. One can refer to the two important surveys on NFRs [23, 24] to form an opinion on the state of the art on the topic. The NFR framework [2], first proposed in the early 90s, provides a simple qualitative framework for modeling NFRs

[1] Renata Guizzardi, rguizzardi@inf.ufes.br
[2] http://www.cnn.com/2013/10/29/politics/obamacare-hearing/

as *softgoals*, i.e., goals with no clear-cut criteria for satisfaction. These can be analyzed using qualitative reasoning techniques. However, applying the NFR framework in practice has shown that softgoals are useful for modeling early requirements elicited from stakeholders, both functional (e.g., *collect real-time traffic information*) and non-functional (e.g., *the system should respond quickly*).

This begs again the question: what are NFRs? There have been some recent initiatives to formalize RE foundations with the use of ontologies [3,4]. In particular, Jureta et al. [4] propose a core ontology for requirements based on DOLCE [5]. We consider this work as the baseline for the ontological interpretation presented in this paper, as some of their concepts (e.g. *softgoal*, *quality constraint*) are consistent with our view on NFRs and related notions. We believe, however, that this ontology is not appropriate for explaining all ontological phenomena required to effectively defining and dealing with NFRs. Therefore, the objective of this paper is an exercise in ontological analysis and conceptual clarification. We aim at spotting what is lacking in this previous ontological account of requirements, and provide the ontological foundations we deem necessary to capture a richer set of ontological phenomena related to NFRs.

In an earlier, short position paper [6], we have presented the preliminary idea of NFRs as requirements over qualifies based on DOLCE. In this work, we greatly extend this initial proposal, and use instead the Unified Foundational Ontology (UFO) [7]. The main motivations behind this decision is the fact that, besides being compatible with DOLCE, at least in the ontology fragment relevant for this work, UFO offers a more complete set of categories to cover some important aspects of the domain we target, especially regarding the analysis of *quality spaces, situations* and *goals*.

The main contributions of this paper are as follows:

- Providing an ontological interpretation of non-functional requirements as requirements over as qualities, grounded on UFO [7,8,9].
- Providing ontological guidelines for distinguishing between non-functional and functional requirements.
- Positioning NFRs relative to other ontological concepts, providing support for the development of an ontology-based syntax to specify NFRs.
- Describing a sound approach for the analysis of the satisfaction of gradable NFRs.

The rest of the paper is organized as follows: Section 1 discusses the core ontology for requirements, also identifying and exemplifying what is missing; Sections 2 and 3 describe the core contributions of this paper, focusing on the ontological interpretation of NFRs based on the UFO foundational ontology; Section 4 elaborates on some practical implications of this interpretation, presenting the aforementioned guidelines, syntax and method that practitioners may use to capture NFRs; Section 5 presents the final consideration and future work.

1. A Core Ontology for Requirements

An initial conceptualization for RE was offered by Jackson and Zave [3] nearly two decades ago, founded on three basic concepts: *requirement*, *specification* and *domain assumption*. Based on this characterization, the classical "*requirements problem*" is defined as follows: given a set of requirements *R*, and a set of domain assumptions *D*, find a set of specifications S *consistent with* D such that $D, S \vdash R$. For example, to satisfy the requirement "*make online payment*" (*R*), a software/service needs to support the function "*pay with credit card*" (*S*) under the (implicit) domain assumption of "*having a credit card with available credits*" (*D*).

On observing that this characterization does not allow partial fulfillment of some requirements and leaves out important notions, Jureta et al. [10,4] have proposed a revised Core Ontology for RE (aka CORE) based on goal-oriented requirements engineering (GORE), which is founded on the premise that requirements are stakeholder goals. The revised account starts from the premise that requirements elicitation consists of communication acts, and is grounded on the DOLCE ontology [5]. CORE distinguishes between non-functional and functional requirements using *qualities* as in DOLCE: (i) A requirement *r* that refers to a quality *q* is non-functional; further, if the quality type *QT* of *q* has an acknowledged shared quality space *QS* among the stakeholders, then *r* is a *quality constraint*; while if the corresponding *QS* is not shared among the RE participants (hence *r* is vague for agreed success), then *r* is a *softgoal*. (ii) If *r* does not refer to a quality, and refers to a *perdurant*, then *r* is a *functional goal*. In addition, stakeholders' preferences over requirements are captured as *evaluations*.

Accordingly, Jureta et al. re-define the "*requirements problem*" as finding *S* such that $D, S \mid\sim G, Q, A^{\succ}$, where *Q* is a set of softgoals and/or quality constraints, *G* is a set of functional goals, and $A^{\succ}$ is a relation indicating preferences among combinations of *D, G* and *Q* instances. *S* contains specifications in the form of tasks to be carried out, as well as AND-refinements of goals into subgoals. The authors also argue that the entailment relation should be non-monotonic (hence, the use of the symbol "$\mid\sim$" instead of "$\vdash$") because newly added domain assumptions or specifications could defeat previously valid conclusions.

Since its proposal in 2008, this core ontology has enjoyed considerable attention, and has served as the baseline of new research directions in RE [11,12]. However, in our experience, its handling NFRs has deficiencies:

1. It is unable to capture a class of requirements that refer to neither qualities nor perdurants, but endurants. E.g., "*the user interface shall have a standard menu button for navigation*", where "*menu button*" is an *endurant*. As a result, this FR cannot fit into any of the categories of CORE.
2. It is difficult to capture requirements that are vague for success but do not refer to qualities. For example, requirements such as "*attract customers*" and "*increase sales*" refer to perdurants rather than qualities, and are accordingly classified as functional goals. However, this conclusion contradicts Jureta et al's claim that "*functional goals are Boolean, i.e., true or false*", since these examples, like softgoals, have no clear-cut criteria for success.
3. We have discovered requirements that refer to both qualities and functions. For example, although we can classify the requirement "*the system shall collect real-time traffic information*" as a softgoal according to the core ontology ("real-time", i.e., timeliness, is a quality of traffic information), we are still left with the question "is it only an NFR?" It seems to be a combination of functional and non-functional requirements, which can eventually be refined into distinct sub-goals.

2. Ontological Foundations

In general, we are in line with Jureta et al. [4] that NFRs are requirements on qualities. In this section, we go deeper to capture the ontological meaning of quality and use it to interpret NFRs. For that, it is important to review some of the concepts defined in UFO [7], the adopted foundational ontology in this work.

Over the years, UFO has been successfully employed to provide ontological semantics and methodological guidelines, as well as for analyzing and redesigning modeling languages, standards and reference models in domains ranging from Bioinformatics, Enterprise Modeling, Telecommunications, Software Engineering, among others[3].

We present here only a fragment of the UFO containing the categories that are germane for the purposes of this article (Figure 1). Moreover, we illustrate these categories and some contextually relevant relations with UML (Unified Modeling Language) diagrams. These diagrams express typed relations (represented by lines with a reading direction pointed by >) connecting categories (represented as rectangles), cardinality constraints for these relations, subsumption constraints (represented by open-headed arrows connecting a sub-category to its subsuming super-category), as well as *disjointness* constraints relating sub-categories with the same super-category, meaning that these sub-categories do not have common instances. Of course, these diagrams are used here primarily for visualization. The reader interested in an in-depth discussion and formal characterization of UFO is referred to [7,8,9,13].

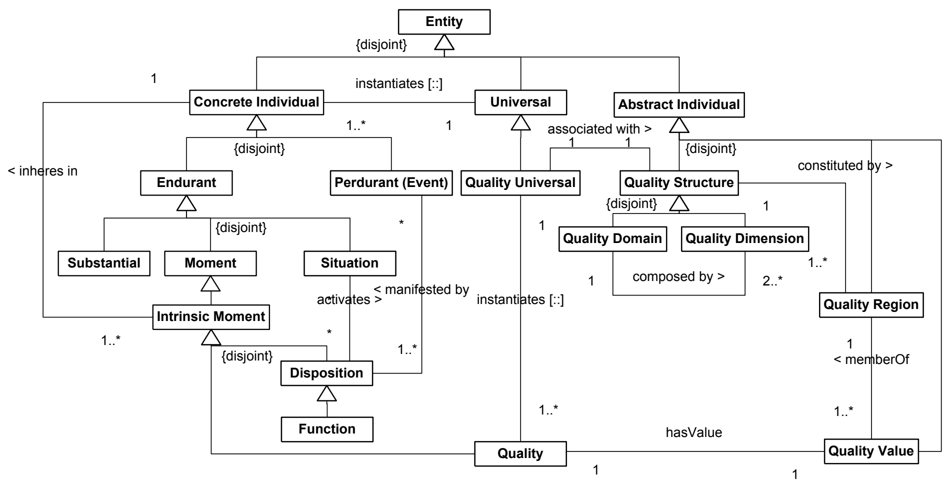

Figure 1 A fragment of UFO representing basic categories (including qualities are related notions)

We start by distinguishing between individuals and universals. ***Individuals*** are entities that exist in reality possessing a unique identity, while ***universals*** are patterns of features that are repeatable in a number of different individuals. A concrete individual can be either an endurant or a perdurant. ***Endurants***[4] do not have temporal parts, and persist in time while keeping their identity (e.g. a person and the color of an apple) while ***perdurants*** (also referred to as ***events***) are composed of temporal parts (e.g. storm, heart attack, trip). ***Substantials*** are existentially independent endurants (e.g. a

[3] See http://nemo.inf.ufes.br/en/publications for publications on the different UFO applications

[4] By convention, if the word "universal" is not part of a term, then the term is assumed to refer to a particular.

person or a car). ***Moments***, in contrast, are existentially dependent on other individuals, inhering in these individuals (e.g. someone's headache and the color of a car). ***Inherence*** (symbolized as *inh*) is a type of non-reflexive, asymmetric and anti-transitive type of functional existential dependent relation connecting a moment to its bearer. We focus here on ***intrinsic moments***, i.e., moments that are dependent on one single individual (e.g., a headache, a color, the disposition of a magnet to attract metallic material). Most distinctions made for individuals, *mutatis mutandis*, also apply to universals; thus, we have the counterparts: ***substantial universal, moment universal*** and ***intrinsic moment universal***. As shown in Figure 1, a ***quality universal*** is defined as an intrinsic moment universal that is associated to a ***quality structure,*** which can be either a ***quality dimension*** or a ***quality domain***.

UFO's notions of quality structure, quality dimension and quality domain are based on the work of Gardenfors [14,15]. According to this work, for all perceivable or conceivable quality universal, there is an associated quality structure in human cognition. For example, height, mass, and response time are associated with one-dimensional structures; other quality universals such as color, taste, and usability are represented by several dimensions. For instance, color can be represented in terms of the dimensions of hue, saturation and brightness; usability in RE is composed of learnability, operability, accessibility, among other dimensions. Moreover, Gardenfors differentiates integral and separable quality dimensions: "*certain quality dimensions are integral in the sense that one cannot assign an object a value on one dimension without giving it a value on the other. For example, an object cannot be given a hue without giving it a brightness value. Dimensions that are not integral are said to be separable, as for example the size and hue dimensions.*" [14]. A quality domain is a set of integral dimensions that are separable from all other dimensions [14]. ***A quality region*** is a convex region C of a quality structure (i.e. either a dimension or a domain); C is convex iff: for all pairs of points (x, y) in C, all points between x and y are also in C [14]. The value of a quality individual can be represented as a point in a quality domain. UFO names this point a ***quality value*** (which DOLCE calls "quale" [5]). For example, a color quality c of an apple a takes its value in a three-dimensional quality domain constituted of the quality dimensions hue, saturation and brightness. It is relevant to highlight that in UFO both physical (e.g., color, height, shape) and nominal quality types (e.g., social security number, the economic value of an asset) are sorts of quality universals and, hence, are associated with quality structures. Figure 1 also shows that a ***quality*** instantiates a quality universal and it has a quality value in a quality structure associated with that quality universal. Moreover, as an intrinsic moment, a quality inheres in individuals. Finally, in pace with DOLCE, if a quality universal is associated to a quality domain, its instances bear sub-qualities that take values in each of the dimensions of that domain. For instance, the color of an individual apple is itself a bearer for individual qualities of hue, saturation and brightness.

Besides quality, we include here the category ***functions*** as a sub-category of intrinsic moments, i.e., as existentially dependent entities. Moreover, we consider functions as particular types of dispositions (capacities, capabilities) exhibited by an entity [8]. Functions (and dispositions, in general); are potential (realizable) property instances manifested through the occurrence of an event that happens if a situation (state of the world) of a particular kind obtains. The occurrence of this event, in turn, brings about a certain situation in the world [16].

In UFO, an ***agent*** is a substantial that creates ***actions***, perceives events and to which we can ascribe mental states (i.e., ***intentional moments***). Intentionality in UFO is intended in a broader sense than “intending something”. Rather, it refers to the capacity of some properties of individuals to refer to possible situations of reality. Thus, “intending something” is a specific type of intentionality termed intention in UFO. ***Intentions*** are intentional moments that represent an internal commitment of the agent to act towards that will. A ***goal*** is a proposition, and more specifically, the propositional content of an intention. Furthermore, a goal is satisfied by a situation iff the situation makes true the proposition expressed by that goal.

3. Non-Functional Requirements and Related Ontological Concepts

This section (including its subsections) aims at explaining the concepts depicted in Figure 2. As already seen, an intention has a goal as propositional content. Goals are specialized into ***NFRs*** (also named ***quality goals***) and ***functional requirements*** (***FRs***). We take that FRs refers to a function (a capability, capacity) that has the potential to manifest certain behavior in particular situations. In other words, a FR has a propositional content that requires of a certain entity to bear a function of a given type. So, contra Jureta et al. [4], we take that FRs refer to perdurants only indirectly, i.e., by referring to a function, which being a disposition is realizable through the occurrence of perdurants of a given type. For example, the “*keyword search*” function of an online shop will be manifested by a process (perdurant) of matching between an input keyword and the list of keywords in the system in a particular situation (when the keyword is given and the search button is clicked) and brings about a certain effect (the matched product will be displayed).

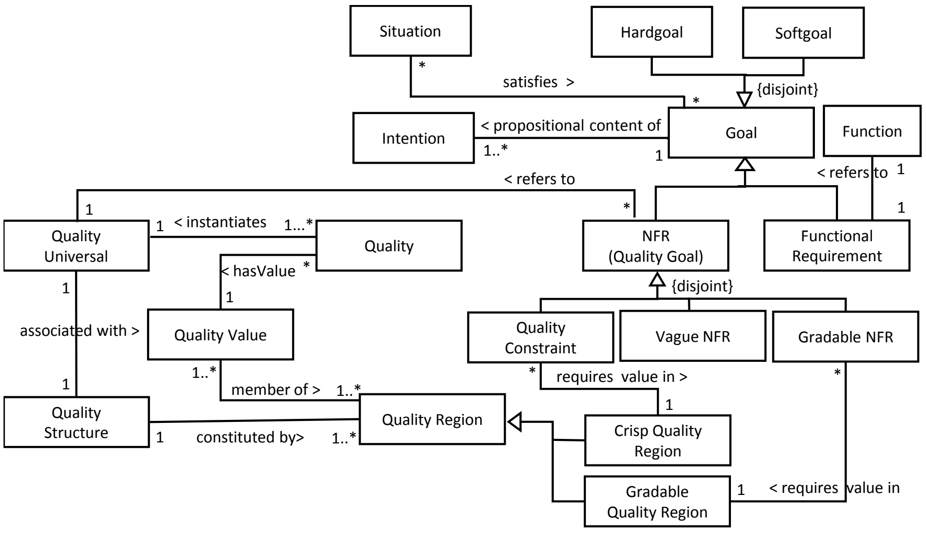

Figure 2. Non-functional Requirements and related concepts

Conversely to FRs, NFRs’ propositional content refers to qualities, i.e., which requires a certain entity to bear a quality or exemplify a quality of a given type. To be more specific, we treat NFRs as requirements that require qualities to take values in particular quality regions in their corresponding quality structures. In general, quality regions can be either crisp (e.g., 0 ~ 5 *seconds*) or vague (e.g., *fast*), hence NFRs (quality goals) can be accordingly crisp or vague. We identify those NFRs that specify crisp quality regions and define them as *quality constraints* (QCs).

As NFRs are goals referring to qualities, one must understand which quality it is and in which individual it inheres. Take, for instance, the requirement "*The user interface must have a standard format*". The quality in this case is *format,* while the bearer is *user interface; standard* is a particular region in the interface format quality structure. Sometimes, the quality may not be explicit, e.g. *"The product should conform to the American Disabilities Act"*, in which case the quality is *regulatory compliance* and the bearer is *the product*.

3.1. NFRs vs. Softgoals

In our view, the distinction between NFRs and FRs is orthogonal to that of ***hardgoals*** and ***softgoals***. Traditionally, hardgoals and softgoals are informally differentiated depending on whether they have clear-cut criteria for success (the former) or not (the latter) [17]. Here, we take the following stance on these concepts, capturing their distinction as follows: a hardgoal is a proposition that is objectively satisfied by a given set of situations. In contrast, a softgoal is an initial and temporary vague expression of intention before the goal at hand is properly refined. As such, we are not able to determine *a priori* the set of situations that satisfy a softgoal, i.e., its truthmaking conditions.

For example, "*design the system's main menu*" is a high-level goal and it can be considered vague (and thus modeled as a softgoal). In addition to capturing high-level vague goals, softgoals are also useful when capturing and analyzing vague NFRs. As previously mentioned, an NFR is a goal referring to a quality (type). In this case, a ***softgoal*** is thus a proposition referring to a vague quality region, meaning that although we are aware that such region exists in the quality structure, we do not know where the boundaries of that region exactly are. For instance, consider that the aforementioned goal is now refined into an NFR: "*The menu buttons must have standard length and width"*. At first, the system's stakeholders and analyst may have difficulties in mapping *standard* to a specific region in the interface format quality structure. As the analysis moves forward, vague NFRs are continuously refined and operationalized (as detailed in the sub-sections to come), and hence such vagueness generally disappears.

The NFR framework [2] models NFRs as softgoals that are not clear for success, and on the other side, the CORE ontology [4] treats functional requirements (FRs) as hardgoals i.e. goals whose satisfaction have a determinate agent-independent truth-value. However, we claim that these definitions of NFR/FR and hardgoal/softgoal are, in fact, orthogonal, allowing us to identify NFRs that are in fact hardgoals as well as FRs that are softgoals. Moreover, the categories of NFR and FR are not disjoint, indicating that a requirement can fall into both categories. See Section 4 for some interesting examples that illustrate the usefulness of this orthogonality principle.

3.2. Refining NFRs

Making NFRs measurable often involves *refinement* operations, where a requirement *r* is refined into *r'* such that *r'* is more precise and/or measurable. Often, refinement consists in conceptually deconstructing the NFR's referred qualities. One way to do this is to identify qualities that inhere in the quality associated with an NFR. For example, a security quality can be refined in its sub-qualities (i.e., qualities inhering in it) *confidentiality*, *integrity* and *availability* [18]. A second way to do this refinement is to identify whether this quality is a resultant quality, which can be conceptually reduced to qualities of parts of the bearer of the original quality. For instance, we could refine the previously stated requirement *"The user interface must have standard format"* by reducing the quality at hand in terms of qualities of parts of the bearer (the interface). Since the interface is usually composed of buttons, fields, icons etc., "*The menu buttons must have standard format*" illustrates a possible refinement of the previous requirement.

As an example of refining an NFR by decomposing its quality, format with respect to buttons may, for instance, be decomposed into "*size*", "*shape*", and "*color*". Considering "*size*", it may be further decomposed into "*height*" and "*width*". Hence, a conjunct of a further refinement is *"The menu buttons must have standard height and width"*.

3.3. Operationalizing NFRs

To make NFRs measurable, we need to *operationalize* them by constraining the referred qualities so that these qualities take values in crisp quality regions (i.e., absolutely defined regions). That is, operationalizing NFRs as quality constraints (QCs).

We may operationalize the NFR "*The menu buttons must have standard length and width"* by defining the quality constraint "*The menu buttons must have height 0.75 cm and width 1.75 cm*". While in this example, qualities are constrained to have specific quality values, in other cases, operationalization of an NFR may concern a region, as in "*The search functionality must be efficient*", operationalized by "*The search results shall be returned within 30 seconds after the user enters search criteria*". In our framework, the value "efficient" here is associated to a region in the time quality dimension, comprehending quality values from 0 to 30 seconds.

Note that terms such as "*efficient*" and "*low*" may refer to different quality regions, depending on the type of the quality bearer. For instance, take the requirement *"Learning to operate the login functionality must be efficient"*. This NFR may be operationalized by "*The user should learn how to operate the login functionality within 3 minutes*". Thus "*efficient*" for learning the login functionality and for returning search results (previous example) may map to different regions in the time quality dimension.

3.4. Gradable NFRs

Consider the satisfaction of a quality constraint (QC) as a function, which results in "1" (if the QC is satisfied) and "0" (if unsatisfied). The key point to determine the satisfaction of a QC is to understand if the measured or perceived quality value is a member of the region to which the QC is associated. If yes, the satisfaction function returns "1" and otherwise, it returns "0". For example, considering *"The search results shall be returned within 30 seconds after the user enters search criteria"* (constraint defined region: 0 < search time ≤ 30 *sec.*), if the runtime measurement of a search duration results in 25 seconds, the QC is satisfied; if the result is 32 seconds, then it is not.

However, this may be too strong a condition. Perhaps a 32 second response is "good enough". In many cases, "good enough" performance is sufficient, i.e., *degree of fulfillment* of a QC is what matters, rather than black-or-white fulfillment. Thus, in order to capture the intended semantics of many NFRs communicated by requirement engineers, the satisfaction function should not be a binary function but should instead return a graded satisfaction value in the interval between "0" and "1". To account for such phenomena, we propose the definition of ***gradable NFRs***, based on the conceptual space theory [14] and some of its recent extensions of the original theory as proposed in [19][20].

In Gardenfors [14], the definition of quality region in the quality structure is based on a combination of prototype theory [21] and the mathematical technique of Voronoi diagrams [22]. Prototype theory claims that some instances of a concept are more representative or typical than others (thus termed prototypes). Thus, the prototype of a quality is nothing other than a point in its quality structure. Creating Voronoi diagrams is a very general technique and may be applied to any metrical space in order to divide the space into cells. Each cell has a center and it contains all and only those points that lie no closer to the center of any other cell than to its own center (please see Figure 3) (A) for an illustration). Combining prototype theory and this technique consists in defining Voronoi diagrams by using the qualities prototypes as their central points.

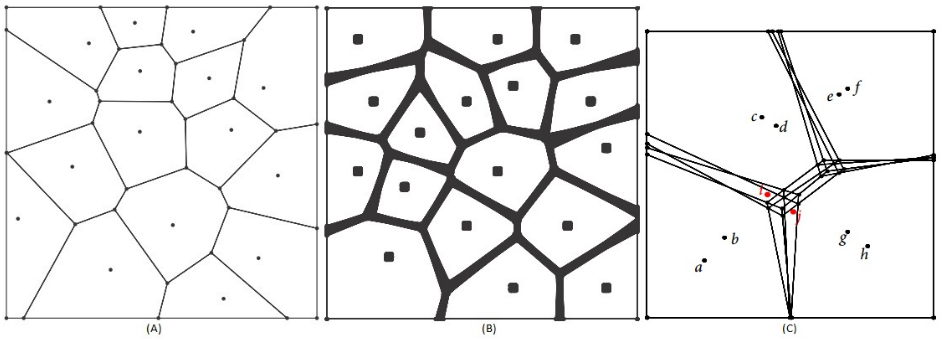

Figure 3. Two dimension (A) Voronoi Diagram and (B) Collated Voronoi Diagram (adapted from [19])

To overcome the limitations in dealing with gradable concepts, Douven et al. [19] extend this approach by assuming that conceptual spaces may contain *prototypical regions* rather than isolated prototypical points. Using these prototypical regions, they develop a technique to generate what they call *Collated Voronoi Diagrams*. This technique starts by considering the set of all possible selections of exactly one prototypical point from each prototypical region. Each element of this set (a vector of prototypical points coming from different regions) can be used to generate a different diagram of the quality structure *S*. Let us call the set of all these diagrams V_S. The Collated Voronoi Diagram can be generated by projecting all elements in V_S onto each other (thus overlaying the resulting diagrams). Figure 3 (B) depicts this idea. In the resulting diagram, the regions created by the tessellation have thicker borders than in the traditional Voronoi diagrams. We term these regions ***Gradable Regions*** and the borders that are shared by these regions *Borderline regions*.

Decock and Douven [20] take one step further in this extended theory. As they point out, Figure 3 (B) is misleading in making one think that the transition from a crisp region (i.e., one of the white polygons/polyhedrons in the figure) to a borderline region is itself sharp. According to them, this interpretation would be phenomenologically incorrect and we should treat this transition as also being a smooth one. This idea is illustrated by the authors with the following example: suppose we have four prototypical regions, each consisting of two points *{a,b}, {c,d}, {e,f}* and *{g,h},* each representing a prototype region of a particular concept. Now, suppose we generate the V_S in the manner previously explained (i.e., V_S would contain $2^4 = 16$ members in this case).

The authors then use the resulting set V_S to provide an interpretation to the idea of *graded membership function* of a particular point to a concept *X*: a point in the quality structure *S* belongs to concept value *X* to degree *D* that equals the number of members of V_S that locate the point in the cell associated with one of the two prototypical points of the region associated with *X* divided by the total number of members of V_S. Figure 3 (C) illustrates this idea. In this example, the point *i* belongs to the concept associated with region $\{a,b\}$ to the degree 0.5, since 8 of the 16 members of V_S locate *i* in the cell associated with a member of $\{a,b\}$. By the same reasoning, *j*, belongs to that concept by a degree of 0.25.

By adopting this view, we define QCs as follows: a QC is an NFR that specifies a crisp region *R* in a quality structure *S*. As such, the satisfaction of a QC for a bearer *B* is defined by the membership (or lack thereof) of the proper quality value of *B* in the region *R*. A ***gradable NFR*** instead refers to a gradable region *R'* of quality structure *S'* (which is a member of a set of prototypical regions associated with *S'*). As such, the satisfaction of an NFR to a certain degree is defined by the graded membership of the proper quality value of *B* in region *R'*.

4. Practical Implications

We show in this section how the ontological interpretation of NFRs (1) provides operational guidelines for distinguishing between non-functional and functional requirements (Subsection 4.1); (2) supports the development of an ontology-based requirements specification language (Subsection 4.2); and (3) enables the analysis of the satisfaction of gradable NFRs (Subsection 4.3).

4.1. Distinguishing between non-functional and functional requirements

The first benefit of capturing the ontological meaning of NFRs is conceptual clarification. As we will demonstrate, the ontological interpretation proposed here enables us to clearly distinguish between non-functional and functional requirements.

Despite many efforts devoted to NFRs, the question "*what are NFRs?*" is still debated [23]. On treating NFRs, there are two general approaches in the literature. One takes the stance that functional requirements describe what a system should do, while non-functional ones specify how well the system should perform its functions [1]. The other is to treat everything that is not a functional requirement (i.e., not related to what a system should do) as a non-functional one (i.e., as a sort of dispersive class defined by negation) [24]. However, when put into practice, both criteria are deficient. For instance, how does one classify Ex. 1 below, which specifies a function ("*support*") that will not be performed by the system but by an external agent ("*the corporate support center*")? One may treat it as an NFR by following the second criterion, but this is conceptually incorrect. In fact, Ex. 1 will be classified as a FR in our proposal (because it requires a function of an entity in the system-to-be ecosystem).

*Ex.*1*: The product shall be supported using the corporate support center.*
*Ex.*2*: The system shall have a standard navigation button for navigation.*
*Ex.*3*: The system shall help administrators to analyze failures/exceptions.*
*Ex.*4*: The transportation system shall collect real-time traffic information.*

Jureta et al. [4] have made the first step in grounding this distinction on qualities in the foundational ontology DOLCE [5]. As we have discussed in Section 1, their requirements ontology still has deficiencies: it is not able to categorize requirements like Ex. 2 (referring to neither qualities nor perdurants), Ex. 3 (referring to a perdurant but being vague for success) and Ex. 4 (referring to both perdurants and qualities).

In our framework, ***if a requirement refers to a particular quality universal, then it is non-functional***; ***if it refers to a function*** (***in the ontological sense***)***, then it is functional***. Adopting this guideline, we can easily classify Ex. 2 as functional, because it concerns a function that is manifested by the navigation button. Note that the distinction between NFR and FR is orthogonal to the one between hardgoal and softgoal. Hence, an NFR can have a clear satisfaction criterion while a FR can also be vague. For instance, Ex. 3 is a FR but one that has a *subjective criterion of satisfaction* (see section 3.1). Moreover, the classes of NFRs and FRs are not mutually exclusive. For example, Ex. 4 specifies a desired function "*collect traffic information*" but also refers to a quality (*timeliness*) of "*collecting traffic information*".

We have evaluated our framework by applying it to the PROMISE requirements dataset [25], which includes 370 NFRs crossing 15 software projects. Using our ontological classification of requirements, we identified 187 NFRs, 52 FRs, and 61 requirements that constitute a combination of FRs and NFRs (the remaining 70 ones are identified as function constraints or domain assumptions). For example, "*The website shall prevent the input of malicious data*", originally labeled as a security NFR, should actually be a FR since it refers to a "*prevent*" function. The result suggests that our framework is effective to help distinguishing between NFRs and FRs used in practice. For more details on the evaluation, interested readers can refer to our companion paper [26].

4.2. Representing Non-functional Requirements

A key benefit of understanding the ontological foundation of NFRs is that it provides support for designing requirement modeling languages. Recall from Figure 1 that UFO uses two fundamental predicates involving qualities: (1) *inheres*(*q#*, *b#*) relates a particular quality *q#* to its bearer *b#* (by convention we use '#' to indicate individuals); (2) *hasValue*(*q#*, *v*) relates a particular quality *q#* to the quality value *v* it currently has. We rephrase this by defining a single, higher-order function *hasQV* as shown in Eq. 1 below, in which *QUS* is a set of all quality universals, e.g., *Color*, *Cost*, and *Size*; *BearerT* is a bearer type; *QVT* is a quality value type, and ':' is used to give type signatures of functions. This function takes as arguments a quality universal *QU* (e.g., *Cost*) which is of type *QUS* (denoted as *QU*::*QUS*), an individual bearer *b#*::*BearerT* (e.g., *trip#*), and returns the quality value *v*::*QVT* of a particular quality *q#*::*QU* (e.g., *cost#*) that inheres in *b#*. Based on this function, an NFR that refers to a single individual having quality value in region QRG is written as Eq. 2, with a formal characterization of its intended semantics provided by Eq. 3; this says that *b#* shall bear a quality *q#* of type *QU*, and *q#* shall have a quality value *v* be in the desired region *QRG*. For example, the requirement "*the cost of trip# should be low*" can be now captured as Eq. 4,

$$hasQV : QUS \rightarrow BearerT \rightarrow QVT \qquad (1)$$

$$hasQV\,(QU)(b\#) :< QRG \qquad (2)$$

$$\exists q\#::QU,\ inheres(q\#, b\#) \wedge [\forall v\ hasValue(q\#, v) \rightarrow in(v, QRG)] \qquad (3)$$

$$hasQV\ (Cost)(trip\#) :< low \tag{4}$$

Some requirements may concern qualities of a *set* of individuals that are instances of a type (e.g., all the trips from Berlin to Paris in July, 2013, or all the executions of a software function). For this purpose, we apply *hasQV*(*QU*)(*b#*) to the set of individuals (i.e., bearers) to get their quality values. By function overloading, we define a new function with the same name *hasQV* as shown in Eq. 5, in which $\wp(BearerT)$, a short-hand of ***PowerSet***(*BearerT*), is the type of a set of individual bearers, each of which is of type *BearerT*, $\wp(QVT)$ is the type of a set of quality values. Accordingly, an NFR that refers to a set of individuals is expressed as Eq. 6, of which the intended semantics is shown in Eq. 7. Hence a requirement like "*the cost of all trips from A to B at period T should be low*" can be represented as in Eq. 8, where *Trip'* is a subtype of *Trip* (*denoting the trips from A to B at period T*).

$$hasQV : QUS \rightarrow \wp(BearerT) \rightarrow \wp(QVT) \tag{5}$$

$$hasQV\ (QU)(BearerT) :< QRG \tag{6}$$

$$\forall b\#::BearT,\ \forall q\#::QU,\ inheres(q\#, b\#) \rightarrow [\forall v\ hasValue(q\#, v) \rightarrow in(v, QRG)] \tag{7}$$

$$hasQV\ (Cost)(Trip') :< low \tag{8}$$

These formulations characterize the ontological foundation of NFRs, and provide us with the semantics that a requirements modeling language (RML) should capture. We introduce "*QU*(*BearerT*) :< *QRG*" as an abbreviation for "*hasQV*(*QU*)(*BearerT*) :< *QRG*", and use ':=' to assign names to expressions, as in Ex.9.

$$\begin{gathered} NFR\#1 := Processing\ time\ (keyword\ search) :< less\ than\ 30\ seconds \\ NFR\#2 := Cost\ (\{trip\#\}) :< low \end{gathered} \tag{9}$$

Here NFR#1 requires each manifestation of keyword search to take less than 30 seconds while NFR#2 requires a particular trip to have low cost. Moreover, this syntax can be further extended to capture more complex NFRs that concern universality (e.g., *the processing time of keyword search shall be less than 30 seconds 90% of the time*) and agreement (e.g., *80% of the users report the interface is appealing*), which are common in practice. We further explore representation of such complex NFRs using a compositional language in our companion paper [26].

4.3. The satisfaction of gradable NFRs

Specifying gradable NFRs can be quite useful in practice since, in many cases (as exemplified in Section 3.4), it may be enough to "almost" reach the satisfaction of an NFR. Thus, another practical application of our ontological interpretation to RE regards the analysis of the satisfaction of gradable NFRs. This can be accomplished by using the graded membership calculation described in Section 3.4.

For instance, suppose that the associated quality value region *low* of the gradable NFR NFR#1 in Eq. 9 is represented by two prototype values 500€ and 700€. Similarly, we can use 800€ and 1000€, and 1200€ and 1500€ to represent the region *medium* and *high*. Given the three prototype regions, the *Vs* will include 8 simple diagrams. Now if we have a cost value as 740€, then we will have 6 out of 8 diagrams classify it to the region *low*. Thus, that *NFR#*1 is satisfied to a degree of 0.75. Interested readers can refer to the calculation details available online[5]. The interesting point here is that we

[5] http://goo.gl/xXZ24E

can use prototype values to represent a region, and then adopt (collated) Voronoi diagrams to reason about the graded membership without the need of inventing made-up numbers as that in fuzzy logic [27].

5. Conclusions

We propose an ontology-based interpretation of NFRs by adopting and applying the UFO ontology. While doing that, we analyze how our proposal compares with and extends the CORE requirement ontology [4], which to the best our knowledge, provides the only existing ontological account of NFRs.

In a nutshell, we treat both NFRs and FRs as goals and differentiate them by claiming that the former refer to qualities while the latter are functions. From an ontological viewpoint, qualities and functions belong to different ontological sub-categories of intrinsic moments. A quality is a type of categorical property that is manifested whenever it exists and which is directly associated to a quality structure. A function, in contrast, is a type of dispositional property that is only manifested under certain circumstances and via the execution of an event. Moreover, as dispositions, functions do not have values which are directly associated to quality structures.

Besides presenting the aforementioned ontological interpretations, this paper discusses how these interpretations differ from existing RE approaches to non-functional versus functional requirements, as well as implications for RE practices.

For future work, we intend to conduct more evaluations by means of case studies. Moreover, we also aim at developing our initial requirements analysis methodology [6], by grounding it on the proposed ontology.

Acknowledgment. This research has been funded by the ERC advanced grant 267856 “Lucretius: Foundations for Software Evolution”, unfolding during the period of April 2011 - March 2016.

References

[1] B. Paech and D. Kerkow, “Non-functional requirements engineering-quality is essential,” in *10th International Workshop on Requirments Engineering Foundation for Software Quality*, 2004.

[2] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, “Non-functional Requirements,” *Softw. Eng.*, 2000.

[3] M. Jackson and P. Zave, “Deriving specifications from requirements: an example,” in *Software Engineering, 1995. ICSE 1995. 17th International Conference on*, 1995, pp. 15–15.

[4] I. J. Jureta, J. Mylopoulos, and S. Faulkner, “A core ontology for requirements,” *Appl. Ontol.*, vol. 4, no. 3, pp. 169–244, 2009.

[5] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari, “Ontology Library,” *WonderWeb Deliv. D18*, 2003.

[6] F.-L. Li, J. Horkoff, J. Mylopoulos, L. Liu, and A. Borgida, “Non-Functional Requirements Revisited,” in *CEUR Proceedings of the 6th International i* Workshop (iStar 2013)*, Valencia, Spain, 2013, pp. 109–114.

[7] G. Guizzardi, *Ontological foundations for structural conceptual models*. CTIT, Centre for Telematics and Information Technology, 2005.

[8] G. Guizzardi, G. Wagner, R. de Almeida Falbo, R. S. Guizzardi, and J. P. A. Almeida, “Towards Ontological Foundations for the Conceptual Modeling of Events,” in *Conceptual Modeling*, Springer, 2013, pp. 327–341.

[9] G. Guizzardi, R. de Almeida Falbo, and R. S. Guizzardi, “Grounding Software Domain Ontologies in the Unified Foundational Ontology (UFO): The case of the ODE Software Process Ontology.,” in *CIbSE*, 2008, pp. 127–140.

[10] I. J. Jureta, J. Mylopoulos, and S. Faulkner, “Revisiting the core ontology and problem in requirements engineering,” in *International Requirements Engineering, 2008. RE’08. 16th IEEE*, 2008, pp. 71–80.
[11] I. J. Jureta, A. Borgida, N. A. Ernst, and J. Mylopoulos, “Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling,” in *Requirements Engineering Conference (RE), 2010 18th IEEE International*, 2010, pp. 115–124.
[12] S. Liaskos, S. A. McIlraith, S. Sohrabi, and J. Mylopoulos, “Integrating preferences into goal models for requirements engineering,” in *Requirements Engineering Conference (RE), 2010 18th IEEE International*, 2010, pp. 135–144.
[13] A. B. Benevides, G. Guizzardi, B. F. B. Braga, and J. P. A. Almeida, “Validating Modal Aspects of OntoUML Conceptual Models Using Automatically Generated Visual World Structures.,” *J UCS*, vol. 16, no. 20, pp. 2904–2933, 2010.
[14] P. Gärdenfors, *Conceptual spaces: The geometry of thought*. MIT press, 2004.
[15] P. Gärdenfors, “How to make the semantic web more semantic,” in *Formal Ontology in Information Systems*, 2004, pp. 19–36.
[16] R. Hoehndorf, J. Kelso, and H. Herre, “Contributions to the formal ontology of functions and dispositions: An application of non-monotonic reasoning,” *ICBO*, p. 173, 2009.
[17] E. Yu, “Modeling strategic relationships for process reengineering,” *Soc. Model. Requir. Eng.*, vol. 11, 2011.
[18] ISO/IEC 25010, “Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models,” 2011.
[19] I. Douven, L. Decock, R. Dietz, and P. Égré, “Vagueness: A conceptual spaces approach,” *J. Philos. Log.*, vol. 42, no. 1, pp. 137–160, 2013.
[20] L. Decock and I. Douven, “What Is Graded Membership?,” *Noûs*, 2012.
[21] E. Rosch, “Cognitive representations of semantic categories.,” *J. Exp. Psychol. Gen.*, vol. 104, no. 3, p. 192, 1975.
[22] F. Aurenhammer, “Voronoi diagrams—a survey of a fundamental geometric data structure,” *ACM Comput. Surv. CSUR*, vol. 23, no. 3, pp. 345–405, 1991.
[23] M. Glinz, “On non-functional requirements,” in *Requirements Engineering Conference, 2007. RE’07. 15th IEEE International*, 2007, pp. 21–26.
[24] L. Chung and J. do Prado Leite, “On non-functional requirements in software engineering,” in *Conceptual Modeling: Foundations and Applications*, 2009, pp. 363–379.
[25] T. Menzies, B. Caglayan, H. Zhimin, K. Ekrem, K. Joe, P. Fayola, and T. Burak, “The PROMISE Repository of empirical software engineering data,” Jun-2012. [Online]. Available: http://promisedata.googlecode.com.
[26] F.-L. Li, J. Horkoff, A. Borgida, R. S. S. Guizzardi, G. Guizzardi, J. Mylopoulos, and L. Liu, “Non-Functional Requirements as Qualities, with a Spice of Ontology,” in *Requirements Engineering (RE), 22nd International Conference*, 2014.
[27] L. Baresi, L. Pasquale, and P. Spoletini, “Fuzzy goals for requirements-driven adaptation,” in *Requirements Engineering Conference (RE), 2010 18th IEEE International*, 2010, pp. 125–134.