A More Expressive Softgoal Conceptualization for Quality Requirements Analysis

Ivan J. Jureta[1], Stéphane Faulkner[1], and Pierre-Yves Schobbens[2]

[1] Information Management Research Unit (IMRU), University of Namur,
8 Rempart de la Vierge, B-5000 Namur, Belgium
iju@info.fundp.ac.be, stephane.faulkner@fundp.ac.be
[2] Institut d'Informatique, University of Namur,
8 Rempart de la Vierge, B-5000 Namur, Belgium
pys@info.fundp.ac.be

Abstract. Initial software quality requirements tend to be imprecise, subjective, idealistic, and context-specific. An extended characterization of the common Softgoal concept is proposed for representing and reasoning about such requirements during the early stages of the requirements engineering process. The types of information often implicitly contained in a Softgoal instance are highlighted to allow richer requirements to be obtained. On the basis of the revisited conceptual foundations, guidelines are suggested as to the techniques that need to be present in requirements modeling approaches that aim to employ the given Softgoal conceptualization.

1 Dealing with Software Quality Requirements

Ensuring the quality of software has become a major issue in software engineering research and practice since the 1970s [5]. As increasingly complex software plays a critical role in business, comprehensive and precise methods and tools are needed to create software products and services that are safe, dependable, and efficient [26].

Software quality is defined by the International Organization for Standardization [12] as the totality of features and characteristics of a software product that bear on its ability to satisfy stated or implied needs. Ensuring the quality of software therefore amounts to making sure that software behavior is in line with stated and implied needs.

It is widely acknowledged that quality needs to be taken into account early in the software development process [8,30,19]. Quality requires specifying stated and implied needs. Approaches focusing on ensuring quality during the development process by guiding functional requirements specification decisions by quality considerations, so that the latter justify the former, are termed process-oriented. In contrast, product-oriented approaches (e.g., [11,13]) evaluate the quality of already developed software products, and are particularly relevant for, e.g., component selection [2].

D.W. Embley, A. Olivé, and S. Ram (Eds.): ER 2006, LNCS 4215, pp. 281–295, 2006.

Although a large body of work deals with quality assurance in a process-oriented manner, a non-negligible part of it relies on the usual Softgoal concept for the representation and reasoning about quality-related requirements. In doing so, procedural aspects of methods for dealing with quality during requirements engineering (RE) activities have been considerably developed, while conceptual foundations have not evolved in a notable manner. In particular, a more extensive view on the conceptualization and formalisms for representing and using quality requirements while taking into account their multi-facetted nature has not been proposed yet. We need to deal with requirements that are not only implicit, but also subjective, context-specific, imprecise, and ordered by preference.

The work presented in this paper is a step towards a more profound understanding of requirements that are expressed usually in requirements goal diagrams (such as, e.g., i^* [32]) as instances of the Softgoal concept. Overall, instances of the original Softgoal concept are seen as frequently containing information that is, not only subjective and context-specific (as assumed in the original definition), but also imprecise and involving preferences of the stakeholder who suggested the requirements modeled as the given softgoal. It is therefore suggested that the Softgoal is a multi-facetted concept that requires specialized techniques for dealing with its additional facets. This paper thus proves useful both in terms of advancing the understanding of a key concept in the RE modeling field, and in arguing that additional considerations need be taken into account when a RE method or framework that employs the Softgoal concept is being constructed and applied. Finally, the reader will undoubtedly notice that the discussion below is independent of a particular RE framework, which supports our arguments regarding the applicability of this discussion to many (at least goal-oriented) RE methods.

The paper is organized as follows. Part of the literature on the treatment of quality requirements, applicable to the discussion in this paper is first overviewed. The bulk of the paper, which discusses and revisits the original Softgoal conceptualization is then presented. A set of general guidelines on the characteristics of RE methods aiming to employ the suggested conceptualization are presented. Finally, conclusions are summarized and directions for future work are identified.

2 Related Work

To facilitate the discussion of related work, Table 1 gives a classification of process-oriented approaches. Formal approaches rely on formal notations such as temporal or fuzzy logic to specify nonfunctional requirements in a precise way, while semi-formal provide structured notations (unrelated to mathematical logic) that are used mainly to organize information about nonfunctional requirements. Qualitative approaches traditionally evaluate the degree of quality requirements satisfaction using subjective qualitative characterizations. In contrast, quantitative techniques focus on estimating the probability of failure of quality-related goals [19], or use informal measures of the degree to which software properties contribute to specific qualities [1]. Decision on placing some approaches in

qualitative or quantitative category is based on the methods described in the cited papers; e.g., adapting a qualitative approach to use quantitative methods remains possible, but is not discussed in the literature.

Table 1. A classification of process-oriented approaches proposed in related work for ensuring quality during the software development process

	Qualitative	*Quantitative*
Formal	[18,20,31]	[19,23]
Informal	[8,22,27,17]	[9,1]

The NFR framework [22,8] has been the first to propose the concept of Softgoal in the RE context (the original concept that is specialized for RE in NFR has a longer history—e.g., [28]) and a process for dealing with nonfunctional requirements. In NFR, Softgoals describe quality requirements in very abstract terms. They are related with contribution links to support qualitative reasoning about the degree to which alternative software properties satisfy the desired qualities. Their intuitiveness and ease of use have led to their integration in goal-oriented RE (GORE) frameworks: *i** [32], Tropos [6], GRL [21], and REF [11]. However, the Softgoal concept remains informally defined and used. Many GORE frameworks that have adopted NFR suffer from the same symptoms, as few extend the NFR Softgoal conceptualization. [9] adds a probabilistic layer to study the impact of requirements change on quality satisfaction. Others [1] use multi-criteria decision techniques to select among alternative software architectures.

Formal approaches have been proposed to provide systematic support when semi-formal techniques are considered inadequate. Instead of Softgoals, [19] is focused on software goals that are precise, but cannot be completely achieved by the software (i.e., they are idealistic). Quality variables are associated with all goals that can only be partially satisfied and objective functions are defined over these variables to indicate ideal software behavior. Quality variables seem to be metrics that measure performance of the behavior specified by the goal to which the variables are associated. Based on a sample of software operation, probabilities of satisfying a goal can be estimated—these probability values indicate the degree to which the goal is satisfied. Imprecise requirements are treated with fuzzy logic in [18,20,31,23]. While fuzzy logic may be an interesting approach for formalizing imprecise requirements, it has been discussed mostly in isolation from typical RE activities, although it is not obvious how such formalisms can be integrated within existing, more extensive frameworks.

Discussion. Expressive formalisms such as fuzzy logic have unfortunately been discussed somewhat separately from confirmed GORE methodologies and frameworks, making the us of the techniques proposed in [18,20,31,23] impractical and difficult. The informed reader will also note that fuzzy logic is merely one among

many approaches to imprecision. While quality requirements are indeed imprecise, they do have other characteristics that need to be accounted for during representation and reasoning. Partial satisfaction, extensively discussed in [19] is relevant, but is discussed with focus on precise goals. It should also be noted that the NFR approach and other RE frameworks using the Softgoal concept fail to address situations in which systematic and formal treatment is required, even though the growing criticality of software increases the need for rigor.

The research presented in the remainder of this paper starts from a hypothesis that the informally defined Softgoal concept, extensively used in NFR, can be more valuable if its facets: subjectivity, context-specificity, idealism, impreciseness, and preference are characterized explicitly. Such a characterization will allow both a systematic treatment of the stated facets, and a closer integration of the proposed Softgoal analysis approach and later RE activities, such as functional goal specification. In this respect, the proposed conceptualization draw on the extensive body of related work to provide a more integrative view on the representation and manipulation of software quality information.

3 The Softgoal Concept Revisited

Softgoals provided by the stakeholders at the outset of the RE process, such as "the software should be fast", can be characterized as *imprecise*, *subjective*, *context-specific*, and *ideal*. Imprecision stems essentially from the inability to specify what "fast" mean, so that they could be measured. Subjectivity results from the fact that two people can evaluate the same software as being fast to a different extent. Context-specificity further entails that "fast", or "usable", "maintainable", "adaptable" (standard qualities of software [8]) will have a different meaning for each project. Finally, implicit preference information is hidden behind terms such as "fast software": various measures can be taken, but low values will be preferred. All of the above characteristics need to be taken into account to deal systematically with quality requirements. To use the Softgoal concept to represent and reason about such requirements, it is necessary to make its traditional definition more expressive and precise. The choice of the Softgoal concept is based on the illustrated usefulness of the underlying goal concept in RE activities, such as elicitation, elaboration, structuring, specification, analysis, negotiation, documentation, and modification of requirements [29].

3.1 Functional and Nonfunctional Goals vs. Hardgoals and Softgoals

A goal can be broadly defined as a constraint on software behavior that is desired by stakeholders involved in the software development project (e.g., [10]). Among the many proposed goal taxonomies (for an overview, see [30]), two are particularly relevant for quality requirements modeling. *Functional* goals have been used to represent services that the software is expected to deliver (i.e., *what* the software does), whereas *nonfunctional* goals refer to quality requirements that the software needs to satisfy while delivering the services (i.e., *how*

the software provides services; e.g., securely, safely, rapidly, etc.). While it is common to equate nonfunctional goals and softgoals (e.g., [22]), it is suggested that *softgoals* belong to another taxonomy, in which they are opposed to *hardgoals* [30]. Although softgoal satisfaction cannot be established in a clear-cut sense [22], the satisfaction of a hardgoal is objective in that it can be established using (formal) verification techniques [10]. Consequently, there are: (i) *functional hardgoals*, which are objective goals about services that software needs to deliver (e.g., "whenever an e-mail marked as important arrives, the user is informed with a pop-up window and a sound"); (ii) *nonfunctional hardgoals* which describe objective criteria for how the services are to be delivered (e.g., "the user should be informed about important e-mail arrival within 1sec"); (iii) *functional softgoals* describe imprecisely stated software services (e.g., "the user should be informed when an e-mail marked as important arrives"); and finally, (iv) *nonfunctional softgoals* characterize imprecise statements for how a service is to be delivered (e.g., "the user should be informed rapidly about the arrival of an e-mail marked as important"). It is likely that statements about the needs that the software is to satisfy will be closer to nonfunctional softgoals than to functional hardgoals at the outset of the RE phase of software development. Notice that there is a large gap in precision between nonfunctional softgoals and functional hardgoals example: the former says nothing on how the user is to be informed, while the latter gives a specific context (the e-mail reader software) and process (e-mail arrives, pop-up is displayed, and a sound is played). Having clarified the informal meaning of Softgoal in relation to goal types, we proceed to its characterization.

3.2 Characterizing Softgoals

The traditional view of softgoals [21] focuses essentially on the subjectivity facet:

> "A softgoal is similar to a (hard) goal except that the criteria for whether a softgoal is achieved are not clear-cut and a priori."

A definition proposed in the REF framework [11] adds details:

> "For a soft goal [...] it is up to the goal originator [i.e., the agent wishing goal achievement], or to an agreement between the involved agents, to decide when the goal is considered to have been achieved [...]. In comparison to hard goals, soft goals can be highly subjective and strictly related to a particular context; they enable the analysts to highlight quality issues (e.g., the concept of a 'fast computer') from the outset, making explicit the semantics assigned to them by the stakeholders."

Softgoals therefore involve subjectivity because of a lack of objective achievement criteria, and the responsibility for evaluating their achievement falls on stakeholders. Notice that it is imprecise to say that quality considerations can mainly be modeled with softgoals, since quality refers to software behavior that can be both objectively and subjectively evaluated. However, there is more to quality requirements than the current softgoal conceptualization allows representing

and reasoning about. Consider a simple example of a quality requirement often encountered in practice: "the software should be fast". By examining the stated and implied information contained in this statement, a notation can be proposed to model the various softgoal facets.

Subjectivity. Since many stakeholders (i.e., parties being influenced by, or having an influence on the development project) are likely to be involved in the RE phase of software development, the specificity of each these parties' views on the software, the development process, and the environment in which the software will operate needs to be accounted for. The usefulness of separating stakeholders' concerns and the use of adapted, different notations is now widely accepted. Such multi-perspective requirements require techniques for making individual views consistent either by reconciling requirements specifications written in different specification languages (e.g., [25]), or written different styles, terminology, etc. (e.g., [14]). The resulting heterogeneous representations need to be integrated to ensure consistency [29], coordination and composition [24].

We argue that subjectivity in softgoals can be accounted for in a relatively straightforward manner by annotating the softgoal with an identifier of its stakeholder and the suggestion time. Then each stakeholder can refine his requirement (by answering, e.g.: "When is this software fast for you?") independently.

Softgoal: E-Mail reader should be fast.
Added on: 08Nov2005
Stakeholder: Mr. J. Smith
Refined into: An e-mail reader is fast if it opens quickly and creates new e-mail messages quickly.

If a similar softgoal is stated, our approach will see it as different:

Softgoal: E-Mail reader should be fast.
Added on: 08Nov2005
Stakeholder: Mr. J. Smith
Refined into: An e-mail reader is fast if it opens quickly and creates new e-mail messages quickly.

Context-Specificity. At an abstract level, information about the context to which the quality requirement refers can be specific to: the software, the software development process, and the environment in which the software will operate (which can be the hardware environment, the human environment, etc.). For example, "development cost should be low" is a softgoal related to the software development process, whereas "the throughput should be high on the production line" is specific to the human environment in which the software will operate. The combination of the software and environment compose the *information system* (IS) [34]. To specify the context of a softgoal, an attribute *applies to* (with *software*, *environment*, *process* as allowed values) is added to the softgoal template:

Softgoal: E-Mail reader should be fast.
Added on: 08Nov2005
Stakeholder: Mr. J. Smith
Refined into: An e-mail reader is fast if it opens quickly and creates

new e-mail messages quickly.
Applies To: Software

Idealism and Preferences. Quality requirements are often not clear-cut. It is thus beneficial to measure the degree to which a quality requirement, modeled as a softgoal, is satisfied. Metrics, called quality variables in [19], can be designed based on refined requirements. Consider the earlier Mr. J. Smith's Softgoal. It can be refined into two Softgoals, each having a quality variable and an objective function.

Softgoal: The E-Mail reader should open fast.
...
Preferences:
Objective Functions:

Name	Def	Type	Modal	Target	Threshold	Current
3SecOpen	P(TimeToOpen $<$ 3sec)	Prob	Max	80%	70%	*unknown*

Quality Variables:
TimeToOpen: *Duration*
Sample space: distribution of old e-mails, size of old e-mails,...
Definition: time between the input of the request to open the software and the moment the software functionalities can be used.

Softgoal: It should be possible to create new e-mail messages quickly.
...
Preferences:
Objective Functions:

Name	Def	Type	Modal	Target	Threshold	Current
2SecCreate	E(TeToCrMail) $<$ 2sec	Durat	Min	1Sec	2Sec	*unknown*

Quality Variables:
TimeToCrMail: *Duration*
Sample space: number of options available when writing an e-mail,...
Definition: time between the input of the request to create a new e-mail message and the moment its content can be written.

Quality variables are random variables whose distribution can be estimated using data collected by experimentation. Sample spaces can be, e.g., related to similar functionality in existing software. The estimated probability distribution functions are then used to estimate the probability of satisfying the softgoal to some desired level and questions such as, e.g., "What is the probability for the software to open in less than 2 seconds?" or "Under what time will the software open in 90% of cases?" can be answered. Objective functions are associated with quantifiable quality variables, and target levels of performance for each variable are specified. A modal (i.e., *max* or *min*) is also added to indicate the preferred direction. Because not all objective functions are stated in terms of probabilities (i.e., there are objective functions defined over quality variables), the tables used in [19] to specify objective functions are extended here with a *type* column,

to give an indication on the type of variable used in the objective function. In addition, a threshold column is added to further distinguish acceptable from unacceptable degree of softgoal satisfaction.

Quality variables combined with objective functions as in [19] allow the degree of softgoal satisfaction to be measured in cases in which the degree of satisfaction is not under stakeholders' complete control (i.e., there is a probabilistic component in the events affecting softgoal satisfaction). While this is often the case, the RE phase will also involve preferences that can be perceived as deterministic. Assume that a stakeholder provides the following view of a softgoal:

Softgoal: Software should not take too much hardware resources.
...
Stakeholder's view: An e-mail reader will take little hardware resources if it does not occupy memory when not running (i.e., it does not run "in the background").

The stakeholder expresses preference in the quality requirement modeled with the above softgoal. Traditional economics preferences conceptualization (e.g., [16]) can be used to make the preference information from the stakeholders' view explicit. Implicitly, a statement of preference provides partial information about alternatives that are to be ordered using preference relations. We use the classical preference formalism to indicate strict, partial or indifference preference. Using this simple formalism, we can write:

Softgoal: Software should not take too much hardware resources.
...
Preferences:
Choice Preferences:
(run software when requested)$\succ$(software runs in background)

Modeling preferences using objective functions can refine the preference formalization given above. For example, "software should not take too much hardware resources" indicates that the degree of hardware resources used by the software would ideally be measured to determine the degree to which alternative software structures would satisfy the softgoal. Consequently, the above partial softgoal specification can be improved by adding a quality variable that can be used to quantify the "too much" term. Notice that the choice between alternatives specified in *choice preferences* influences the value of quality variables.

Imprecision. Without an accurate notion of the stated and implied needs, the degree of quality satisfaction by software cannot be measured and software properties that could satisfy quality requirements cannot be determined. The use of fuzzy logic has been suggested to formalize imprecise requirements to allow for conflicts between them to be studied ([18,20,31,23]) unfortunately outside the goal-oriented RE field. A key limitation of this approach (see, [19] for a discussion) is that the degree of imprecise requirements satisfaction, measured through a "satisfaction function" that maps software behavior to a degree (comprised between 0 and 1) to which it satisfies a fuzzy (in the sense of [33]) requirement is measure-independent. There are no specific metrics involved, and it is not

obvious how the measurement could be made objective, as in [19]. It would be beneficial if objective metrics and fuzzy logic notation can be combined to express formally the information given in the softgoal template.

Imprecision is dealt with here in a procedural approach, consisting of progressively increasing the precision of initially imprecise information contained in a softgoal template. This is achieved through the application of a set of transformations that manipulate specialized formalisms defined to characterize the above discussed facets of quality requirements modeled as softgoals. The formalisms are necessary to assist stakeholders in representing and reasoning about quality requirements in a systematic manner. A softgoal formalization, based on the discussions above is as follows.

Formal Characterization of Softgoal. We make explicit the facets of softgoals described above by modeling a softgoal $\mathcal{S}$ as a tuple:

$$\mathcal{S} = \langle n, t, St, v, c, P \rangle \tag{1}$$

where n is the softgoal identifier, t is the time of softgoal statement, St is the set of stakeholders that agree on the softgoal, v is the view of the softgoal shared by members of St, c is the context of the softgoal where $c \in software, process, environment$), and P is the preference information associated with the softgoal, including utility. The utilities are evaluated over a set of alternatives for softgoal operationalization (call this set B), each including a combination of the software, environment, and development process. The softgoals are then aggregated to produce the global utility, corresponding to the top softgoal of the project.

The preference information in a softgoal can be represented with a tuple P:

$$P : Obj \times Mod \times T \times Thr \times Curr \times U \tag{2}$$

where Obj is the objective function, Mod is the modality Min, Max of the objective function, T its target value, Thr its threshold value, $Curr$ the quality variable value of the existing alternative, U indicates whether the objective function can be considered as a local utility (see the classical utility theory [15]). The definition of the objective will often make use of auxiliary quality variables. They are defined by an expression, the metric function that (implicitly) depends on the alternative $b \in B$. An objective function is thus a metric function with an associated modality: $mod(m(b))$, where $mod \in Mod$. The modality indicates in which direction the metric function will influence the global utility.

The notation defined above allows the requirements engineer to compare alternatives by:

1. Defining an order among alternatives, as a first approximation.
2. State quality variables Qv to quantitatively compare alternative behaviors, as in [19] but not limited to random variables.
3. Defining metric functions $m(b)$ to associate alternatives b_i to metric values.
4. Combining metric functions $m(b)$ with modalities mod to construct objective functions $mod(m(b)) \in Obj$ which indicate preferred metric values T.

5. Refining metrics to local utilities, and aggregating them to obtain the global utility. Tradeoffs between degrees of satisfaction can be evaluated using the marginal rate of substitution (MRS), a concept taken from economics (e.g., [15]) which, in the terminology used here, indicates the maximal amount of a metric value that a stakeholder is willing to sacrifice for a unit increase in value of another metric. Techniques described in [31] can be reused here, although with caveats noted earlier.
6. Using linguistic facilities as in fuzzy logic: a value above threshold will be deemed *acceptable*, above target *good*.

The formalisms themselves do not eliminate imprecision, but point to information to look for and a way to organize it in order to reduce imprecision.

Imprecision can further be reduced by using logic to formalize the information about alternative behaviors contained in the softgoal. This allows closer integration with later steps of software development.

3.3 Formally Specifying Softgoals

To this point, the traditional softgoal concept has been enriched with templates that allow the expression of subjective, idealist, and context-specific facets of quality requirements information. Imprecision has been indicated, and treated with a simple formal model of the enriched softgoal concept. The model, while summarizing softgoal information in a precise way, does not alleviate imprecision. However, sources of imprecision have become clearer: the fuzzy set of behaviors and time-dependency of preferences which is derived from the fuzziness of the set of behaviors (i.e., preferences change because stakeholders learn about previously unknown behaviors during the development process). Both can receive further treatment: the former through formalization of behaviors, and the latter, through the transformation activities, presented in Sect. 4.

While behavior can be represented in various ways, the goal concept, discussed earlier, proves invaluable in the RE phase (e.g., [10,19,29]). It allows more freedom in the specifications, than, e.g., pre/post condition specification of state transitions used in [18,20,31]. As precise representation of behavior is needed, and since behavior represents *what* software or stakeholders do, functional goals (see, [30]) are used as a concept to model behavior. To remain general, the choice of formal acquisition language for functional goal specification is left to the requirements engineer. It is suggested that temporal logic be used for expressivity reasons. Softgoal formalization then consists of writing formal specification of behaviors $b \in B$ using the chosen acquisition language.

Return to the "software should not take too much hardware resources" softgoal in the previous subsection. Two behaviors appear in its choice preferences attribute. Using the KAOS framework (where a goal is defined as a constraint on behavior [10]), the two behaviors can be specified as KAOS goals (i.e., precise functional goals):

Goal: Maintain [SoftwRunsInBackgr]

Definition: The e-mail reader runs constantly.

Formal Def: $os : OperatSyst; mr : MailReader; os.status = on \Rightarrow mr.status = on$

Goal: Achieve [RunSoftwWhenRequest]

Definition: When the os receives a request to start the mail reader, the mail reader should start running.

Formal Def: $os.status = on \wedge mr.status = off \wedge os.start = mr \Rightarrow mr.status = on$

The above formalization is reflected in the softgoal template by adding a keyword becomes after the imprecise preference relation and rewriting that information using the identifiers for specified behaviors. The imprecise formulation is maintained for traceability reasons.

...

Preferences:

Choice Preferences:

(run software when requested)$\succ$(software runs in background)

becomes Achieve [RunSoftwWhenRequest] $\succ$ Maintain [SoftwRunsInBackgr]

In the NFR framework [22,8] terminology, the above would be represented with a softgoal, two goals, and a contribution link between each of the goals and the softgoal. The preference relation can be translated into a positive and a negative contribution. However, NFR is less expressive, since metrics and most other facets of the softgoal concept presented above are missing.

4 General Guidelines for RE Frameworks

On the basis of the revisited conceptual foundations, guidelines can be suggested as to the transformations that can be applied to softgoals and that need to be present in requirements modeling approaches that aim to employ the given Softgoal conceptualization. Any such transformation activities need to be constructed so that they can deal with all of the four Softgoal facets identified above. Ideally, the transformations would allow initially imprecise, subjective, context-specific, and idealistic softgoals to be transformed into a consistent set of hardgoals (e.g., similar to those of the KAOS acquisition language [10]). We argue that two classes of transformation activities are useful—one for dealing with individual softgoals, and another for transforming several softgoals together.

Single-Softgoal Transformations are aimed at arriving, for each softgoal, at a template in which the initially vague statement of need is made more precise, subjectivity is made explicit, objective metrics are found, and alternative behaviors influencing the degree of softgoal satisfaction are informally identified:

- *(T1) Build an initial softgoal template.* To discover softgoals, the requirements engineer will ask questions about *what* and *how* the software and the wider context should do, according to each stakeholder. The *what* and *how* questions are likely to result in informal and imprecise statements of needs that may be both related to behaviors (i.e., functional aspects) of the context, and how the behaviors need to be exhibited (i.e., nonfunctional aspects; e.g., rapidly, safely, securely, etc.). Consequently, the requirements engineer will need to fill in softgoal templates in a rather sketchy manner at first. As a result of T1, the template needs to contain information about the name of

the softgoal (n), statement time (t), stakeholder identifier (St), stakeholder's view (v), and the context relevant to the softgoal (c).

- *(T2) Identify alternative behaviors that are likely to influence softgoal satisfaction.* The *what* and *how* questions will also lead stakeholders to indicate alternative behaviors whose execution will satisfy to a varying degree the softgoal. The set of identified alternative behaviors for a softgoal j (B_j) can be enlarged by looking at similar existing contexts (and observing, e.g., limitations, errors, etc.), seeking expert opinion on the specific problems, etc.
- *(T3) De-idealize the softgoal.* Stakeholders may express views which qualify or quantify behaviors in ideal ways (e.g., development cost should be lower than X—where X is simply impossible to achieve). De-idealization can be realized by further discussing alternative target values and/or behaviors, or by taking into account benchmarks, which would provide evidence on the idealistic nature of stated needs.
- *(T4-A) Construct objective measurements of softgoal satisfaction.* The aim is to find a set of quality variables (Qv), for the softgoal j. For each quality variable q_{jk}, there should be a metric function ($m_{jr}(b_i)$) to which a modal mod_{jr} is associated to form an objective function ($mod_{jr}(m_{jr}(b_i))$). A target value (t_{jr}) should be defined for the objective function. Quality variables can be derived from information contained in the stakeholders' view (as in the example in Sect. 3.2), in the parent softgoal (see transformation T7), and/or can be based on company-/industry-specific benchmarks. Metric functions can come from knowledge about the events generated by behaviors that are to be measured, from company-/industry-specific standards, and/or behavior categories (i.e., KAOS goal categories [19]). Benchmarks are an invaluable source of target values.
- *(T4-B) Establish preference relations over alternative behaviors.* Based on subjective indications of the stakeholder that has provided the information for softgoal j, alternative behaviors found by application of T2 can be related with preference relations. Preferences can also be established based on objective measurements, when, e.g., the current value for a metric is closer to the target value for a behavior over some other behavior. Notice that preference relations can be objectively constructed only when actual measurements exist (based, e.g., on similar systems) so that current values of quality variables under different behaviors can be observed.

The above transformations are likely to be given as a toolset to the requirements engineer. The order of application will probably be T1 to T4 initially, but iterations should not come as a surprise, especially when additional behaviors are suggested by the stakeholder or due to preference variability over time.

Many-Softgoal Transformations are aimed at establishing relationships between two or more softgoals, to indicate inter-softgoal contribution and refinement. Contribution and refinement are based on widely accepted conceptualizations of such relationships initially given in the NFR framework [22,8], while relying here on a formal model for argumentation of contribution and refinement choices, which itself employs the formal softgoal model proposed above.

- *(T5) Negotiate to avoid conflict.* Contribution between softgoals indicates the degree to which a softgoal supports or obstructs the satisfaction of another softgoal. Contribution is interesting mainly when negative, or conflicting contribution exists between softgoals. Conflict between softgoals may appear in the form of inconsistencies resulting from different terminology (due to subjectivity and imprecision), conflicting preferences, and/or different target values of objective functions. Because of imprecision, the requirements engineer will not be able him/herself to resolve conflicts. Instead, negotiation can be used to lead stakeholders to common understanding, consensus, and closer terminology.
- *(T6) Argument modeling decisions.* Argumentation during negotiation can be recorded using a logical model of argument (for an overview of the research specific to logical models of argument, see [7]). Rigor in recording argumentation during the early phases is relevant not only for traceability reasons, but also because it confronts stakeholders to discuss quality requirements (allowing the requirements engineer to potentially find more information about preferences and alternative behaviors)..
- *(T7) Merge softgoals.* Negotiation will ideally lead stakeholders to a shared terminology and an agreement on quality requirements that have been initially perceived differently. Merging two or more softgoals consists of selecting a subset of preference information available in all softgoals to merge, while using a shared softgoal name, view, context, etc. Objective functions from merged softgoals can be aggregated, provided that quality variables they refer to be converted into compatible types.
- *(T8) Refine a softgoal.* A softgoal is refined if there are sub-softgoals whose joint partial satisfaction is considered equivalent to partially satisfying the refined softgoal. In practical terms, refinement can consist of, e.g., decomposing a softgoal according to some taxonomy (see, e.g., [3] for a privacy and [22] for an accuracy and a performance requirements taxonomies) into sub-softgoals, or making the softgoal more specific through each sub-softgoal. For example, "software should be fast" can be refined into a set of softgoals, e.g., "operation A should be fast",..., "operation Z should be fast".

The result of these transformations can be considered as completed when all of the following conditions hold: (i) a set of behaviors is associated with each leaf softgoal; (ii) there are no conflicting softgoals; (iii) all disagreements on softgoals have been resolved through negotiation; (iv) the set of softgoals is considered sufficiently complete by the stakeholders.

5 Conclusions and Future Work

The aim of the work presented in this paper is primarily a more profound understanding of the Softgoal concept that is commonly used to model requirements in the early stages of requirements engineering. It has been argued that there is more to the information commonly represented using Softgoal instances, than currently established Softgoal definitions seem to indicate. In particular, four

facets of the Softgoal concept are identified—namely: imprecision, subjectivity, context-specificity, and idealism (which involves implicit preference orderings of the stakeholders who state the information represented using Softgoal instances). A tentative formalism for this extended Softgoal conceptualization is suggested, to summarize the information that we argue the requirements engineer can and should attempt to extract from a Softgoal instance (or, in relation to it). It is also illustrated how richer requirements can be obtained when the extended conceptualization is taken into account.

Although a rather simple example has been employed to illustrate the facets we consider relevant, we believe that a powerful insight comes from this paper: a more elaborate treatment of imprecise, subjective, context-specific, and idealistic requirements, usual at the outset of a RE project, can be realized if a commonly used Softgoal concept is extended. Ultimately, this is likely to lead to richer requirements specifications and more stakeholders who are satisfied with the performance of the systems built for them.

Important directions for future work include extending the Softgoal concept further, by possibly identifying additional facets. The formalism needs to be operationalized within already common specification languages. Additional transformation techniques, more effectively exploiting the extended conceptualization remain to be explored.

References

1. Al-Naeem, T., Gorton, I., Ali Babar, M., Rabhi, F., Benatallah, B.: A Quality-Driven Systematic Approach for Architecting Distributed Software Applications. Proc. Int. Conf. Softw. Eng. (2005) 244–253.
2. Alves, C., French, X., Carvallo, J.P., Finkelstein, A.: Using Goals and Quality Models to Support the Matching Analysis During COTS Selection. In French, X., Port, D.: Proc. Int. Conf. on COTS-Based Software System (2005) 146–156.
3. Anton, A., Earp, J., Reese, A.: Analyzing Website Privacy Requirements Using a Privacy Goal Taxonomy. Proc. IEEE Int. Conf. Req. Eng. (2002) 23–31.
4. Avesani, P., Bazzanella, C., Perini, A., Susi, A.: Facing Scalability Issues in Requirements Prioritization with Machine Learning Techniques. Proc. IEEE Int. Conf. Req. Eng. (2005) 297–305.
5. Boehm, B.W., Brown, J.W., Kaspar, H., Lipow, M., MacLeod, G.J., Merritt, M.J.: Characteristics of Software Quality. North-Holland, Amsterdam (1978).
6. Castro, J., Kolp, M., and Mylopoulos, J.: Towards requirements-driven information systems engineering: the Tropos project. Info. Syst. 27, 6 (2002) 365–389.
7. Chesnevar, C.I., Maguitman, A.G., Loui R.P.: Logical Models of Argument. ACM Comput. Surv. 32, 4 (2000) 337-383.
8. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Publishing (2000).
9. Cleland-Huang, J., Settimi, R., BenKhadra, O., Berezhanskaya, E., Christina, S.: Goal-Centric Traceability for Managing Non-Functional Requirements. Proc. Int. Conf. Softw. Eng. (2005).
10. Dardenne, A., van Lamsweerde, A., Fickas S.: Goal-directed requirements acquisition. Sci. of Comput. Prog. 20 (1993) 3–50.

11. Donzelli, P.: A goal-driven and agent-based requirements engineering framework. Req. Eng. 9 (2004) 16–39.
12. ISO: Int. Standard ISO 8402. Quality – Vocabulary. Int. Org. for Standardization, Geneva (1986) (and later).
13. Issarny, V., Bidan, C., Saridakis, T.: Achieving Middleware Customization in a Configuration-based Development Environment: Experience with the Aster Prototype. Proc. Int. Conf. Config. Distrib. Syst. (1998) 207–214
14. Jackson, D.: Structuring Z Specifications with Views. ACM Trans. Softw. Eng. Method. 4, 4 (1995) 365–389.
15. Keeney, R.L., Raiffa, H.: Decisions with multiple objectives: preferences and value tradeoffs. Wiley, New York (1976).
16. Kreps, D.: Notes on the Theory of Choice. Westview Press, Boulder (1988).
17. Landes, D., Studer, R.: The Treatment of Non-Functional Requirements in MIKE. Proc. Europ. Softw. Eng. Conf. (1995).
18. Lee, J., Kuo, J-Y.: New Approach to Requirements Trade-Off Analysis for Complex Systems. IEEE Trans. Knowl. Data Eng. 10, 4 (1998) 551–562.
19. Letier, E., van Lamsweerde, A.: Reasoning about Partial Goal Satisfaction for Requirements and Design Engineering. Proc. ACM SIGSOFT Symp. Found. of Softw. Eng. (2004) 53–62.
20. Liu, X.F., Yen, J.: An Analytic Framework for Specifying and Analyzing Imprecise Requirements. Proc. Int. Conf. Softw. Eng. (1996) 60–69.
21. Liu, L., and Yu, E. Designing information systems in social context: a goal and scenario modeling approach. Info. Syst. 29 (2004) 187–203.
22. Mylopoulos, J., Chung, L., Nixon, B.: Representing and Using Nonfunctional Requirements: A Process-Oriented Approach. IEEE Trans. Softw. Eng. 18, 6 (1992).
23. Noppen, J., van der Broek, P., Aksit, M.: Dealing with Imprecise Quality Factors in Software Design. Proc. Worksh. Softw. Qual., (2005) 1–6.
24. Nuseibeh, B., Finkelstein, A., Kramer, J.: Fine-Grain Process Modelling. Proc. Int. Worksh. Softw. Spec. Des. (Dec.1993) 42–46.
25. Nuseibeh, B., Kramer, J., Finkelstein, A.: A Framework for Expressing the Relationships Between Multiple Views in Requirements Specifications. IEEE Trans. Softw. Eng. 20, 10 (1994) 760–773.
26. Osterweil, L.: Strategic Directions in Software Quality. ACM Comput. Surv. 28, 4 (1996) 738–750.
27. Rosa, N.S., Justo, G.R.R., Cunha, P.R.F.: A Framework for Building Non-Functional Software Architectures. Proc. ACM Symp. Appl. Comput. (2001).
28. Simon, A. H.: The Sciences of the Artificial. 2nd Ed. MIT Press, 1981.
29. van Lamsweerde, A.: Divergent Views in Goal-Driven Requirements Engineering. Proc. ACM SIGSOFT Worksh. Viewpoints Softw. Dev. (1996) 252–256.
30. van Lamsweerde, A.: Goal-Oriented Requirements Engineering: A Guided Tour. Proc. IEEE Int. Conf. Req. Eng. (2001) 249–263.
31. Yen, J., Tiao, W.A.: A Systematic Tradeoff Analysis for Conflicting Imprecise Requirements. Proc. IEEE Int. Conf. Req. Eng. (1997) 87–96.
32. Yu, E.: Modelling Strategic Relationships for Process Reengineering. Ph.D. Thesis, Univ. of Toronto (1995).
33. Zadeh, L.A.: Fuzzy Sets. Information and Control 8 (1965) 338–353.
34. Zave, P., Jackson, M.: Four Dark Corners of Requirements Engineering. ACM Trans. Softw. Eng. Meth. 6, 1 (1997) 1–30.