Modelling and Analysis of Quantitative Quality Requirements
Emmanuel Letier
Dept. of Computer Science, University College London
and London Software Systems

Requirements engineering (RE) is concerned with the identification of stakeholders' needs concerning a future system, the specification of services and constraints that satisfy these needs, and the assignment of the resulting requirements to agents such as humans, hardware devices, and software components. Functional requirements describe what functions must be provided to satisfy the stakeholders' needs, quality requirements describe how well the functions must be provided to satisfy these needs. Examples of quality requirements include concerns such as security, performance, reliability, availability, maintainability, and scalability.

While there are many techniques for eliciting, modelling, and reasoning about functional requirements in a precise way, the treatment of quality requirements during the early stages of system development has so far remained largely informal. This situation is extremely undesirable as quality requirements are critical to a project's success and need to be designed into the system since the very early stages of its development. Vague statements of quality requirements provide insufficient information to make key design decisions, particularly when difficult tradeoffs need to be made among competing qualities. Vague statements of quality requirements are also problematic during verification and validation as they provide insufficient information for adequate analysis and testing. There is therefore a strong need for systematic techniques for the elicitation, precise specification, and analysis of quality requirements expressed in a quantitative, testable way.

Our aim is to provide a sound engineering approach to the elaboration and analysis of quality requirements. Our approach relies on the use of quantitative goal models. These models help requirements engineers to define stakeholders' goals in a precise, measurable way, and relate them to precise, testable specifications of software quality requirements. The models can be used to assess the relative values of alternative system designs by computing their impact on the levels of goal satisfaction. Such assessment can be used to guide the selection of a most preferable alternative among the one already indentified, or to guide the generation of further alternative that may provide better levels of satisfaction.

1. Background

Goal orientation is a recognized paradigm for elaborating, structuring and analysing software requirements [Chu00, Lam00, Lam04, Myl06]. Goals are descriptions of stakeholders' needs specified as properties that the system must satisfy. The word system here refers to the composite system made of humans, hardware devices and software. Goals may refer to functional or quality concerns and range from high-level, strategic properties (such as, for an ambulance despatching system, "an ambulance must arrive at the incident scene within 14 minutes after the first call reporting the incident") to low-level, technical ones (such as "the ambulance tracking system shall update the information about each ambulance location at least once every 30 seconds"). Goals are gradually refined into subgoals until the resulting properties can be assigned as the responsibility of single agents. Goals can be specified in first-order linear temporal logic giving the modellers enhanced tool support for elaborating models and reason about their correctness [Lam00, PI10]. However, these techniques allow one to reason about absolute goal satisfaction only. In practice, many goals cannot be satisfied in such an absolute, clear-cut sense (for example, for an ambulance dispatching system, not all incidents will be responded to within the required 14 minutes). Partial goal satisfaction may be due to limited resources, possible failures in the software or application domain, or conflicting goals. A significant part of the RE process consists in exploring alternative system proposals in which more or less functions are automated and different quality requirements are imposed on the software system. Each alternative contributes to different degree to the satisfaction of stakeholders' goals. Understanding these contributions is necessary in order to select the most appropriate system for implementation.

Qualitative approaches such as the Non-Functional Requirements framework [Chu00] and the WinWin framework [Ino1] allow one to reason qualitatively about the positive and negative
impacts of alternative system proposals on stakeholder's goals. These approaches are useful to
gain a first, high-level understanding of the system quality goals and their interactions.
However, for most systems, the information contained in such models is too poor to
adequately decide among alternative system designs.

**Quantitative approaches** allow one to express the impact of alternatives on quality goals
using numbers instead of qualitative values. Many techniques proposed in the literature are
based on quantitative values that have no precise physical interpretation in the application
domain. A well-know example is the house-of-quality weight matrix in the Quality Function
Deployment method [Ak90]. With such techniques, the impact of alternative designs on quality
goals can be quantified (for example, one can express that the level of satisfaction of the goal
"rapid intervention" is 0.8 for one design of an ambulance despatching system and 0.7 for
another), but because the numbers have no physical interpretation in the application domain,
there is no objective way to test, not even in principle, whether the level of satisfaction of
quality goals are actually achieved in the running system. This violates a fundamental principle
of requirements engineering calling for precise and testable requirements. Furthermore,
because such techniques do not provide support for defining quality goals in a precise way, the
goals often remain specified in vague terms only.

Other quantitative approaches are based on more precise specifications of quality goals
expressed in terms of **measurable criteria** that have a precise physical interpretation in the
application domain. The concept of "fit criteria" in the VOLERE requirements engineering
method is a typical example of this [Rob99]. Example of measurable criteria are, for an
ambulance despatching system, the percentage of incidents resolved within 14 minutes, or for
safety critical systems, the mean time between application-specific hazards. With these
approaches, quality requirements are defined in a more precise way but they are still treated
informally; there are no systematic techniques for analysing the completeness and adequacy
of quality requirements, handling conflicts between them, or identifying and reasoning about
their interactions with functional requirements.

Dedicated techniques can be used to reason about specific categories of quality goals: for
example, queuing models for assessing system performance; reliability block diagrams, fault-
trees, event trees and Markov models for assessing system reliability [Tri02]. Probabilistic
temporal logics and associated model-checking tools are also increasingly being used to
analyse quantitatively qualitative properties of software systems [Kwi02]. Causal models such
as Bayesian networks have also been used for analysing non-functional properties such as
safety or reliability [Fen01]. However, there is currently little constructive guidance for
elaborating the models to be analysed and for reasoning about interactions between competing
qualities in an integrated framework.

At the **software architecture level**, there is a growing body of research about the precise
analysis of quality requirements. For example, the Architecture Tradeoff Analysis Method
(ATA/) provides general guidelines for quantitatively assessing the impact of architectural
parameters on multiple quality requirements in order to determine satisfactory trade-offs
[Kaz99]. Quality requirements also play an important role in the context of service-oriented
computing, and proposals for the precise specification and analysis of such requirements are
emerging [Ske04]. These techniques however provide no support for identifying what the
appropriate software quality requirements should be and for relating them to stakeholders'
goals in the application domain.

We have previously developed a **quantitative goal modelling framework** that extends goal
refinement graphs with a probabilistic layer for the precise specification of quality concerns
expressed in terms of application-specific measures. Quantitative goal models are goal-
refinement graphs where behaviour goals are annotated with objective functions referring to
domain-specific quality variables, and goal refinement links are annotated with domain-specific
equations relating the quality variables of the parent goal to quality variables of its subgoals
[PI7]. Once elaborated, quantitative goal models can be used bottom-up to compute the
impact of various alternative system designs on the levels of goals satisfaction in order to
guide the selection of the most appropriate alternative. They can also be used top-down to
compute fine-grained, testable specifications of software quality requirements necessary to
achieve high-level goals to a certain level under various quantitative assumptions about the
environment. Applications of the framework on non-trivial systems including an ambulance
despatching system [PI7], a train control system, and a credit card fraud detection system in the context of an industrial project [PI11] showed promising results. Lessons learnt from these case studies inform our research objectives.

2. Project Objectives and Beyond

The specific objectives for this project are as follows.

- **Automating computations of quantitative goal models.** Currently, computations on the elaborated quantitative goal models are performed through ad-hoc use of mathematical software; this is tedious and error-prone; our objective is to develop tool support to perform such computations fully automatically.

- **Systematic techniques for elaborating quantitative goal models.** The correct mathematical specification of quality goals and equations relating quality variables at different levels of abstraction can be difficult and subject to errors; our objective is to develop a tool-supported system of patterns that will guide the elicitation and specification of quantitative goal models in a way that simplifies the use of the mathematical formalism for the requirements analysts.

- **Case study construction.** We will identify and compile a corpus of large-scale case studies with critical quality requirements and design decisions that we will use for further validation and development of our techniques. Initial quantitative goal models will be built for such systems.

The proposed research will be carried out over a one year period by the principal investigator (PI) and a post-doctoral research associate (RA).

3. Work Packages

**Work Package 1. Automated computations of quantitative goal models.** We will develop a tool supporting the following analysis capabilities.

- **WP1.1. Automated top-down and bottom-up computations for a single design.** Currently, the top-down and bottom-up computations of our quantitative goal models are performed in an ad-hoc way with the help of mathematical software such as Mathematica. The objective in this task is to automate such computations for the analysis of a single system design. One possibility for this is to map our probabilistic models to standard Bayesian Beliefs Networks (BBN) for which efficient algorithms and tools have been developed. Unlike BBN, our models contain a mix of continuous and discrete random variables. Furthermore, the variables in our models are related through declarative equations rather than through BBN’s probability tables. Recent developments of BBN techniques such as the possibilities to automatically and dynamically transform continuous random variables into discrete BBN nodes [Nei06] and to derive BBN probability transition tables from declarative equations will facilitate the mapping of our models to existing BBN tools.

- **WP1.2. Comparing large numbers of design alternatives.** Goal refinement graphs are able to concisely represent a very large number of alternative systems in a single model through the use of OR-refinements and OR-responsibility links. Each OR-refinement or OR-responsibility link corresponds to a decision point. Computing quality functions for each alternative separately is likely to be impractical for most models, because the number of system alternatives grows exponentially with the number of decision points. We will therefore investigate ways to make the analysis of a large number of alternatives as efficient as possible using the OR-refinement structure of goal graphs, and to concisely present the results of evaluating such large number of alternatives to the users in the most meaningful way.

- **WP1.3. Sensitivity analysis.** We will develop sensitivity analysis functions to study how variations in model parameters affect the results of our computations. Such sensitivity analyses could be used by the techniques of WP3.2 to identify “sensitivity points” in system designs [Kaz99] and guide the generation of improved system designs.

- **WP1.4. Validation.** Initially, we will validate the tools on small test cases and larger quantitative goal models among on those we have already developed [PI7]. Additional models will be developed during and after the project.
Work Package 2. Techniques for elaborating quantitative goal models.

- **WP2.1. Patterns for specifying quantitative quality goals.** We will develop and validate a system of reusable patterns of objective functions specifications associated to specific goal categories of the KAOS framework (satisfaction goal, information goal, safety goal, security goal, etc.) [Lam09]. Security goals will be given a particular attention as they represent an important category of goals for which reasoning quantitatively about risks and levels of security is of paramount importance for deciding on an adequate system design.

- **WP2.2. Patterns for specifying quality variables refinements equations.** We will develop and validate a library of reusable refinement patterns between abstract quality variables appearing in the quality functions patterns developed in WP1.2. These quality variables refinement patterns will enrich our existing catalogue of formal goal refinement patterns [Lam09, PI4] with a quantitative probabilistic layer. A few patterns have already been developed to show the feasibility and potential benefits of the approach [P17].

- **WP2.3.** This system of patterns will be constructed and validated by mining examples of precise specification of quality requirements in the literature and in requirements documents.

Work Package 3. Case study construction.

We envision developing case studies from two sources. One is from industrial contacts that we intend to develop during this project. The second is from various projects within University College London. UCL is a large organisation that spends millions of pounds on the procurement, development and customization of software to support his processes. Examples include the management of exam papers, student records, research information, project finances, estate and facilities, and human resources. These projects have numerous stakeholders with complex sets of conflicting needs and quality requirements, notably regarding costs, scalability, security, and usability. The problems faced during these projects are typical of those faced by other large scale organisations, both in the public and private sectors. From the point of view of requirements engineering research, these projects represent valuable resources that are currently largely unexploited. Using these projects for research purposes has many advantages: we already possess most of the domain knowledge, key stakeholders are easily accessible within the organisation, information is readily available without limitations of confidentiality, and we can observe the development, usage and evolution of such systems over long periods of time.

**References**


