GOAL, SOFT-GOAL AND QUALITY REQUIREMENT

Thi-Thuy-Hang Hoang and Manuel Kolp
Université Catholique de Louvain, Place des doyens, 1, 1348, Louvain-La-Neuve, Belgium

Keywords: Quality requirement, Goal, Hard-goal, Soft-goal, Functional requirement, Non-Functional Requirements.

Abstract: Requirements are input for the process of building software. Depending on the development methodology, they are usually classified into several subclasses of requirements. Traditional approaches distinguish between *functional* and *non-functional requirements* and the modern goal-based approaches use *hard-goals* and *soft-goals* to describe requirements.

While *non-functional requirements* are known also as *quality requirements*, neither *hard-goals* nor *soft-goals* are equivalent to *quality requirements*. Due to the abstractness of quality requirements, they are usually described as soft-goals but *soft-goals* are not necessarily *quality requirements*. In this paper, we propose a way to clear the problematic ambiguity between soft-goals and quality requirements in goal-based context. We try to reposition the notion of *quality requirement* in the relations to *hard-goals* and *soft-goals*. This allows us to decompose a *soft-goal* into a set of *hard-goals* (required functions) and *quality requirements* (required qualities of function). The immediate applications of this analysis are quality-aware development methodologies for multi-agent systems among which QTropos is an example.

1 INTRODUCTION

Requirements engineering is usually located at the beginning of the development process of software. During this stage, requirements on the system-to-be are elicited and collected from the initial description documents, then are analyzed to define the principal components of the system. The requirement analysis must determine what and how the system will have to provide in order to meet the initial needs of building the system. As a consequence, requirements engineering has been playing the leading role of every software development process.

Zave and Jackson (1997) provide what may appear to be one of the most complete definitions of Requirement Engineering (RE):

Requirement Engineering. *"is the branch of software engineering concerned with the real-world goals for, function of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behaviour, and to their evolution over time and across software families"*.

From the above definition, one can identify all the main components of requirements: goals, functions and constraints. However, this does not provide a proper classification of requirements. To classify better requirements, traditional approaches distinguish between *functional and non-functional requirements* while modern goal-based approaches use *hard-goals and soft-goals* to describe requirements.

Functional requirements describe WHAT the system will have to perform and *non-functional requirements* describe HOW the system will do its jobs (Sommerville 2007). It can be known that some requirements are first considered as non-functional but after having being detailed they appear to be functional. And for some other requirements, it is rather unclear whether they are functional or non-functional. However, it will be clearer if the system's boundary is well-defined and functional requirements resulted from a non-functional requirements are considered as *additional operations* carried out to satisfy additional constraints. It is, thus, preferable to refer functional requirements to system *services* and non-functional requirements to service *quality requirements* such as: accuracy, security, availability, etc. to avoid any ambiguity.

In the meantime, goal-based requirements engineering (Yu 1995) (Chung et al. 2000) divides requirements into *hard-goals and soft-goals*. Both type of goals represent states of affairs that software

Hoang T. and Kolp M. (2010).
GOAL, SOFT-GOAL AND QUALITY REQUIREMENT.
In *Proceedings of the 12th International Conference on Enterprise Information Systems - Information Systems Analysis and Specification*, pages 13-22
DOI: 10.5220/0002872100130022
Copyright © SciTePress

are expected to achieve. For hard-goals, there are clear-cut criteria for whether they are achieved. But there are no clear-cut criteria for soft-goals. It is then usually very hard or even impossible to satisfy completely a soft-goal. One might compare hard-goals to functional requirements and soft-goals to quality requirements. However, this is not an adequate comparison since, for examples, *Availability* (of system) requirement is, indeed, a soft-goal but *Happy Customers* soft-goal is not a quality requirement. As we will see, the relation between soft-goals and quality requirements is not simply an inclusion.

A recent approach presented in (Jureta, Mylopoulos & Faulkner 2008) uses the notion quality requirements to define well-defined and measurable quality and uses soft-goals to define *abstract* qualities that cannot be defined in a clear cut way. Actually, these definitions only provide a classification of quality requirement in to well-defined qualities and abstract qualities and they do not take into account other soft-goals than qualities. Although this approach uses a very interesting analysis, i.e. analysis of speech acts in communicated content between stakeholders, to ground its concepts, its applicability is still open and depends on future applications.

In order to give quality requirements a new role that is qualifier on hard-goals and soft-goals, we propose, in this paper, a new definition for the three most important notions: hard-goals, soft-goals and quality requirements. We also propose the analysis with this new definition to keep track of quality requirement from early requirements until the final products. As a result, quality requirements will play a much more active role in the development process than being just criteria for comparing design alternatives. Goal decompositions with quality requirements will be easy to be derived. The paper will also reveal a clearer connection between hard-goals/soft-goals and functional/non-functional requirements.

From the analyses proposed by the paper, quality requirements can be seen as an additional dimension to the goal dimension. Goals elicit quality requirements and qualities constrain goals in a reinforcing relationship.

We start this paper by giving an overview, in the Section 2, of goal-based requirements engineering. In Section 3, we will reposition quality requirements inside this framework in order to introduce the quality-aware goal analysis. QTropos, a quality-aware agent-oriented software development process that keeps track of quality requirements from the requirement analysis to the final products will be briefly presented in Section 4. We will end this paper with some concluding remarks.

2 GOAL-BASED REQUIREMENT ENGINEERING

In software requirements engineering, goals have become promising tools for requirements eliciting and elaborating. Goals whose satisfaction criteria can be formally described and correctly checked are hard-goals; otherwise they are soft-goals. We can *satisfy* hard-goals but only *satisfice* soft-goals (Chung et al. 2000). Satisficing a goal is a weaker notion of satisfaction where goals are only *partly* satisfied. In practice, the term goal is used extendedly for hard-goals in contrast to soft-goals. This is sometimes very misleading. We will explicitly use the terms *hard-goal* and *soft-goal* and the term *goal* will used to specify the general concept containing both hard-goals and soft-goals.

In the remaining part of this section, we will outline some essential points that make goals really attractive in requirements engineering.

First, goals are objectives that developers expect the system-to-be to achieve. By definition, a goal describes only a state of the world that the system should bring about. Usually, a high level goal does not contain any detail of what and how the system should do to obtain it. Examples of such goals are *Market Share Increased*, *Happy Customer*, etc. These high-level goals can be found quite easily in preliminary documents, interviews of stakeholders, etc. Lower-level goals are then elicited through the analysis of higher-level goals or through the subsequent interactive communications between developers and stakeholders. A common mistake is to consider goals as function designs or algorithmic sketches. Indeed, a goal is usually a description of the outcome of an operation (*posterior conditions*), while a function design is a plan of how to carry out an operation. Given a goal, there can be zero, one or more than one ways (plans or function descriptions) to completely achieve it. It can be said that using goals to model requirements can reduce the gap between the stakeholders' desires and the outcomes of the system by starting *directly* from the needs and imposing *minimally*, a priori, on the choice of structure and technique to be used.

Second, goals provide a greater expressibility, the ability to express, than functional and non-functional requirements. Goals can be divided into two classes based on their satisfiability. Compared to traditional approaches, every functional requirement, defining a function or a component of software system, such as *Turn on a Printer*, *Print a Document* can be described by hard-goals but *Patient Examined* hard-goal is not a functional requirement of a hospital management system. Likewise, every non-functional requirement such as: *Security*, *Availability* can be represented by means of soft-goals, but *Market Share Increased* soft-goal cannot be represented by a non-functional requirement. This proves that hard-goals and soft-goals are richer than functional and non-functional requirements. Up till now, non-functional requirements are considered as a sub-class of soft-goals. However, this position can be revised, as done later in this paper, while preserving the richness of goal-based paradigm.

Third, representing requirements as hard-goals/soft-goals can make use of the very efficient goal decomposition analysis (Van Lamsweerde 2001) to break down the requirements. *AND-decomposition* breaks a goal into a set of sub-goals in which the satisfaction of the parent goal is met only when all the sub-goals are satisfied. On the other hand *OR-decomposition* defines a set of sub-goals such that the satisfaction of only one sub-goal is sufficient for the satisfaction of the parent goal. In the literatures, the two above decompositions are applied only for goals where the satisfaction can be defined in a clear-cut way. For soft-goals, above AND/OR links are not applied since, in most cases, we can only *satisfice* soft-goals. Instead, one can use *contribution links* by which sub-goals and/or sub-soft-goals can contribute positively or negatively with some degree to the parent soft-goal. Other types of analysis are introduced by Tropos project (Castro, Kolp & Mylopoulos 2002) using the i* framework (Yu 1995) as modelling language. In such analysis, requirements are modelled by dependencies (hard-goal, soft-goal, task and resource) between system stakeholders and between system agents.

Forth, conflicts between requirements can be negotiated and cleared by weighing the potential risks produced by each option. For example, Non-Functional Requirement Framework (Chung et al. 2000) provides a way to propagate the positive/negative effect up through the decomposition tree in order to evaluate the risk factor produced by an analysis option. The winner will be the one that could produce less damages.

Fifth, at the lowest level, all hard-goals and soft-goals can be exactly or approximately operationalized. This will results in the functional design of the system-to-be where hard-goals and soft-goals are replaced by future functions of the system.

Further implications of goal-based requirements engineering can be found in (Castro, Kolp & Mylopoulos 2002) (Letier 2001) (Hoang 2008). In the following section, we present a way to separate quality requirements from soft-goals in order to emphasize the use of quality dimension in the requirement engineering.

3 QUALITY REQUIREMENT REPOSITIONING

To reposition quality requirements in the software development process, we first consider the software development in agent-oriented framework with its typical quality requirements.

3.1 Quality Requirements in Reality

Being a promising development trend that can replace traditional development techniques such as structured and object-oriented, *agent-oriented software development* has become a modern trend in software engineering. Software agents are expected to substitute human agents in a lot of tasks. To play some human roles, they are given certain autonomy and intelligence. Moreover, they are designed to live in a virtual society in which agents interact with each others to exchange their knowledge, to reason about the environment and to act towards individual goals as well as social goals. On the one hand, this offers the flexibility and the extendibility to the system, but on the other hand, the liberty of agents may harm the integrity and the coherence of the system. The security flaws will be also among the principal concerns if such systems are deployed in a large scale. These potential problems are probably one of the main reasons why the agent-oriented structure has not been widely used. To gain the popularity, quality requirements such as: flexibility, extensibility, integrity, etc. of the system as a whole and of services of the system must be addressed.

Quality requirements describe not WHAT the system will do but HOW the system will do its job. For example, if *Promptness* is the only quality requirement that have to be satisfied, then between two systems that can do the same job, the one that accomplishes the job in less time, given the same condition, is normally chosen. Our objective is not just using the quality requirements to compare between already-built systems but to develop new software systems that conform to the given quality requirements. To do so for multi-agent systems, when the overall behaviour depends on the atomic interactions between agents, quality requirements must be taken into account at both the system level and agent level. This is crucial since traditional approaches such as (Chung et al. 2000) often treat the quality requirements *only* at the system level and use quality requirements *only* as the criteria in selecting the right option among alternatives. And this seems not sufficient for multi-agent system. Quality requirements must play more active roles during the development process, especially in the requirement analysis. This is the reason why we opt not to analyze quality requirements as a sub-class of soft-goals in order to pay sufficient attentions in the quality aspects. In this new goal-based analysis, we should redefine the roles of and the relations among the three entities: hard-goals, soft-goals and quality requirements.

3.2 Quality Requirements in Focus

We consider again the similarity between functional requirements and hard-goals, between quality requirements and soft-goals. This similarity is strengthened by the fact that it is usually impossible to define the satisfaction criteria of a quality, which makes quality requirements similar to soft-goals. However soft-goals are definitely not quality requirements. One of the reasons is that quality requirements are defined as attributes and/or properties of system functions and are concerned with the quality of services offered by the system, while a soft-goal is a state that the system should achieve. Soft-goals can, thus, describe indirectly functions of a system but neither do quality requirements **(1)**.

To be more specific, soft-goals can be divided into two classes based on their main concerns. When a soft-goal describes a state related to services *inside* of the system, it, in fact, describes directly or indirectly some system functions (*often with* some quality requirements on them). When a soft-goal describes a state related to external actors of the system, it should actually be reflected by some hard-goals or soft-goals inside the system because this is the only way the system can achieve that soft-goal. For example, *Happy Customer* is a soft-goal that acts on the external actor to the system (i.e. *Customer*) while *Transaction Treated in Security and Reliability* is a soft-goal related to services of the system for which *Security* and *Reliability* are two important qualities. And the latter together with some others hard-goals and soft-goals can be the reflection of the former, i.e. *Happy Customer* soft-goal, inside the system. In other words, *Transaction Treated in Security and Reliability* can make *Customers Happy.*

As pointed out earlier that, a function (or object) can be ascribed with some attributes or possesses some properties that represent qualities of that function (or object). This is to say that quality requirements should act on a function (or an object) inside/outside of the system-to-be. Or at least, they act on the whole system, hence everything in the system. As a consequence, it is improper to consider quality requirements as states of the world without specifying on which they act on **(2)**.

The points **(1)** and **(2)** show us that the definition of soft-goals and quality requirements, in which quality requirement is a sub-class of soft-goal, that we usually see in the literatures are no longer applicable. We come to the following set of definitions:

Definition 1. *a hard-goal describes state of some <u>specified objects</u> that the system wants to achieve for which the satisfaction criteria are precisely defined.*

Definition 2*. a soft-goal describes a state of some <u>specified objects</u> that the system wants to achieve for which the satisfaction criteria are not defined in a clear-cut way.*

Definition 3*. a quality requirement describes constraints or perfection levels <u>independent on objects</u> that are constrained to satisfy or satisfice it. When a quality requirement constrains a hard-goal (or soft-goal), it, in fact, constrains the means to achieve that hard-goal (or soft-goal).*

The term *object* in the above definitions is referred to entities that can be ascribed with attributes and/or can possess properties. Popular objects in the context of software system are: stakeholders of the system, system functions, system resources, etc.

By examining the above definitions, we can identify the following differences between these definitions and other definitions in the literature:

- A hard-goal (or soft-goal) must be associated with some specific objects. This condition does not allow a quality requirement to be goal.
- Quality requirements stay outside of hard-goals and soft-goals and influence the way that hard-goals and soft-goals to be achieved.

We can illustrate this by some examples:

- *Security* is a quality requirement but not a soft-goal.
- *Secured Banking Transaction* is a soft-goal since it describes a state of banking transaction of being *secured*.
- The *Web Pages Served* hard-goal is constrained to have *High Availability*. Here *High Availability* is a quality requirement constrains the web server that serves web pages.

In our point of view, the reason for which it is difficult to define the satisfaction criteria of a soft-goal is the presence of one or more quality requirements inside the soft-goal, i.e. *Security* in *Secured Bank Transaction* soft-goal. Moreover, these quality requirements are abstract or implicitly and indirectly stated, in general. Examples are: soft-goals representing the state related to external actors that the system-to-be must achieve, e.g. *Happy Customer* soft-goal. Viewing this soft-goal in this state, it is not clear how to relate this soft-goal to any system functions and their quality requirements. In these cases, the following observation can help to draw a direction that guides our analysis of goal.

Observation. *It is always possible and preferable to transform any stand-alone soft-goal into hard-goals of the system-to-be (on which none or some qualities are required) and these derived qualified hard-goals satisfy or at least satisfice the soft-goal. However, this transform can be indirect in the sense that a soft-goal can be transformed into sub-soft-goal before being transformed into qualified hard-goals.*

In this observation, we treat only stand-alone soft-goals in order to rule out any possibility of conflicts between soft-goals of which may hurt the soft-goal satisfaction. More on conflict negotiation technique will be discussed in the next section in which several analyzing techniques are revised to help developers to deal with the new quality requirement concept. Now let us point out some other important remarks that can be drawn from the above definitions and observation:

- The notion of soft-goal does not play a persistent role in the development process since it is possible to transform into *qualified* hard-goals. And we should do so since hard-goals are clearer notions for describing the system-to-be. Ideally, at some stages of the development process, there will be no more soft-goal. This allows us to have better operationalizations.
- Quality requirements might not be stated explicitly in the high level hard-goals/soft-goals. But they can be elicited from the transform from soft-goals to qualified hard-goal.
- Quality requirements exist thorough the development process and play the role of qualifiers or constraints on goals. They may constrain also soft-goals at the beginning of the analysis process.

As we will keep the notion of quality requirement along side with hard-goals and soft-goals as an additional layer constraining hard-goals and soft-goals, we will use the following graphical notion for quality requirements in our analysis. We call the link from quality requirements to goals *Qualification Link*. In textual form, we use the following predicate to show the qualification link:

```
QUALIFY([Quality], [Goal]).
```

in which, the name of goals and qualities are written inside a pair of square brackets [.]. The link types are in upper case. A set of elements are written inside a pair of curly brackets {.} and are used as abbreviations of individual link. In a link predicate, the order of parameters defines the direction of the arrow in the corresponding graphical representation. Here we omit all the node predicates that specify the node type, i.e. QUALITY(.), HARDGOAL(.) and SOFTGOAL(.).

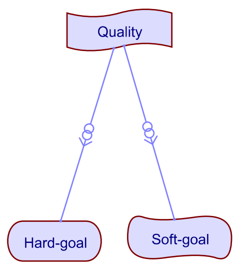

Figure 1: Quality on goal and soft-goal.

In the remaining part of the section, we present how to deal with quality requirements in standard techniques of goal analysis (Van Lamsweerde 2001). Hard-goals and soft-goals are usually decomposed using the goal refinement graph on which two basic operations are: AND decomposition and OR decomposition can be applied to goals and the contribution analysis can be applied in cases where soft-goals can only be approximated. Quality requirements are propagated from the parent node to the sub-nodes following the refinement tree and the type of decomposition.

3.3 Goal Analysis with Quality Requirements

With the presence of quality requirements, we will introduce several new analyses such that: *qualification link*, *quality elicitation* and *quality contribution* together with the standard goal-based operators: AND and OR decomposition.

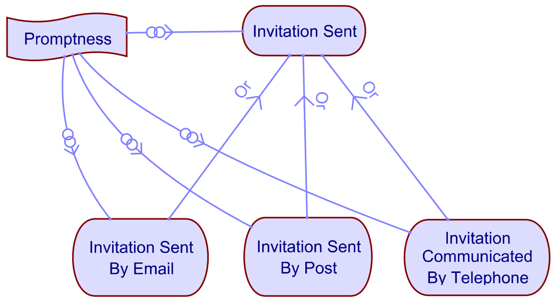

Figure 2: OR decomposition with quality requirement.

3.3.1 OR Decomposition

OR decompositions represent alternatives to fulfil a goal. In Figure 2, the *Invitation Sent* hard-goal is satisfied by either the *Invitation Sent By Email* hard-goal or *Invitation Sent by Post* hard-goal or *Invitation Communicated By Telephone* hard-goal. Since the parent goal is required by *Promptness* quality, all the alternatives should be also required *Promptness*. In textual form, we write

```
OR (
    {
            [Invitation Sent by Email],
            [Invitation Sent by Post],
            [Invitation Communicated by Telephone]
    },
    [Invitation Sent]
).

QUALIFY(
    [Promptness],
    {
            [Invitation Sent By Email],
            [Invitation Sent By Post],
            [Invitation Communicated by Telephone]
    }
).
```

3.3.2 AND Decomposition

In AND decompositions, a goal is satisfied if and only if all the leaf nodes of the decomposition tree are satisfied. In Figure 3, the *Music Played* hard-goal with the *Legality* requirement is satisfied if and only if *Source File is found and downloaded legally*. Then the downloaded *Music File* is *opened* to send

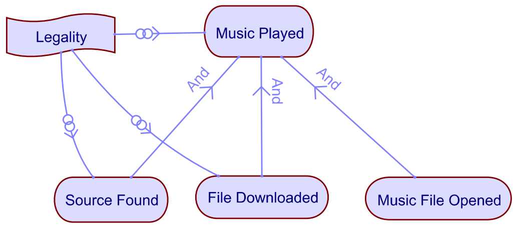

Figure 3: AND decomposition with quality requirement.

sound to speakers. Textually, we write:

```
AND(
  {
             [Source Found],
             [File Downloaded],
             [Music File Opened]
    },
  [Music Played]
).

QUALIFY([Legality], [Music Played]).

QUALITY(
    [Legality],
    {[Source Found], [File Downloaded]}
).
```

Notice that, in this example, the hard-goal *Music File Opened* is not required to have the *Legality* quality since it can be completely satisfied by the two other goals. In general case, when the quality Q itself can be decomposed into several sub-qualities (i.e. Q1, Q2, …), see, for example (Chung et al. 2000), for a taxonomy of quality requirements. The AND decomposition can be rewrite differently if all the sub-qualities are correctly distributed among the leaf nodes.

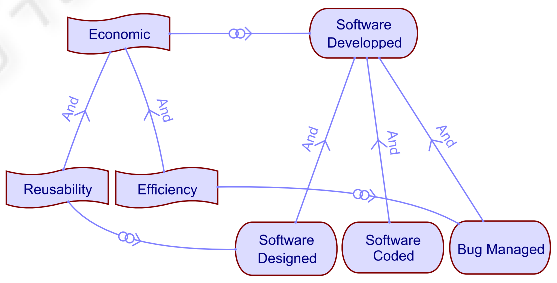

Figure 4: AND decomposition and extended quality requirement.

Figure 4 presents the extended version of AND decomposition. The quality *Economic* is decomposed in the two sub-qualities: *Efficiency* and *Reusability*. To have *Software Developed Economically*, it must (i) be *designed* by using old and creating new *reusable* components, (ii) be provided with an *efficient* bug management system. The coding activities do not affect much the development cost of the software. We then can rewrite the decomposition of *Software Developed* hard-goal as follow:

```
AND(
  {[Efficiency], [Reusability]},
    [Economic]
).
QUALIFY([Economic], [Software Developed]).
AND(
    {
            [Software Designed],
            [Software Coded],
            [Bug Managed]

  },
  [Software Developed]
).

QUALIFY([Efficiency], [Bug Managed]).
QUALITY([Reusability], [Software Designed]).
```

3.3.3 Quality Elicitation

As we have assumed that qualities can be implicitly contained in soft-goals, it is possible that at some stages, some quality requirements are elicited.

In Figure 5, the soft-goal *Email Confidentially Sent* is decomposed into the *Email Sent* hard-goal with the quality *Confidentiality*. Formally we write:

```
ELICIT(
    [Email Confidentially Sent],
    [Confidentiality]
).

AND([Email Sent], [Email Confidentially Sent]).
QUALIFY([Confidentiality], [Email Sent]).
```

Remarks. The three refinements above reveal the link between the notions of functional/non-function (quality) requirements and the notions of hard-goals/soft-goals. This link can be summarized as follow:

- A functional requirement can be represented by a hard-goal. And every hard-goal at the lowest level (right before the operationalization) can be considered equivalent to a functional requirement.
- A soft-goal is defined by a sub-tree of the refinement tree. Starting from a soft-goal and taking only the leaf nodes in the corresponding sub-tree, one can extract all the possible combinations of hard-goals and quality requirement that can satisfice the soft-goal.
- Hard-goals and soft-goals tend to appear at the root of refinement trees while functional and quality requirements tend to be at the leaf nodes.
- At the leaf nodes, there should not be any soft-goal left. Instead, there should be only hard-goals and quality requirements on them.

These allow us to reconfirm that hard-goals/soft-goals are the better choice for requirement engineering (compared to functional/non functional requirements). Moreover, they provide a higher degree of abstraction and expressibility and can be used to capture the *preliminary* requirements only from the very early intentions of the system's stakeholders.

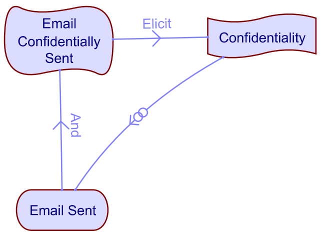

Figure 5: Quality requirement elicitation.

3.3.4 Quality Requirement Fulfilment

To depict the fulfilment of a quality requirement using a goal, we use the contribution link and the following contribution labels (Chung et al. 2000): -- (break), - (hurt), + (help) and ++ (make).

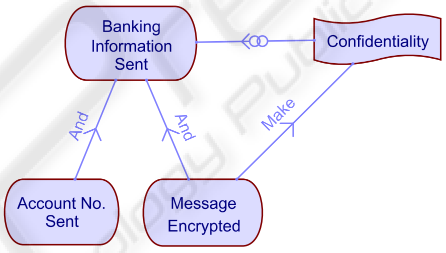

Figure 6: Quality requirement fulfilment.

Figure 6 shows that the *Message Encrypted* hard-goal sufficiently contributes in the fulfilment of the *Confidentiality* quality requirement. Textually, we can write:

```
QUALIFY(
    [Confidentiality],
    [Banking Information Sent]
).

AND(
  {[Account No. Sent], [Message Encrypted]},
  [Banking Information Sent]
).

CONTRIB<Make>(
  [Message Encrypted],
  [Confidentiality]
).
```

3.3.5 Conflict Negotiation

As pointed out earlier, quality requirements can sometimes create conflicts. We take a simple example of an online shop where the *Online Payment Offered* hard-goal (pay for an online purchase) is required to satisfy both *Security* and *Easy To Use* quality requirements, as showed in Figure 7. We have two possibilities to realize that that hard-goal by either: *In-House Service Built* or *Third-Party Service Used.*

In the one hand, with the option *In-House Service*, the shop is free to design its own payment service including the interface to simplify and facilitate the customers' payment. The accounting data is also kept and easily controlled. However, credit card process and management are very complex and often very vulnerable to attacks. Building and maintaining a secured payment system solely for the store are very expensive.

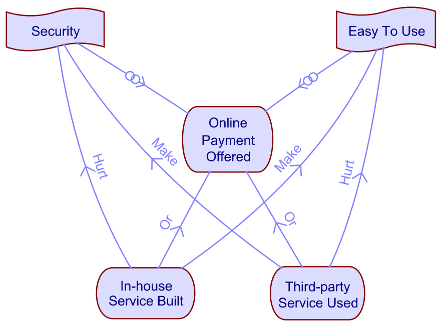

Figure 7: Quality conflict negotiation.

In the other hand, existing third-party services are very well built and maintained. Although it still cannot provide a full guarantee for the security issue, it can be considered a better choice than a self-developed service. The inconvenient side of this service is that it makes the payment process a more complicated. The customers may need to jump back and forth between the shop's website and online payment website to complete a payment.

To resolve conflicts among quality requirements, one possible way is to define the priority of quality requirements as a numerical value in an open scale. A quality requirement that has a higher priority will be fulfilled before other lower-priority ones are fulfilled. In the above example, when issuing a payment, the *Security* requirement is surely more important than the fact that it is *Easy to Use*. We set, for example, the priority of *Security* requirement equal to 2 and of *Easy to Use* to 1. Then the tie is broken as shown in Figure 8.

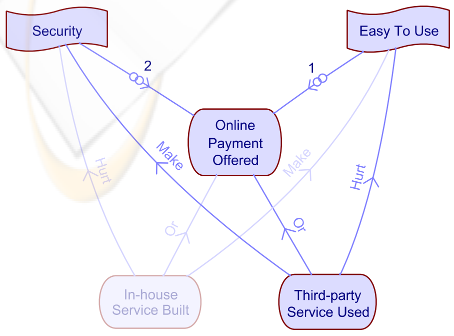

Figure 8: Priority or quality requirement.

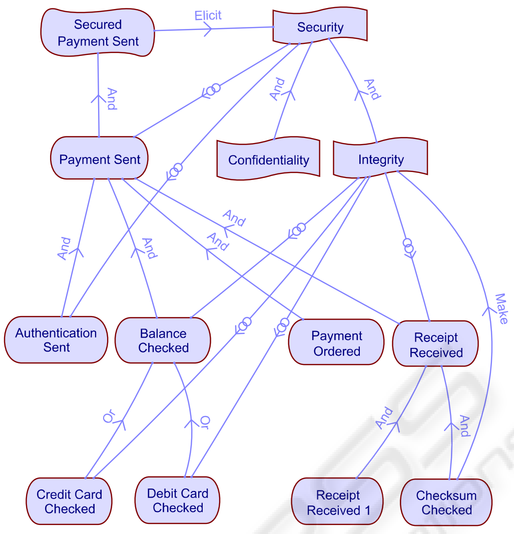

Figure 9: Secured payment example.

3.4 All-in-one Example

To illustrate all the above analysis of goals in one example, we consider, the soft-goal *Secured Payment Sent* taken from a test case of an online shop.

Figure 9 shows a part of the analysis of the soft-goal *Secured Payment Sent*. Based on the defined operations, the analysis of this soft-goal is carried out through the following steps:

- *Secured Payment Sent* soft-goal is decomposed into *Payment Sent* hard-goal and elicits *Security* quality requirement. *Payment Sent hard*-goal is then constrained by *Security* requirement.

```
AND([Secured Payment Sent], [Payment Sent]).
QUALIFY([Security], [Payment Sent]).
```

- *Security* quality requirement is AND-decomposed into *Confidentiality* and *Integrity* quality requirements.

```
AND({[Confidentiality], [Integrity]}).
```

- *Payment Sent* hard-goal is AND-decomposed *Authentication Sent*, *Balance Checked*, *Payment Ordered* and *Receipt Received*.

```
AND(
    {
       [Authentication Sent],
       [Balance Checked],
       [Payment Ordered],
       [Receipt Received]

    },
    [Payment Sent]
).

QUALIFY([Security], [Authentication Sent]).
QUALIFY(
    [Integrity],
    {[Balance Checked], [Receipt Received]}
).
```

- The *Balance Checked* hard-goal is OR-decomposed into *Credit Card Checked* hard-goal and *Debit Card Checked* hard-goal.

```
OR(
 {[Credit Card Checked], [Debit Card Checked]},
 [Balance Checked]
).

QUALIFY(
 [Integrity],
 {[Credit Card Checked], [Debit Card Checked]}
).
```

- The *Receipt Received* hard-goal is AND-decomposed into *Receipt Received 1* hard-goal and *Checksum Checked* hard-goal, where *Checksum Checked* fulfils sufficiently the *Integrity* quality requirement.

```
AND(
 {[Receipt Received 1], [Checksum Checked]},
 [Receipt Received]
).

QUALIFY([Integrity], [Checksum Checked]).
CONTRIB<Make>([Checksum Checked], [Integrity]).
```

At the end of the goal refinement process, the satisficing tree of *Secured Payment Sent* can be summarized by the following two alternatives:

```
AND(
 {
     [Authentication Sent],
     [Credit Card Checked],
     [Payment Ordered],
     [Receipt Received 1],
     [Checksum Checked]
   },
   [Secured Payment Sent]
).

QUALIFY([Integrity], [Credit Card Checked]).
CONTRIB<Make>([Checksum Checked], [Integrity]).
```

or

```
AND(
 {
     [Authentication Sent],
     [Dedit Card Checked],
     [Payment Ordered],
     [Receipt Received 1],
     [Checksum Checked]
   },
   [Secured Payment Sent]
).

QUALIFY([Integrity], [Debit Card Checked]).
CONTRIB<Make>([Checksum Checked], [Integrity]).
```

Note that the above two possibilities of satisficing *Secured Payment Sent* soft-goal does not contain any OR-combination since all the OR-decomposition in the refinement tree will be translated into the alternative possibilities as done for *Balance Checked* hard-goal.

One can argue that the bottom part of Figure 9 is somewhat similar to the diagram of hard-goals/soft-goals integration presented in (Mylopoulos et al. 2001). However, in their approach, soft-goals are mainly quality requirements and hard-goals and soft-goals are analyzed separately and are only correlated very lately in the analysis process. Compared to this, our approach offers the developers with a *top-down* analysis of hard-goals, soft-goals and quality requirements in an integrated scheme from the earliest requirements to the latest design with a sound reasoning procedure.

4 QTROPOS

The presented idea was applied successfully to the Tropos methodology (Castro, Kolp & Mylopoulos 2002) to derive the *Quality-Aware Tropos (QTropos)*.

Tropos has become very popular. It uses *i** modelling framework introduced by (Yu 1995) as the underlying analysis tool through its four phases of software development, namely Early Requirement, Late Requirement, Architecture Design and Detailed Design. The process is often completed with the implementation step using agent-oriented programming languages among which JACK agent would be the best suited.

*i** uses the notion of *distributed intentionality* to model the overall intention of a group of individuals called *actors*. The connecting links are the dependencies between these actors. A dependency is formed if an actor (called *depender*) depends on another actor (called *dependee*) to acquire a *dependum*. There are four types of dependum: *hard-goal, soft-goal, task and resource*.

A *resource* is a physical or informational entity that can be delivered by the depender to the dependee. A *task* is a list of operations that the depender wants the dependee to carry out. *Hard-goals* and *soft-goals* are exactly what we have used in this paper but are put in the context of *depender* and *dependee*.

In the original Tropos process, we can list here some limitations in the quality treatment:

- Quality requirements are included in soft-goals.
- Qualities can be considered as early as in the early requirement phase of Tropos. However, the propagation and elicitation of quality soft-goals from one analysis stage to the next one are rather unclear.
- The use of social patterns at the architectural design is not quality-aware.
- No social pattern designed for the fulfilment and control of quality requirement.

To make it quality-aware, we have to separate quality requirements from soft-goals in Tropos. Since there are four different type of dependencies, quality requirements can constrain on all the dependency types. These dependencies can be seen in the Strategic Dependency model whose results are inputs for the Strategic Rationale model where the mean-ends analysis and task decomposition take place. These analyses can be easily adapted using similar analyses to those in the previous section to incorporate quality requirements. Only one exception is that there are four different types of nodes: Hard-goal, Soft-goal, Task and Resource.

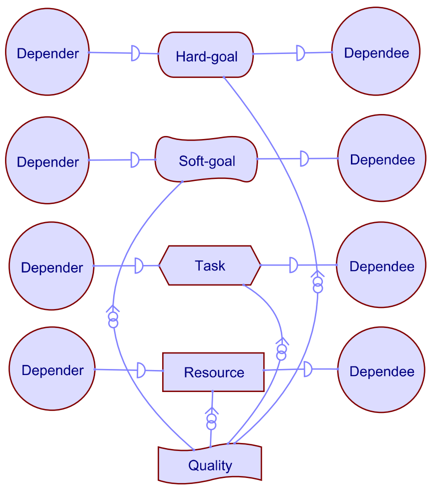

Figure 10: Qualified dependencies.

One of the powerful features of Tropos is that the internal element to an actor can depend or can be depended by other elements of other actors. This allows Tropos models to represent the overall intentions through individuals' intentions.

Strategic Dependency and Strategic Rationale models are used in the *early* and *late requirement* stages of QTropos in order to clarify both intentions of stakeholders and of the system-to-be. From these intentional analysis, developers will be able to take any design and architecture options to match the initial intentions. With the quality requirements are added into this process, QTropos can meet the expectations of the stakeholders in terms of both the functional aspects and the software qualities as well.

More details about QTropos and quality-aware social patterns can be found in (Hoang 2008) and (Hoang & Kolp 2009). A prototype of an analyzing tool namely QTroposCase is also being developed that includes all the developments in this paper as well as other analyses for QTropos. All the figures in this paper are exported from this tool.

5 CONCLUSIONS

Knowing the importance of quality requirement in the context of multi-agent system and of goal-based requirement engineering, this paper proposes a way to elicit, to analyze and to validate the quality requirements. This is done by separating gradually quality requirements from soft-goals, which help the developers to keep track of quality requirements thorough the development process.

The proposed quality-aware goal analysis can be applied to enrich many development processes where goal-based requirement engineering plays the leading role, such as (Q)Tropos.

REFERENCES

Castro, J, Kolp, M & Mylopoulos, J 2002, 'Towards requirements-driven information system engineering: the Tropos project', *Information System Journal*, no. 27, pp. 365-389.

Chung, L, Nixon, BA, Yu, E & Mylopoulos, J 2000, *Non-functional Requirements in Software Engineering*, Kluwer Academic Publishers.

Hoang, TTH 2008, 'Quality-aware agent-oriented software development', Iinternal Report, Louvain School of Management, Université catholique de Louvain.

Hoang, TTH & Kolp, M 2009, 'Social patterns for quality control in agent-oriented systems', *ICSOFT*.

Jack, AOSPL 2002, 'JACK intelligent agents - User guide'.

Jureta, I, Mylopoulos, J & Faulkner, S 2008, 'Revisiting the core ontology and problem in requirements engineering', *Proceedings of the 2008 16th IEEE International Requirements Engineering Conference*.

Letier, E 2001, 'Reasoning about Agents in Goal-Oriented Requirements Engineering', PhD thesis, Université catholique de Louvain, Belgium.

Mylopoulos, J, Chung, L, Liao, S, Wang, H & Yu, E 2001, 'Exploring alternatives during requirements analysis', *IEEE Software*, vol 18, pp. 92-96.

Sommerville, I 2007, *Software engineering*, 8th edn, Addison-Wesley.

Van Lamsweerde, A 2001, 'Goal-oriented requirements engineering: A guided tour', *Proceedings of the 5th IEEE International Symposium on Requirements*, IEEE Computer Society, Washington, DC, USA.

Yu, E 1995, 'Modeling strategic relationships for process reengineering', PhD Thesis, University of Toronto.

Zave, P & Jackson, M 1997, 'Four dark corners of requirements engineering', *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol 6, no. 1, pp. 1-30.