Reasoning about Partial Goal Satisfaction for Requirements and Design Engineering

Emmanuel Letier and Axel van Lamsweerde
Département d'Ingénierie Informatique, Université catholique de Louvain
B-1348 Louvain-la-Neuve (Belgium)
{eletier, avl}@info.ucl.ac.be

ABSTRACT

Exploring alternative options is at the heart of the requirements and design processes. Different alternatives contribute to different degrees of achievement of non-functional goals about system safety, security, performance, usability, and so forth. Such goals in general cannot be satisfied in an absolute, clear-cut sense. Various qualitative and quantitative frameworks have been proposed to support the assessment of alternatives for design decision making. In general they lead to limited conclusions due to the lack of accuracy and measurability of goal formulations and the lack of impact propagation rules along goal contribution links. The paper presents techniques for specifying partial degrees of goal satisfaction and for quantifying the impact of alternative system designs on the degree of goal satisfaction. The approach consists in enriching goal refinement models with a probabilistic layer for reasoning about partial satisfaction. Within such models, non-functional goals are specified in a precise, probabilistic way; their specification is interpreted in terms of application-specific measures; impact of alternative goal refinements is evaluated in terms of refinement equations over random variables involved in the system's functional goals. A systematic method is presented for guiding the elaboration of such models. The latter can then be used to assess the impact of alternative decisions on the degree of goal satisfaction or to derive quantitative, fine-grained requirements on the software to achieve the higher-level goals.

Categories & Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specification - *methodologies, languages*.

General Terms

Design, Languages, Documentation.

Keywords

Partial satisfaction of requirements, probabilistic requirements modeling, reasoning about design alternatives, non-functional requirements, goal-oriented requirements engineering.

1 INTRODUCTION

Requirements engineering (RE) is concerned with the identification of the goals to be achieved by the system-to-be, the operationalization of such goals into specifications of services and constraints, and the assignment of responsibilities for such services and constraints among human, physical, and software components forming the system.

A significant part of the RE process consists in exploring alternative system proposals in which more or less functionality is automated and different services and constraints are retained to achieve the higher-level goals.

A *goal* is a prescriptive statement of intent whose satisfaction in general requires the cooperation of some of the agents forming the system [18]. *Agents* are active components such as humans, devices, legacy software or software-to-be components that play some *role* towards goal satisfaction. Some agents thus define the software whereas others define the environment. Goals may refer to functional or non-functional properties and range from high-level concerns (such as "an ambulance must arrive at the incident scene within 14 minutes" for an ambulance dispatching system) to finer-grained ones (such as "information about ambulance location must be updated every 5 seconds"). *Requirements* are fine-grained goals under the sole responsibility of the software-to-be; *expectations* are fine-grained goals under responsibility of some agent in the environment.

Goals often do not need to be satisfied in an absolute, clear-cut sense. For example, standards for ambulance dispatching require that "an ambulance must arrive at the incident scene within 14 minutes after receiving the first call *in at least 95% of the cases*" [19]. For a mine pump control system, a typical goal is that "the mine should not be overflowed *for more than 1 working shift in a thousand*" [14]. For a train control system, a typical goal is that "the distance between two trains should always be bigger than the worst-case stopping distance of the following train, *with a mean time between hazards of the order of* 10^9 *hours*" [34]. Partial degree of satisfaction may be due to, e.g., limited resources, possible failures, or conflicting goals.

In the process of exploring alternative system proposals, requirements, and designs, different alternatives have in general different impacts on the degree of satisfaction of higher-level goals. Consider the goal stating that an ambulance must arrive at the incident scene within 14 minutes, and non-functional goals concerning development costs and time. The elaboration of requirements to meet such goals may lead to the identification of numerous alternatives. For example, call taking and the recording of incident details may be done on paper or on-screen; on-screen call taking may be done with or without a map gazetteer feature to help locate incidents; ambulance allocation could be made fully automatically, interactively, or by only a human allocator; communication with ambulance crews could be done via radio or via mobile data terminals in ambulances; alternative designs can also be considered for detecting failed mobilizations and

SIGSOFT'04/FSE-12, Oct. 31–Nov. 6, 2004, Newport Beach, CA, USA.

recovering from them; and so on [17, 20, 21]. All such alternatives have different impacts on higher-level goals. Evaluating such impacts is necessary to guide the selection of a "most preferred" alternative.

In previous work, we have developed systematic techniques for *producing* alternative system designs by (a) generating alternative goal refinements and responsibility assignments [3, 21], (b) detecting goal conflicts and finding alternative conflict resolutions [16], and (c) generating potential obstacles to goal satisfaction and finding alternative obstacle resolutions for more robust systems [17]. Those techniques consider absolute goal satisfaction only; they do not support the evaluation of generated alternatives to guide the selection of most appropriate ones.

The purpose of this paper is to address such limitations. We present techniques for specifying partial degrees of goal satisfaction and for quantifying the impact of different system alternatives on high-level goals that may be satisfied only partially. Such techniques may then be used to guide requirements elaboration and design decision making.

The paper is organized as follows. Section 2 summarizes some material on goal-oriented RE together with our running example. A short state of the art on reasoning about partial satisfaction is also presented there to motivate our approach more precisely. Section 3 describes our technique for specifying goals that may be achieved only partially. Section 4 presents a technique for propagating partial degrees of satisfaction along goal refinement/abstraction links. The use of this technique for various kinds of analysis is discussed in Section 5.

2 BACKGROUND

2.1 Goal-oriented requirements engineering

Fig. 1 shows a portion of the goal model built in [17, 20] for the London Ambulance Service (LAS). This case study is based on a report on the inquiry that followed a major system failure [19]. The top goal is specified as follows:

Goal Achieve [AmbulanceIntervention]

Definition *For every urgent call reporting an incident, there should be an ambulance at the incident scene within 14 minutes after receiving the first call.*

FormalDef $\forall$inc: Incident
Reported (inc)

$\Rightarrow \Diamond_{\leq 14min}$ ($\exists$ amb:Ambulance) Intervention (amb, inc)

This specification fragment introduces a goal named AmbulanceIntervention together with its natural language definition The *Achieve* keyword declares a goal pattern; in this case, it states that some target property must eventually hold. A formal counterpart may be specified optionally in a real-time linear temporal logic for formal reasoning.

The top goal in Fig. 1 is *AND-refined* in two subgoals: AmbulanceMobilized and MobilizedAmbulanceIntervention. The former requires an ambulance to be mobilized for an incident once an urgent call is received; the latter requires an ambulance to arrive at the incident scene once it is mobilized. An AND-refinement is *complete* if the conjunction of the subgoals, together with domain properties, is sufficient to establish the satisfaction of the parent goal [3]. In the above example, the conjunction of the assertions formalizing the goals AmbulanceMobilized and MobilizedAmbulanceIntervention can be proved to entail the assertion formalizing the parent goal.

A goal may also be *OR-refined* into alternative sets of subgoals. Each alternative corresponds to an alternative design for achieving the parent goal. Consider the goal AllocatedAmbulanceMobilized in Fig. 1; this goal requires an ambulance to be effectively mobilized once it has been allocated to an incident. In a first alternative, the goal is AND-refined into subgoals AmbulanceMobilization CommunicatedOnRadio and AmbulanceMobilizedFromRadio Mobilization, which corresponds to a non computer-based ambulance dispatching system. In the second alternative, the goal is AND-refined into subgoals AmbulanceMobilizationCommunicatedOnMobileDataTerminal and AmbulanceMobilizedFrom MobileDataTerminalMobilization, which corresponds to a computer-based mobilization system.

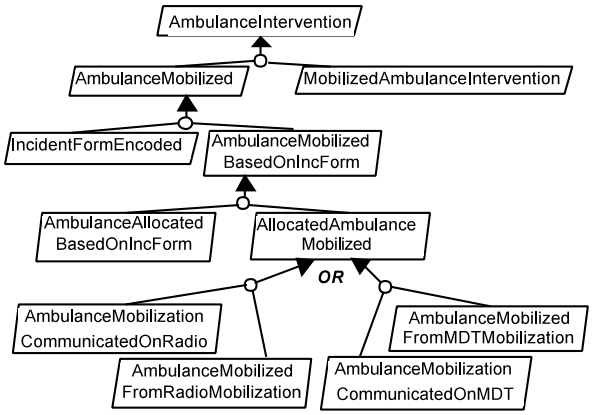

Figure 1 - Portion of a goal model for the LAS

Goals need to be refined until they can be *assigned* as responsibilities of single agents [18]. In general, alternative assignments of goals to agents have to be explored as well. For example, the goal requiring the most appropriate ambulance to be selected for an incident might be assigned to a human allocator agent, to a software agent, or to a combination of the two (in which case the goal has to be further refined to reach single responsibility assignments).

First-sketch goals elicited during goal modelling tend to be unrealistic; they are likely to be violated from time to time due to unexpected agent behavior that cannot be controlled by the software. An *obstacle* to some goal is a condition that may prevent the goal from being satisfied [17]. For example, obstacles such as AmbulanceCrewPushingWrongButtonOnMDT or AmbulanceCrewTakingUnalloccatedVehicle obstruct the goal AccurateAmbulanceStatusInfo; an obstacle such as RadioBlackSpot obstructs the goal AmbulanceMobilizationCommunicatedOnRadio. Obstacles can be generated and resolved systematically [17]. In general, *alternative* obstacle resolutions need to be explored.

2.2 Reasoning about partial goal satisfaction

The problem of evaluating alternatives with respect to degrees of goal satisfaction has been addressed by qualitative and quantitative reasoning techniqes.

2.2.1 Qualitative reasoning techniques

Typical examples are the NFR framework [2] and the WinWin model [11]. The idea is to expose positive or negative influences of different alternatives on non-functional goals formulated in high-level terms.

Fig. 2 shows a simplified NFR model for the LAS system. Compared with the goals in Fig. 1, which have a precise semantics defined in terms of admissible system behaviors [20], NFR *softgoals* are goals whose satisfaction cannot be established in a clear-cut sense [25]. They are used to compare refinement alternatives; the extent to which each alternative contributes to the softgoals is determined qualitatively. In Fig. 2, the NFR softgoal FastIntervention is refined into softgoals FastMobilization and CloseMobilization. Each of these softgoals is further refined into subgoals.

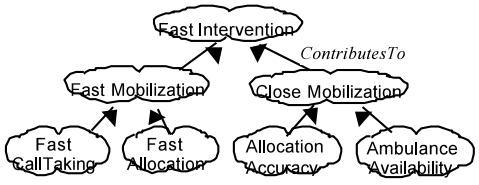

Figure 2 - Portion of a NFR model of the LAS system

Suppose that we have to choose between two alternative options *A* and *B* that contribute with different degrees to the various softgoals in Fig. 2. Table 1 suggests the contribution of the two alternatives to the leaf softgoals, e.g., *A* contributes very positively to FastAllocation, positively to AllocationAccuracy, very negatively to FastCallTaking, and negatively to AmbulanceAvailability.

	Fast CallTaking	Fast Allocation	Allocation Accuracy	Ambulance Availability
Option A	–	++	+	-
Option B	++	--	++	+

Table 1 - Qualitative contributions to softgoals

Qualitative *propagation rules* are proposed in the NFR framework to compute the qualitative degree of goal satisfaction for each alternative system. They allow the satisfaction status of non-functional goals to be determined in each alternative, namely "satisficed" (partially satisfied), "denied" or "undetermined".

NFR models provide useful information about the kind of influence different alternatives may have on high-level goals identified at an early stage of the RE process. However, the goals remain too vague for deep, accurate understanding of the model; degrees of satisfaction have no clear meaning; the propagation rules often result in undetermined satisfaction for higher-level goals. The applicability of such frameworks for accurate decision support appears thus limited.

2.2.2 Quantitative reasoning techniques

Qualitative techniques may be extended by quantifying positive and negative influences of alternative options on the degree of goal achievement. The simplest and most obvious technique consists in replacing qualitative contribution values such as "++" or "-" by numerical weights. For example, Table 2 suggests quantitative measures on contributions of options *A* and *B* to the leaf softgoals in Fig.2. The numbers there capture degrees to which the goals are satisfied in the corresponding alternative. Each contribution link in Fig. 2 would be numerically weighted as well, e.g., the contribution weights of FastMobilization and CloseMobilization on the parent goal FastIntervention might be given the values 3 and 5, respectively.

The degree of satisfaction of a goal is then computed as some weighted average of the degree of satisfaction of its subgoals. The alternative for which the top-level goal has highest degree of satisfaction would then be selected.

	Fast CallTaking	Fast Allocation	Allocation Accuracy	Ambulance Availability
Option A	0.1	0.7	0.5	0.3
Option B	0.7	0.1	0.7	0.5

Table 2 - Quantitative contributions to softgoals

This elementary principle is used fairly often in practice [33]. There are, however, still key problems with such quantifications.

- *What do such numbers mean?* In general, the meaning of numbers capturing partial satisfaction and partial contribution is subjective; such numbers have no physical interpretation in the application domain. What does it mean to say that the softgoal FastIntervention is satisfied at degree 0.7? Goal formulations and degrees of satisfaction remain vague and non measurable. This violates a fundamental principle of requirements engineering calling for precise and measurable requirements and specifications. Numerous quantitative techniques proposed in the literature are based on subjective concepts of partial satisfaction – see, e.g., the house-of-quality weight matrix in the QFD method for evaluating impacts of alternatives on customers goals [1]; the utility weights in [28] for requirements negotiation support; the fuzzy logic values in [35] for detecting conflicts among imprecise requirements; the degrees of evidence of satisfiability/deniability in quantitative extensions of the NFR framework [8]; and degrees of goal achievement in [6] for guiding the selection of alternatives to prevent or mitigate risks.
- *Where are such numbers coming from?* The usual answer is that stakeholders are responsible for providing numerical values for degrees of satisfaction/contribution. There are two ways of eliciting such values: by considering absolute values for goal contribution weights, as in [6], or by considering relative values through pairwise comparison, as in the AHP technique [30] used in [12] for requirements prioritization. In methods based on weighted averages, contribution weights correspond to substitution rates between criteria. Eliciting relative values through pairwise comparison is therefore preferable as it may reflect mathematical properties of the weights. It provides the additional benefit of allowing consistency checks on the elicited values. In both cases, however, there are no clear criteria for validating estimations – e.g., how can we check whether the contribution weights of goals FastMobilization and CloseMobilization are really 3 and 5? Even after system deployment the subjective nature of parameter estimations makes it difficult to test whether such estimations were correct or not.

Beside quantitative techniques based on subjective criteria, there are many quantitative methods based on *objective criteria,* that is, criteria with some domain-specific physical interpretation such as the percentage of incidents resolved within 14 minutes or the mean time between application-specific hazards. Dedicated techniques are widely used to reason about specific categories of

non-functional properties. For example, queuing models is a standard technique for assessing system performance; reliability block diagrams, fault-trees, event trees and Markov models are standard techniques for assessing system reliability; stochastic models are also used for analyzing system availability. More recently, probabilistic models have been proposed to quantitatively assess system security [24].

The application of such techniques during requirements engineering is limited in two respects: (a) there is little constructive support for elaborating the models to be analyzed; (b) there is little integration of such models with the other models used in the RE process.

Our objective is therefore to integrate quantitative techniques for reasoning about non-functional properties, based on objective criteria, with goal-oriented techniques for building requirements models. In particular, systematic support should be provided to:

- specify and propagate partial degrees of goal satisfaction precisely, in terms of domain-specific phenomena;
- guide the elaboration of models that help stakeholders assess the impact of alternatives on the degree of goal satisfaction.

Our work is related to the Architecture Tradeoff Analysis Method (ATAM) [13]. This method provides support for quantitatively assessing the impact of architectural parameters on multiple non-functional requirements in order to determine satisfactory trade-offs. The goal-oriented method described below might be used as input to ATAM in order to guide (a) the identification, refinement and precise specification of the non-functional requirements that drive architectural decisions, and (b) the elaboration of the quantitative models needed to quantify the impact of alternative designs on those non-functional requirements.

Our work is also related to the use of Bayesian networks (BN) for making predictions about non-functional properties such as safety or reliability [7]. The elaboration and validation of BNs prove to be very difficult for complex systems. As it will be seen below, our approach provides constructive support for building probabilistic networks that are somewhat similar to BNs.

2.3 A few basics from probability theory

A *sample space* S is the set of possible outcomes of a random experiment. A *random variable* V of type T on a sample space S is a mathematical function $V: S \rightarrow T$. The *cumulative distribution function* (CDF) of a random variable V is the probability that the value of V on an experiment run is less than or equal to some value x: $CDF_V(x) = P(V \leq x)$. The *probability density function* (pdf) of a continuous variable V is the derivative of its cumulative distribution function: $pdf_V(x) = d\,CDF_V(x) / dx$. A comprehensive introduction to probability theory can be found in [31].

3 SPECIFYING DEGREES OF GOAL SATISFACTION

We first introduce objective functions and quality variables to specify partial goal satisfaction in a semi-formal but precise, application-specific way (Section 3.1). Heuristics for identifying objective functions and quality variables from goal specifications are presented next (Section 3.2). Section 3.3. then shows how to optionally specify objective functions formally in a probabilistic temporal logic.

3.1 Objective functions and quality variables

The partial degree of satisfaction of a goal is modeled by annotating that goal with the following domain-specific attributes.

- One or more *quality variables* corresponding to domain-specific, goal-related random variables defined over specified sample spaces; each quality variable has a name, a sort, and a natural language definition that relates it to the quantity it denotes in the application domain.
- Zero, one or more *objective functions* that define domain-specific, goal-related quantities to be maximized or minimized; each objective function is given a name, a mathematical definition refering to the goal quality variables, a modality recording whether the function is to be maximized or minimized, and a target value for the system-to-be. (The value currently achieved by the system-as-is may be specified as well for comparison purpose.)

As a first example, the degree of satisfaction for the goal Achieve[AmbulanceIntervention] in Fig. 1 might be specified as follows:

Goal Achieve[AmbulanceIntervention]

Def *For every urgent call reporting an incident, an ambulance must arrive at the incident scene within 14 minutes.*

Objective Functions

Name	Def	Modal	Target	Current
8MinRespRate	P (RespTime ≤ 8')	Max	50%	35%
14MinRespRate	P (RespTime ≤ 14')	Max	95%	80 %

Quality Variables

ResponseTime: *Time*

***{Sample Space:** set of reported incidents.*

***Def:** time between first report of the incident and arrival of the first ambulance at the incident scene}*

The goal Achieve[AmbulanceIntervention] has two objective functions, namely, 8MinRespRate and 14MinRespRate, defined as the probability that the response time for an incident is less than 8 minutes and 14 minutes, respectively. These functions have to be maximized. (For the real LAS system, the target values for these objective functions were set in 1992 by government standards to 50% and 95%, respectively.) Those objective functions refer to a quality variable ResponseTime of sort Time.

There may be different choices of objective functions and quality variables for the same goal. For example, an alternative objective function for the above goal might have been to minimize the average response time. Identifying appropriate objective functions is critical; wrong decisions may be taken if they are based on wrong objectives. This observation is among the key reasons for preferring a domain-specific approach for modeling partial goal satisfaction to a subjective, domain-independent one.

A single goal may have more than one objective function, and the range of such functions is not necessarily [0, 1] – think of, e.g., the average response time for a performance goal, the mean time between hazards for a safety goal, and so on.

Objective functions need not to be specified for every goal. As we will see later, propagation rules are defined on quality variables, not on objective functions. Such functions need to be defined only when there are *measures* on the goal quality variables that are of interest to the decision making process. Typically, objective functions will be specified on important high-level goals of the application domain, and on a few key lower-level goals.

Our approach ensures a complete *separation of concerns* between the behaviors prescribed by a goal (as specified by its informal and formal definition in temporal logic) and the quantitative aspects of partial satisfaction (specified by the quality variables and objective functions). Such separation of concerns provides several benefits.

- One may elaborate and analyze the behavioral aspects of a goal model, using techniques mentioned in Section 2.1, *independently* from the quantitative, probabilistic aspects. It is thus possible to model and reason about partial goal satisfaction at a later stage of the RE process, only if desired.
- Requirements engineers may specify and reason about quantitative goal models even when the goals are specified only *semi-formally*. When the optional formal layer of the goal model is used, however, probabilistic extensions of temporal logic may be used to specify objective functions formally and relate them to the goal specification in temporal logic (see Section 3.3).
- Quality variables and objective functions may change independently from the behavioral specification of the associated goal. For the goal Achieve[Ambulance Intervention], the definition of objective functions related to response time may be different from one region to another; they may evolve over the years to take into consideration different incident categories (e.g., incidents such as strokes requiring faster intervention than others). However, the behavior prescribed by the goal, namely, that an ambulance must arrive at the incident scene, remains the same.

3.2 Identifying quality variables and objective functions

Identifying the right objective functions and quality variables, and specifying them correctly, is critical for evaluating alternative designs against the right criteria. We propose five heuristics to support this task.

(H1) Identify quality variables from conditions constrained by the goal

Goals constrain system behaviors. Finding out in a goal specification which conditions are being constrained by the goal provides a good source for candidate quality variables; such conditions may explicitly or implicitly refer to them.

For example, the specification of Achieve[Ambulance Intervention] constrains two conditions: (i) an ambulance eventually arriving at the incident scene, and (ii) within a specified delay of 14 minutes. From condition (i) we may identify a Boolean quality variable IncidentDropped, defined on the sample space of all reported incidents, whose value is true when no ambulance ever responds to the incident. One objective function on this quality variable would be to to minimize the probability that IncidentDropped is true. From condition (ii) we may identify the quality variable ResponseTime specified in the previous section.

As another example, consider the following subgoal of Achieve [Ambulance Intervention] (see Fig. 1):

Goal Achieve [Ambulance Mobilized]
Def *For every urgent call reporting an incident, an ambulance must be mobilized to the incident location within less than three minutes; the mobilized ambulance should be at less than 11 minutes from the incident location.*

This goal constrains four conditions: (i) an ambulance being mobilized, (ii) the location to which the ambulance is mobilized being the incident location, (iii) the mobilization being within the specified delay, and (iv) the distance between the ambulance and the incident location being less than 11 minutes. The following quality variables are then identified systematically from these conditions:

NoAmbMobilized: *Boolean*
{**Sample Space** *set of reported incidents*
Def *NoAmbMobilized is true if no ambulance is ever mobilized for the incident*}

MobilizationTime: *Time*
{**Sample Space** *set of reported incidents*
Def *time between first report of the incident and mobilisation of the first ambulance*}

WrongMobDest: *Boolean*
{**Sample Space** *set of reported incidents*
Def WrongMobDest *is true if the mobilized ambulance is mobilized to a wrong location}*

DistMobilizedAmbulance: *Time*
{**Sample Space** *set of reported incidents*
Def *distance, measured in time to go from one location to the other, between the first mobilized ambulance and the incident location*}

Some of these quality variables, like WrongMobDest, are clearly related to obstacles on the same goal [17]. The concepts of obstacle and quality variable are in fact both related to the idea that a goal may not be *fully* satisfied. However, the two concepts are quite different and should not be confused. An obstacle is a condition that captures a set of behaviors violating the goal; a quality variable is a random variable used to assess how well the goal is satisfied. The purpose of obstacle analysis is to identify what might go wrong; the purpose of quality variables is to help determine how much it might go wrong or, conversely, how much the goal will be satisfied.

As another example of use of this heuristic, consider the following goal for a mine pump control system [14].

Goal Achieve [WaterAlarmIffCriticalWaterLevel]
Def *The water alarm should be raised if, and only if, the water level exceeds the 'Overflow' limit.*

This goal constrains two conditions: (i) the water alarm being raised when water level is critical, and (ii) the water alarm being not raised when the level is not critical. This suggests two quality variables to be specified: WaterAlarmFailure and FalseWaterAlarm; the former refers to failure to raise the alarm when the water level is critical; the latter refers to alarm raising when the water level is not critical. (These variables will be specified more precisely in the next section.)

(H2) Specify quality variables according to quality variables of parent goals

The precise definition of quality variables and objective functions for a goal is strongly related to the choice of quality variables and objective functions for the parent goals.

In the above mine pump example, the need to raise an alarm when the water level is critical contributes to the satisfaction of the goal Avoid [MinerInOverflowedMine]; the objective function for the latter goal is the *conditional* probability that a miner is in the mine when the mine is overflowed. The definition of the quality variable WaterAlarmFailure and its sample space will be defined accordingly:

WaterAlarmFailure: *Boolean*
{**Sample Space** *set of all time points at which the water level becomes critical*

Def WaterAlarmFailure *is true if the alarm is not raised when the water level becomes critical. Its probability is the conditional probability that the alarm is not raised when the water level becomes critical.}*

Similarly, the need to raise the alarm only if the water level is critical contributes to the goal Avoid[WorkingShiftLost]; the objective function for the latter goal is to minimize the probability of losing an 8-hour working shift in order not to lose more than one shift in a thousand. The choice of a quality variable for false water alarms might therefore be defined on the same time interval:

FalseWaterAlarmInShift: *Boolean*

*{***Sample Space** *set of all possible working shifts*

Def FalseWaterAlarmInShift *is true if at some time during the working shift (8 hours) the water alarm is raised when the water level is not critical*

Objective functions for the goal Achieve [WaterAlarmIffCriticalWaterLevel] might then be defined as follows:

Name	Def	Modal	Target
AlarmFailureRate	P (WaterAlarmFailure)	Min	10^{-4}
FalseAlarmRate	P (FalseWaterAlarmInShift)	Min	10^{-3}

(H3) Specify quality variables from required level of analysis

The choice of quality variables is also strongly related to the level of detail at which the quantitative analysis needs to be performed.

For example, the above static Boolean quality variable *WaterAlarmFailure* leads to simple time-independent quantitative analysis of the water alarm system, in the spirit of quantitative fault-tree analysis. Alternatively, we might have modelled a dynamic quality variable

WaterAlarmFailure: *Time* → Bool

that allows for time-dependent analysis of water alarm system failures, in the spirit of Markovian reliability analysis. We might have made yet another choice in which the same objective functions would have been defined in terms of quality variables *TimeToWaterAlarmFailure* and *TimeToFalseWaterAlarm* of type *Time*, in the spirit of mean-time-between-failures analysis.

At the early stages of requirements engineering we might start by building a simple model, for example based on time-independent analysis, then use this model to identify which parts of the system may have critical impact on decisions to be taken, from which more precise models might be defined for more detailed analysis.

(H4) Identify objective functions from domain standards

Many application domains already have well-established quantified performance measures from which objective functions can be identified. The standards imposed by the UK government for ambulance response time is an example of such an objective function. As another example, two essential objective functions mentioned in the preliminary description of the BART train control system are to maximize the number of passengers and maximize the mean time between hazards [34].

(H5) Identify objective functions from goal categories

Goal categories [18] have specific objective functions that can be instantiated to the system being modelled. For example, generic objective functions for *safety goals* are to minimize the probability of failure on demand, to maximize the mean time between hazards, or to maximize reliability over a specific time interval. *Performance goals* have standard objective functions related to response times and throughputs. Objective functions for *security goals* may refer to mean time to security failures [23, 24]. Numerous templates of measurable objective functions for a variety of functional and non-functional requirements are suggested in [29]. Further work is required to build a rich, coherent catalog of reusable objective functions for different goal categories.

3.3 Formal specification of objective functions

At the optional formal layer of the language, objective functions may be specified more precisely using probabilistic extensions of temporal logics such as PCTL [9]. PCTL extends the branching-time temporal logic CTL with a probabilistic operator *Pr* for specification of properties such as "*an ambulance must arrive at the incident scene within 14 min in 95% of the cases*". This property is formalized by the following PCTL assertion:

$\forall$ inc: Incident

Reported (inc)

$\Rightarrow Pr_{\geq 95\%}\ \Diamond_{\leq 14min}$ ($\exists$ amb:Ambulance) Intervention (amb, inc)

In this assertion, the CTL symbols ***F*** (finally) and ***G*** (globally) are noted $\Diamond$ and $\Box$, respectively; P $\Rightarrow$ Q is a shorthand for the branching-time assertion ***AG*** (P $\rightarrow$ Q).

The semantics of PCTL assertions is defined over different forms of probabilistic transition systems like Discrete Time Markov Chains or Markov Decision Processes. Probabilistic model checkers such as PRISM [15] or ETMCC [10] may be used to check whether a given probabilistic transition system satisfies some probabilistic temporal assertion.

Formal PCTL definitions of objective functions are derivable in a systematic way from the linear temporal logic assertion formalizing the goal and the semi-formal definitions of the objective functions. For example, the above PCTL assertion defining the *14MinResponseRate* objective function is derivable from the linear temporal logic assertion defining the goal *Achieve[AmbulanceIntervention]* in Section 2.1 and the semi-formal definition of this objective function in Section 3.1.

In principle, such systematic derivation is hampered by the incomparability in expressive power of the linear-time and branching-time paradigms [32]. In practice, however, our goal assertions are based on formal patterns [3], similar to those of [4], that hide the differences between the two paradigms.

We extended our catalog of formal goal patterns with a first set of objective function specifications. For example, the following three objective function patterns cover a wide range of PCTL formulas we encountered in our case studies and in the literature:

$C \Rightarrow Pr_{\geq p}\ \Diamond_{\leq d}\ T$ $\qquad$ $C \Rightarrow Pr_{\geq p}\ \circ T$ $\qquad$ $Pr_{\geq p}\ \Box_{\leq d}$ Inv

The first pattern is frequently found for objective functions on *Achieve* goals; it was used in the LAS case study to specify performance-related objective functions. The other two patterns may be used to specify "probability of failure on demand" and "reliability" objective functions on safety goals. The objective functions *AlarmFailureRate* and *FalseAlarmRate* for the mine pump example in Section 3.2.2 are formalized just by instantiating these patterns, which yields:

WaterLevel $\geq$ 'Overflow' $\Rightarrow$ $Pr_{\geq 1\text{-}10^{-4}}$ $\circ$ WaterAlarm = 'On'

$Pr_{\geq 1\text{-}10^{-3}}\ \Box_{\leq 8h}$ (WaterAlarm = 'On' $\rightarrow$ WaterLevel $\geq$ 'Overflow')

Our emphasis here is on providing guidance for identifying objective functions and specifying them formally. Checking these assertions with a probabilistic model checker assumes a detailed

operational model of the software and its environment to be available under the form of a probabilistic transition system. Techniques for systematic construction of such probabilistic transition systems are to our knowledge still missing.

In the next section, objective function values will be computed with the help of *declarative* propagation rules that relate quality variables of subgoals to quality variables of parent goals. Their formal PCTL definition will therefore not used for that purpose.

4 PROPAGATING PARTIAL DEGREES OF SATISFACTION

Propagation rules define how the degree of satisfaction of a goal is determined from the degrees of satisfaction of its subgoals. Section 4.1 shows how quality variables of a goal are related to those of its subgoals through refinement equations. Section 4.2 discusses generic patterns for guiding the specification of such refinement equations. Section 4.3 describes how probability density functions on quality variables can be derived from their refinement equations in order to compute objective functions.

4.1 Refinement equations on quality variables

Refinement equations are domain-specific equations that relate the quality variables of a parent goal to the quality variables of its subgoals.

Let us consider the goal Achieve[AmbulanceIntervention], refined in Fig. 1, with quality variables ResponseTime and IncidentDropped. The quality variables for its subgoal Achieve[AmbulanceMobilized] were specified in Section 3.2 using heuristics (H1). The other subgoal Achieve [MobilizedAmbIntervention] has two quality variables, namely, AmbDelay and AmbIntervFailure. The former captures the difference between the actual and expected times for a mobilized ambulance to reach the incident location; the latter captures the fact that the mobilized ambulance fails to arrive at the incident scene.

The refinement equations for quality variables ResponseTime and IncidentDropped on the parent goal Achieve[AmbulanceIntervention] are then specified as follows.

Goal Achieve [AmbulanceIntervention]
RefinedInto Achieve [AmbulanceMobilized] , Achieve [MobilizedAmbulanceIntervention]

Quality Variable Refinements:

ResponseTime =
MobilizationTime + DistMobilizedAmbulance + AmbDelay

IncidentDropped =
NoAmbMobilized **or** WrongMobDest **or** AmbIntervFailure

Note that such equations are associated with goal refinement links rather than with quality variable specifications because in case of alternative goal refinements different refinement equations must be defined for each alternative.

Refinement equations may also be extended to take into consideration further goals introduced to resolve obstacles in the ideal goal graph.

For example, Fig. 3 shows the goal Achieve[AmbIntervFailure Recovered], introduced to mitigate the obstacle MobilizedAmbulanceFailure. To take such resolution goals into account, the refinement equations for the goal Achieve[Ambulance Intervention] need to be modified as follows:

IncidentDropped =
NoAmbMobilized **or** WrongMobDest
or (AmbIntervFailure **and** AmbFailureNotRecovered)

ResponeTime =
MobilizationTime + DistMobilizedAmbulance + AmbDelay
(**if** AmbIntervFailure = **false**)
MobilizationTime + DistMobilizedAmbulance + AmbDelay
+ AmbFailureRecovDelay
(**if** AmbIntervFailure = **true**)

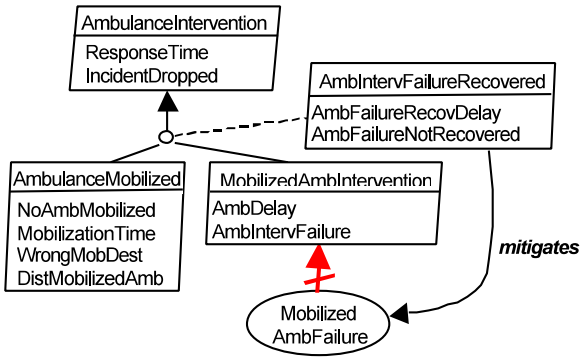

Figure 3 - Quantitative refinement with obstacle mitigation

Quality variables and refinement equations can be defined similarly on obstacles, obstacle refinements and obstruction links between obstacles and goals. This important issue is not discussed further for lack of space.

Refinement equations are not always definable in terms of simple equations on quality variables. They may need to be defined in terms of probability distribution functions on such variables. For example, the quality variable *DistAllocatedAmbInfo* on the goal Achieve[AmbulanceAllocatedBasedOnIncForm] in Fig. 1 captures the distance, measured in time units, between the incident location and the nearest available ambulance *according the information known about ambulances*. In our model, this variable is related to quality variables on subgoals by an equation taking the form:

$$P(\text{DistAllocatedAmbInfo} \leq x) = 1 - (1 - x^2/L^2)^{\text{NbAvAmb} * (1 - P(\text{FalseUnavailability}))}$$

where L represents the size of the city (for simplicity we assumed a square town), *NbAvAmb* denotes the number of ambulances truly available, and *FalseUnavailability* is a Boolean quality variable on the goal AccurateAmbulanceInfo, capturing that an available ambulance is believed to be unavailable.

Quantitative models composed of quality variables on goals and refinement equations on goal refinement links might be seen as probabilistic networks akin to Bayesian networks. The differences are that (a) our models are composed of both discrete and continuous variables (BNs are usually composed of discrete variables only), and (b) the relationships between variables are defined here by refinement equations instead of probability tables. More importantly, the techniques described here provide systematic guidance for building such probabilistic networks from goal models while establishing strong relationships between the two models.

4.2 Using quantitative refinement patterns

In order to help requirements engineers specify refinement equations, we built a preliminary catalog of *quantitative* goal refinement patterns. A probabilistic layer was simply added to frequently used patterns from our existing catalog [3, 21].

As an example, Fig. 4 shows the probabilistic extension of the milestone-driven refinement pattern.

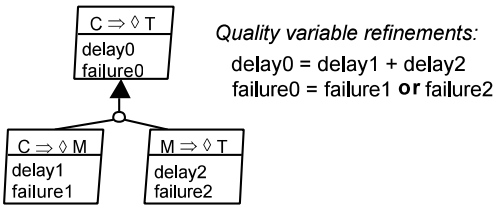

Figure 4 - Quantitative milestone-driven refinement pattern

This pattern was used extensively in the LAS case study. A few other quantitiative patterns have been defined for refinement patterns such as "introduce accuracy goal" [21] and "split by chaining" [3]. Further work is needed to enrich this catalog.

4.3 Computing objective functions

Refinement equations provide a basis for computing objective functions from estimations of probability distributions on leaf quality variables. This section describes how to derive equations defining the probability density functions (*pdfs*) of quality variables from the *pdfs* of lower-level quality variables.

To obtain such equations, we need to assume that leaf quality variables are mutually independent. (This assumption is also made for quantitative analysis of fault trees and for the computation of Bayesian networks.) If this assumption cannot be made, the dependent quality variables need to be refined further until leaf variables are reached that may be assumed to be independent.

For Boolean quality variables that are refined entirely in terms of other Boolean quality variables, the refinement equations may be seen as a fault tree (see, e.g., the IncidentDropped variable on the goal Achieve[AmbulanceIntervention]). Well-known techniques may therefore be used to compute minimum cut sets and probabilities for such variables [31].

For non-Boolean variables whose refinement involves a mix of continuous and Boolean random variables, the refinement equations are used to derive equations on probability density functions on the quality variables. For example, if the refinement of a continuous random variable *delay0* is given by the equation

$$\text{delay0} = \text{delay1} + \text{delay2},$$

its probability density function $\text{pdf}_{\text{delay0}}$ is given by the following well-known formula, under the assumption that delay1 and delay2 are independent:

$$\text{pdf}_{\text{delay0}}(x) = \int_0^x \text{pdf}_{\text{delay1}}(i) \times \text{pdf}_{\text{delay2}}(x-i)\,di$$

To apply such formulas we need to make sure that the quality variables on the right-hand side of the equations are independent. If we assume that leaf quality variables are mutually independent, we can infer that quality variables are independent if they have no common subvariables.

When quality variables are not independent, we first have to eliminate the non-independent variables from the refinement equation by expanding their own refinement equation until all variables in the transformed refinement equation are mutually independent. For example, the refinement equation of the quality variable ResponseTime at the end of Section 4.1 refers to the variables MobTime, DistMobilizedAmb, AmbDelay and AmbFailureRecovDelay. In our model, these variables are not independent because AmbFailureRecovDelay depends on the three other variables according to the following refinement equation:

AmbFailureRecovDelay = AmbFailureDetectionTime + MobilizationTime
+ DistMobilizedAmb + AmbDelay

The variable AmbFailureRecovDelay is therefore eliminated from the refinement equation of ResponseTime which yields the following equation:

ResponeTime =
MobilizationTime + DistMobilizedAmb + AmbDelay
(**if** AmbIntervFailure = **false**)
2 (MobTime + DistMobilizedAmb + AmbDelay)
+ AmbFailureDetectionTime
(**if** AmbIntervFailure = **true**)

All variables in the right hand side of the equation are now independent, and the *pdf* of ResponseTime may be expressed in terms of the *pdf* of the other variables in the equation.

Once refinement equations have been reformulated in terms of probability density functions, standard mathematical software tools such as *Mathematica* may be used to compute the integrals numerically.

The process of identifying non-independent quality variables by identifying shared subvariables may reveal important problems in the system design. In the mine pump control system, we thereby identified that our design used the same water sensor to detect high water levels, that would trigger the pump, and water levels that would raise the alarm in case of pump failure. In our model, quality variables on the goals PumpOnWhenHighWater and AlarmWhenMineOverflowed were therefore not independent. Such interdependence has a disastrous effect on the higher-level goal Avoid[MinerInsideOverflowedMine] to which those two goals contribute [20].

5 REASONING WITH QUANTITATIVE GOAL MODELS

Section 5.1 describes how alternative system designs can be evaluated by *bottom-up* propagation of quality variable values. Section 5.2 briefly discusses how quantitative requirements can be derived by *top-down* propagation of such values.

5.1 Evaluating alternative designs

The evaluation of alternative system designs proceeds in three steps:

1. For each alternatbles of the corresponding model.
2. For each alternative, compute the objective functions of the higher-level goals by bottom-up propagation from the estimated distribution functions (as described in Section 4.3).
3. Compare the resultive, estimate the distribution functions for the leaf quality varias obtained.

Distributions of leaf quality variables can be estimated in various ways, e.g., from statistical data about the current system or about similar systems, from judgments and experience of experts, from typical reliability figures about standard devices (such as sensors, water pumps, radio communication systems, etc.), or from standard human reliability figures such as those used in safety-critical applications [22]. For example, an analysis of the old, paper-based LAS system might have revealed that AmbDelay follows a normal distribution with a mean of 10 seconds and a

standard deviation of 30 seconds, that CallTakingTime follows an exponential distribution with a mean of 60 seconds, that the probability of an ambulance believed to be available and being actually unavailable (FalseAvailability) is 1%, etc.

Rough estimations of distributions for leaf quality variables in new system versions can be estimated in comparison with the current system, based on experience with similar systems, on-site experiments with simulations of the new system, etc. For example, experts might estimate that on-screen call taking would lower the mean call taking time from 60 to 50 seconds.

While estimations on quality variables of goals assigned to agents in the environment are *quantitative assumptions*, estimations on quality variables of goals assigned to the software-to-be are *quantitative requirements*. For example, the above mean call-taking time for the new on-screen call-taking system yields a quantitative usability requirement.

As an illustration, Table 3 shows computed objective function values for three alternative LAS systems based on plausible (but fictitious) estimations for some of the important parameters in our model. The *Mathematica* tool was used to perform a mix a symbolic and numerical integrations. The three systems are the following:

- the old, paper-based system that was running before automation took place [19];
- a system *S1* in which call taking and ambulance allocation are automated, which reduces the mean call-taking time, allocation time, and allocation errors (ambulances are still mobilized through radio communication as in the paper-based alternative);
- a system *S2* in which ambulance mobilization is automated as well, which is expected to reduce the mean ambulance mobilization time but increase the mean time to detect a mobilization failure (due to loss of direct radio communication with ambulance crews).

Parameters	**Paper-based system**	**Automated system *S1***	**Automated system *S2***
NbAmbulances	300	300	300
Mean CallTakingTime	60 sec	**50 sec**	**50 sec**
Mean AllocationTime	60 sec	**30 sec**	**30 sec**
Mean AllocationDistError	90 sec	**30 sec**	**30 sec**
Mean MobilizationTime	80 sec	80 sec	**50 sec**
P (FalseAvailability)	1%	1%	1%
P (FalseUnavailability)	2 %	2 %	2 %
P(MobFailureForAvAmb)	1%	1%	1%
Mean MobFailureDetectionTime	2 min	2 min	**4 min**
Objective Functions			
8MinRespRate	30%	47%	52%
14MinRespRate	84%	93%	95%

Table 3 - Expected response times for alternative LAS systems

The quantitative model may also be used to estimate the impact of obstacles on objective functions. For example, in our estimations for the automated system *S2*, the probabilities of FalseAvailability, FalseUnavailability, and MobFailureForAvAmb were assumed to be the same as in the old paper-based system. According to our model, if the probabilities of these three variables increase to 5%, 10%, and 10%, respectively, the 8-minute and 14-minute response rates would drop to 45% and 90%, respectively.

Note that such numbers are not intended to capture accurate expected values for the objective functions; they are obtained from rough estimates about leaf quality variables and from refinement equations in which simplifying assumptions have been made. More accurate characterizations of expected objective function values would be hard to get at such early stages of the development process. The evaluation here is aimed at *comparing* alternatives to guide critical high-level decisions that need to be taken early during requirements and design engineering. We are therefore more interested in the *relative* strengths and weaknesses of alternatives, with respect to objective functions, than in their accurate quantification.

Validating quantitative models is a key issue when such models are used for guiding critical decisions. In contrast with quantitative techniques based on subjective criteria, our approach makes it possible to validate or unvalidate the models. *Before* deployment of the system-to-be, collecting data about the system-as-is allows one to make assumptions on distribution functions of quality variables, validate the refinement equations used in the model, and possibly rectify them. In case many goals and refinements are common to alternative systems (as it is often the case at higher levels of goal AND/OR graphs), the validation of refinement equations may concern large common parts of the quantitative models for alternative systems-to-be. On the other hand, *after* deployment of the selected system-to-be, the validity of parameter estimations and refinement equations may be checked through run-time monitoring. Detecting violations on quality variable estimations at run time may trigger on-the-fly changes of system parameters or even switching to a completely different design at run time [5].

The identification of all relevant quality variables is another important issue when such quantitative models are elaborated from scratch. We showed how such variables are systematically identifiable from goal definitions. Formal techniques and tools can be used in addition to identify missing subgoals and assumptions together with their associated quality variables [3, 17, 27].

5.2 Deriving quantitative requirements

Beside the bottom-up use of refinement equations for propagation-based evaluation of alternative models, one might use such equations top-down to derive quantitative requirements, as defined in the previous section, that are necessary to achieve given targets on the objective functions.

For example, given estimations for all leaf variables in Table 3 except *FalseAvailability* and *FalseUnavailability*, we could derive maximum probabilities for these variables to guarantee the 52% and 95% target values of the *8MinRespRate* and *14MinRespRate* objective functions in system *S2*, respectively. Such probabilities would define quantitative requirements for achieving the goal Achieve[AccurateAmbulanceAvailability Info] .

6 CONCLUSION

Our aim is to provide constructive, meaningful and accurate support for decision-making during requirements and design engineering. To achieve this, we quantify the impact of alternative

system designs on the degree of satisfaction on non-functional goals. Partial degrees of satisfaction are characterized precisely in terms of *application-specific* phenomena; they are specified compositionally with declarative behavioral specifications; and they are propagated upwards or downwards in goal refinement graphs according to application-specific equations. Rules were provided to identify, from goal specifications, application-specific parameters and functions to be used for quantifying impacts of alternatives requirements options on degrees of goal satisfaction.

Our techniques were applied so far to two non-trivial safety-critical systems: one for ambulance despatching [19] and the other for mine pump control [14]. Validation of the probabilistic models with respect to real operational data would be helpful in determining the level of accuracy that can be obtained from such models. More experience in this direction is expected in a near future from a real air traffic control project.

Our emphasis here was on constructive support for probabilistic modeling. The actual calculation of objective functions is done through ad-hoc use of mathematical software. This may turn out to be hard for complex refinement equations. Dedicated tools to perform such computations more effectively should be provided. Our framework should also be extended to handle uncertainties on parameter estimations by use of confidence intervals. Another interesting avenue concerns the generation of test cases and operational profiles from our quantitative models. Extensions to the Objectiver toolset [26, 27] are planned along those directions.

Many systems have to satisfy conflicting non-functional goals. Our techniques for evaluating alternative designs against them could therefore be complemented by techniques for multicriteria decision making [33].

System design engineering by identification and evaluation of alternatives is an iterative process. Further work is also required towards systematic techniques for generating better designs from the insights gained in the evaluation of the current alternatives.

ACKNOWLEDGEMENT. The work reported herein was partially supported by the Belgian "Fond National de la Recherche Scientifique" (FNRS).

7 REFERENCES

[1] Y. Akao, *Quality Function Deployment QFD: Integrating Customer Requirements into Product Design*, Productivity Press, 1990.

[2] L. Chung, B. Nixon, E. Yu and J. Mylopoulos, Non-functional requirements in software engineering. Kluwer Academic, 2000.

[3] R. Darimont and A. van Lamsweerde, "Formal Refinement Patterns for Goal-Driven Requirements Elaboration", *Proc. 4th ACM Symp. Foundations of Software Engineering*, Oct. 1996, 179-190.

[4] M.B. Dwyer, G. S. Avrunin and J.C. Corbett, "Patterns in Property Specifications for Finite-State Verification", *Proc. 21st Intl. Conference on Software Engineering*, Los Angeles, May 1999.

[5] M. Feather, S. Fickas, A. van Lamsweerde, and C. Ponsard, "Reconciling System Requirements and Runtime Behavior", *Proc. IWSSD'98 - 9th International Workshop on Software Specification and Design*, Isobe, IEEE CS Press, April 1998.

[6] M.S. Feather, S.L. Cornford, J. Dunphy & K. Hicks, "A Quantitative Risk Model for Early Lifecycle Decision Making", in *Proceedings of the Conference on Integrated Design and Process Technology*, Pasadena, California, June 2002.

[7] N. Fenton and M. Neil, "Making Decisions: Using Bayesian Nets and MCDA", *Knowledge-Based Systems* 14, 307-325, 2001.

[8] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani. Reasoning with Goal Models, *Proc. 21st International Conference on conceptual Modeling (ER2002)*, October 2002.

[9] H. Hansson and B.Jonsson, "A Logic for Reasoning about Time and Probability", *Formal Aspects of Computing* Vol.6, 1994, 512–535.

[10] H. Hermanns, J. P. Katoen, J. Meyer-Kayser, and M. Siegle, A Markov Chain Model Checker, *Proc. Tools & Algorithms for the Construction & Analysis of Systems*, LNCS 1785, 2000.

[11] H. In, B. Boehm, T. Rodgers and M. Deutsch, "Applying WinWin to Quality Requirements: A Case Study", *Proc. 23rd International Conference on Software Engineering*, 2001, pp. 555-564.

[12] J. Karlsson, K. Ryan, A Cost-Value Approach for Prioritizing Requirements, *IEEE Software* 14(5): 67-74 (1997)

[13] R. Kazman, M. Barbacci, M. Klein, S.J. Carrière, "Experience with Performing Architecture Tradeoff Analysis," Proceedings of ICSE'99. Los Angeles, CA, May 1999, 54-63.

[14] J. Kramer, J. Magee, M. Sloman et al, CONIC: an Integrated Approach to Distributed Computer Control Systems. *IEE Proceedings*, Part E 130, 1, January 1983, pp. 1-10.

[15] M. Kwiatkowska, G. Norman and D. Parker, "Probabilistic Symbolic Model Checking with PRISM: A Hybrid Approach", In *Proc. TACAS'02*, LNCS 2280, pages 52-66, Springer-Verlag, April 2002.

[16] A. van Lamsweerde, R. Darimont, E. Letier, "Managing Conflicts in Goal-Driven Requirements Engineering", *IEEE Transactions on Software Engineering*, November 1998.

[17] A. van Lamsweerde and E. Letier, "Handling Obstacles in Goal-Oriented Requirements Engineering", *IEEE Transactions on Software Engineering*, Special Issue on Exception Handling, October 2000.

[18] A. van Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour", *Invited Minitutorial, Proc. RE'01 - 5th Intl. Symp. Requirements Engineering*, Toronto, August 2001, pp. 249-263.

[19] *Report of the Inquiry Into the London Ambulance Service*. February 1993. The Communications Directorate, South West Thames Regional Authority. Also available at http://www.cs.ucl.ac.uk/staff/A.Finkelstein/las.html

[20] E. Letier, Reasoning about Agents in Goal-Oriented Requirements Engineering. Ph. D. Thesis, University of Louvain, May 2001; http://www.info.ucl.ac.be/people/eletier/thesis.html.

[21] E. Letier and A. van Lamsweerde, "Agent-Based Tactics for Goal-Oriented Requirements Elaboration", *Proc. ICSE'02: 24th Intl. Conf. on Software Engineering*, Orlando, IEEE Press, May 2002.

[22] N.G. Leveson, *Safeware - System Safety and Computers*, Addison-Wesley, 1995.

[23] B. Littlewood et al, "Towards operational measures of computer security", *Journal of Computer Security*, Vol. 2, 1993, 211-229.

[24] B.B. Madan et al, "Modeling and Quantication of Security Attributes of Software Systems". *Proc. Int. Conf. on Dependable Systems and Networks (DSN'02)*, June 2002.

[25] J. Mylopoulos, L. Chung and B. Nixon, "Representing and Using Nonfunctional Requirements: A Process-Oriented Approach", *IEEE Trans. on Software. Engineering*, Vol. 18 No. 6, June 1992, 483-497.

[26] http://www.objectiver.com.

[27] A. Rifaut, P. Massonet, J.F. Molderez, C. Ponsard, P. Stadnik, H. Tran Van and A. van Lamsweerde, "FAUST: Formal Analysis of Goal-Oriented Requirements Using Specification Tools", *Proc. RE'03*, Sept. 2003, 350.

[28] W.N. Robinson, "Negotiation Behavior During Requirement Specification", *Proc. 12th Intl. Conference on Software Engineering*, IEEE Computer Society Press, Nice, 1990, pp. 268-276.

[29] S. Robertson and J. Robertson, *Mastering the Requirements Process*. Addison-Wesley, 1999.

[30] T.L. Saati, The Analytic Hierarchy Process, McGraw-Hill, 1980.

[31] K.S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, 2nd Edition, Wiley, 2002.

[32] M. Y. Vardi, Branching vs. linear time: Final showdown, *Proc. TACAS'2001: Tools and Algorithms for the Construction and Analysis of Systems*, LNCS Vol. 2031, Springer-Verlag, 2001, 1-22.

[33] P. Vincke, *Multicriteria Decision-Aid*, Wiley, 1992.

[34] V. Winter, R. Berg, and J. Ringland, *Bay Area Rapid Transit District, Advance Automated Train Control System: Case Study Description*. Technical Report, Sandia National Labs, 1999. Also available at http://www.pst.informatik.uni-muenchen.de/dagstuhl/.

[35] J. Yen and W. Amos Tio, "A Systematic tradeoff Analysis for Conflicting Imprecise Requirements", *Proc. Third IEEE Intl. Symp. on Requirements Engineering (RE'97)*, Jan 97.