A Probabilistic Framework for Goal-Oriented Risk Analysis

Antoine Cailliau and Axel van Lamsweerde

Département d'Ingénierie Informatique
Université catholique de Louvain
Louvain-la-Neuve, Belgium
{antoine.cailliau, axel.vanlamsweerde}@uclouvain.be

***Abstract*— Requirements completeness is among the most critical and difficult software engineering challenges. Missing requirements often result from poor risk analysis at requirements engineering time. Obstacle analysis is a goal-oriented form of risk analysis aimed at anticipating exceptional conditions in which the software should behave adequately. In the *identify-assess-control* cycles of such analysis, the assessment step is not well supported by current techniques. This step is concerned with evaluating how likely the obstacles to goals are and how likely and severe their consequences are. Those key factors drive the selection of most appropriate countermeasures to be integrated in the system goal model for increased completeness. Moreover, obstacles to probabilistic goals are currently not supported; such goals prescribe that some corresponding target property should be satisfied in at least *X*% of the cases.**

The paper presents a probabilistic framework for goal specification and obstacle assessment. The specification language for goals and obstacles is extended with a probabilistic layer where probabilities have a precise semantics grounded on system-specific phenomena. The probability of a root obstacle to a goal is thereby computed by up-propagation of probabilities of finer-grained obstacles through the obstacle refinement tree. The probability and severity of obstacle consequences is in turn computed by up-propagation from the obstructed leaf goals through the goal refinement graph. The paper shows how the computed information can be used to prioritize obstacles for countermeasure selection towards a more complete and robust goal model. The framework is evaluated on a non-trivial carpooling support system.

Keywords — Obstacle analysis, risk assessment, probabilistic goals, requirements completeness, goal-oriented requirements engineering, risk analysis, quantitative reasoning.

I. INTRODUCTION

Missing requirements and assumptions are reported as one of the major causes of software failure [23]. Incompleteness often results from a lack of anticipation of unexpected conditions under which the software should behave adequately. A natural inclination to conceive over-ideal systems prevents adverse conditions from being properly identified and, when likely and critical, resolved through appropriate countermeasures.

Risk analysis should thus be at the heart of the requirements engineering process [21, 13, 23, 4, 28]. A *risk* is commonly defined as an uncertain factor whose occurrence may result in some loss of satisfaction of some corresponding objective. A risk has a *likelihood of occurrence*, and one or several undesirable *consequences* associated with it. Each consequence is uncertain as well; it has a likelihood of occurrence if the risk occurs. A consequence has a *severity* in terms of degree of loss of satisfaction of the corresponding objective. Depending on the category of objective being obstructed, risks may correspond to safety hazards [26, 27], security threats [2, 22], inaccuracy conditions on software input/output variables with respect to their environment counterpart [21], and so forth.

Risks must be identified, assessed against likelihood and severity, and controlled through appropriate countermeasures [7, 17, 23, 28]. At requirements engineering time, risks can be systematically *identified* from prescriptive requirements and descriptive domain properties [21]. For risk *assessment*, we can use qualitative scales to support quick but rough estimates of likelihood and severity [12] (e.g., from 'unlikely' to 'very likely' and from 'low' to 'highly critical', respectively), possibly in relation with a requirements model [4]. Alternatively, quantitative scales can be used to capture such estimates more precisely [6, 13], possibly in relation with a requirements model [31]. For risk *control*, we may explore alternative countermeasures and select most effective ones [13]; such exploration may be driven by risk-reduction tactics such as reduce risk likelihood, avoid risk, reduce consequence likelihood, avoid risk consequence, or mitigate risk consequence [23].

In goal-oriented modeling frameworks, obstacles were introduced as a natural abstraction for risk analysis [3, 20]. An *obstacle* to a goal is a precondition for the non-satisfaction of this goal. Obstacle analysis [21] consists of (a) *identifying* as many obstacles as possible to every leaf goal in the goal refinement graph from relevant domain properties; (b) *assessing* the likelihood and severity of each obstacle; and (c) *resolving* likely and critical obstacles by systematic model transformations encoding the preceding risk-reduction tactics in order to integrate appropriate countermeasures in the goal model. Obstacle analysis has been successfully used in a variety of mission-critical systems, see, e.g., [29, 11].

The risk/obstacle assessment step is obviously crucial for focusing the resolution step on those risks that are determined to be likely and have likely and severe consequences. No systematic techniques are available to date to support this step.

To fill this gap, the paper presents a simple yet effective technique for quantitative risk assessment. This technique is intended to meet the following objectives.

- *Formal semantics for statements to be assessed:* Unlike [13, 4, 28], the specification of goals and risks should have a clear, precise semantics in terms of desirable/undesirable system behaviors. Such semantics enables their precise interpretation and the integration

of risk assessment with other techniques for risk generation [21, 1], countermeasure derivation [21] and goal model analysis [23], including goal refinement/operationalization checking or behavior model synthesis.

- *Measurable statements:* Unlike [13, 4], the specification of goals and risks should be grounded on application-specific phenomena that are measurable in the environment of the software-to-be; this attenuates the common problems with subjective estimations. For the importance of making requirements measurable, see [30].
- *Model-based assessment*: Unlike [13], the assessment process should take advantage of the refinement structure provided by the goal/obstacle model to allow for more accurate estimation of probabilities of coarser-grained statements from finer-grained ones.
- *Probabilistic requirements:* Unlike existing techniques, requirements that prescribe some property to hold in at least *X*% of the cases should be handled within the same assessment framework.

Partially satisfied goals were introduced in [25] for evaluating alternative system options. The degree of satisfaction of such goals is modeled there by objective functions on quality variables they refer to. The goals are specified formally and interpreted in terms of application-specific measures. Their associated quality variables are refined according to the refinement structure of the goal model. Degrees of satisfaction are determined bottom-up by computing the probability density function of higher-level quality variables from the probability density functions of lower-level ones. This results in accurate estimations at the price of fairly complex computations. Bayesian networks might also be used for making predictions about partially satisfied assertions [14]; their construction and validation however does not take advantage of the available goal structure, and turn to be very difficult for complex systems.

Our technique for determining the probability of obstacles and the probability and severity of their consequences is intended to be simpler as it exploits the goal/obstacle refinement structure and propagates probabilities directly along that structure.

As a result, obstacles get prioritized by degree of criticality. Such prioritization can then be used for guiding the selection among alternative countermeasures identified through risk-reduction tactics [21]. We thereby obtain more evidence-based answers to questions such as, e.g., what are the most critical obstacles to be resolved in view of the high-level, safety-critical goal stating that 'an ambulance shall be on the incident scene within 14 minutes in 95% of cases'?

The paper is organized as follows. Section II introduces some necessary background on goal-oriented modeling and obstacle analysis. Section III introduces our model-based probabilistic framework for goals, obstacles, and their refinements. Section IV shows how obstacle probabilities are up-propagated through obstacle refinement trees and how obstacle consequences are up-propagated through the goal model. Section V discusses the identification of critical obstacle combinations to be resolved, based on a prioritization of obstacles according to the severity of their consequences. Section VI summarizes our evaluation of the technique on a carpooling support system. Section VII discusses related work.

II. BACKGROUND

Goal-Oriented System Modeling. A *goal* is a prescriptive statement of intent to be satisfied by the agents forming the system. The word *system* refers to the software-to-be together with its environment, including pre-existing software, devices such as sensors and actuators, people, etc. Unlike goals, *domain properties* are descriptive statements about the problem world (such as physical laws).

A goal may be *behavioral* or *soft* dependent on whether it can be satisfied in a clear-cut sense or not. In the context of risk analysis, this paper focuses on behavioral goals.

A behavioral goal captures a maximal set of intended behaviors declaratively and implicitly; a *behavior* is a sequence of system state transitions. A behavior thus violates a goal if it is not among those prescribed by the formal specification of the goal [21].

Linear temporal logic (LTL) may be used for formalizing behavioral goals to enable their analysis. The goals then take the general form

$$C \Rightarrow \Theta\, T$$

where Θ represents a LTL operator such as: $\bigcirc$ (in the next state), $\Diamond$ (sometimes in the future), $\Diamond_{\leq d}$ (sometimes in the future before deadline d), $\Box$ (always in the future), $\Box_{\leq d}$ (always in the future up to deadline d), W (always in the future unless), U (always in the future until), and where $P \Rightarrow Q$ means $\Box\,(P \rightarrow Q)$. The following standard logical connectives are used: $\wedge$ (and), $\vee$ (or), $\neg$ (not), $\rightarrow$ (implies), $\leftrightarrow$ (equivalent).

A behavioral goal can be of type *Achieve* or *Maintain/Avoid*. The specification pattern for an *Achieve* goal is "**if C then sooner-or-later T**", that is, $C \Rightarrow \Diamond T$, where C denotes a *current* condition and T a *target* condition, with obvious particularizations to *Immediate Achieve*, *Bounded Achieve* and *Unbounded Achieve* goals. The pattern for a *Maintain* (resp. *Avoid*) goal is "**[if C then] always G**" (resp. "**[if C then] never B**", that is, $[C \Rightarrow]\,\Box G$ (resp. $[C \Rightarrow]\,\Box \neg B$), where G and B denote a *good* and *bad* condition, respectively.

A behavioral goal must obviously be consistent with all known domain properties, that is,

$$\{G, Dom\} \not\models false \qquad \text{(domain-consistency)}$$

A *goal model* is an AND/OR graph showing how goals contribute positively or negatively to each other [10, 15, 23]. Parent goals are obtained by abstraction whereas child goals are obtained by refinement. In a goal model, leaf goals are assigned to single system agents; they are *requirements* or *assumptions* dependent on whether they are assigned to the software-to-be or to an environment agent, respectively.

Refinement patterns are available to help building goal models, e.g., the *Milestone-Driven*, *Case-Driven*, *Guard-Introduction*, *Unmonitorability-Driven*, *Uncontrollability-*

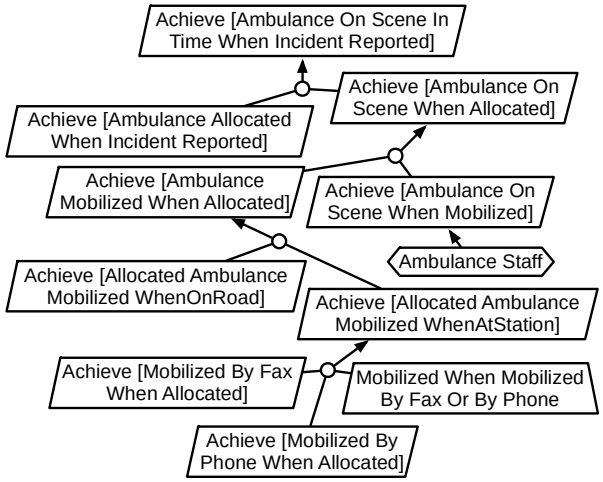


Fig. 1. Partial goal model for an ambulance dispatching system

Driven, or *Divide-and-Conquer* patterns [10, 23]. Fig. 1 illustrates two milestone-driven refinements (top), one case-driven refinement (middle) and one refinement fitting no specific pattern (bottom). For example, the top goal is:

$$\text{IncidentReported} \Rightarrow \Diamond_{\leq 14\ \text{min}}\ \text{AmbulanceOnScene}$$

Its two subgoals are obtained by application of the Milestone-Driven refinement pattern with AmbulanceAllocated as milestone condition:

$$\text{IncidentReported} \Rightarrow \Diamond_{\leq 1\ \text{min}}\ \text{AmbulanceAllocated}$$
$$\text{AmbulanceAllocated} \Rightarrow \Diamond_{\leq 13\ \text{min}}\ \text{AmbulanceOnScene}$$

AND-refinement links in a goal model should ideally be complete, consistent and minimal. A refinement is *complete* if the satisfaction of all subgoals is sufficient for the satisfaction of the parent goal in view of known domain properties:

$$\{SG_1, \ldots, SG_n, Dom\} \models G \qquad \text{(complete refinement)}$$

A refinement is *consistent* if no subgoal contradicts other sub-goals in the domain:

$$\{SG_1, \ldots, SG_n, Dom\} \not\models \text{false} \qquad \text{(consistent refinement)}$$

A refinement is *minimal* if all the subgoals are needed for the satisfaction of the parent goal:

$$\text{for all i: } \{SG_1, \ldots, SG_{i-1}, SG_{i+1}, \ldots SG_n, Dom\} \not\models G$$

The partial goal model in Fig. 1 shows four complete, consistent and minimal AND-refinements.

Obstacle analysis. An *obstacle* to a goal is a domain-satisfiable precondition for the non-satisfaction of this goal [21]:

$$\{O, Dom\} \models \neg G \qquad \text{(obstruction)}$$
$$\{O, Dom\} \not\models \text{false} \qquad \text{(domain consistency)}$$

Similarly to goals, obstacles can be AND/OR refined into sub-obstacles, resulting in a goal-anchored form of risk tree. In such tree, the root obstacle is the negation of the associated leaf goal in the goal model; an AND-refinement captures a combination of sub-obstacles entailing the parent obstacle; an OR-refinement captures alternative ways of entailing the parent obstacle —and, recursively, of obstructing the corresponding leaf goal; the leaf sub-obstacles are single, fine-grained obstacles whose likelihood can be easily estimated.

Each sub-obstacle in an OR-Refinement must entail the parent obstacle:

$$\{SO_i, Dom\} \models O \quad \text{for all } SO_i \qquad \text{(entailment)}$$

OR-Refinements should ideally be domain-complete and disjoint:

$$\{\neg SO_1, \ldots, \neg SO_n, Dom\} \models \neg O \qquad \text{(domain-completeness)}$$
$$\{SO_i, SO_j, Dom\} \models \text{false} \quad \text{for } SO_i \neq SO_j \qquad \text{(disjointness)}$$

Formal and heuristic techniques are available for the identification of obstacles [21, 1] and for the generation of alternative countermeasures [21]. In particular, for *Achieve* and *Maintain/Avoid* goals, specific domain properties are worth eliciting. They take the form "**if T then N**" or "**if G then N**", that is, $\mathbf{T} \Rightarrow \mathbf{N}$ or $\mathbf{G} \Rightarrow \mathbf{N}$, where ***N*** denotes a *necessary* condition for the target condition ***T*** or good condition ***G***. They result in obstacles taking the form "**sooner-or-later** [**C and**] ***never*** **N**" or "**sooner-or-later** [**C and**] **sooner-or-later** ***not*** **N**", that is, $\Diamond([\mathbf{C} \ni] \Box \neg \mathbf{N})$ or $\Diamond([\mathbf{C} \ni] \Diamond \neg \mathbf{N})$, respectively. For example, consider the goal Achieve [AmbulanceOnScene WhenMobilized] in Fig. 1:

$$\text{AmbulanceMobilized} \Rightarrow \Diamond_{\leq 11\ \text{min}}\ \text{AmbulanceOnScene}$$

Negating this goal yields the root obstacle:

$$\Diamond\ (\text{AmbulanceMobilized} \ni \Box_{\geq 11\ \text{min}} \neg\ \text{AmbulanceOnScene})$$

The necessary conditions for the target include the following:

$$\text{AmbulanceOnScene} \Rightarrow \neg\ \text{AmbulanceInTrafficJam}$$

This yields the bottom left sub-obstacle in Fig. 2, namely:

$$\Diamond\ (\text{AmbulanceMobilized} \ni \Box_{\geq 11\ \text{min}}\ \text{AmbulanceInTrafficJam})$$

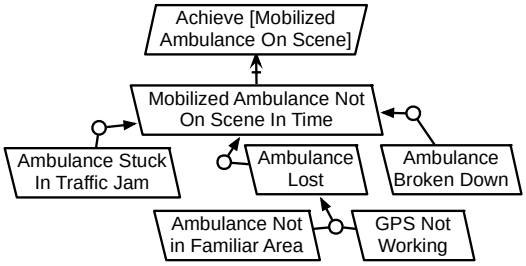


Fig. 2. Partial obstacle model for ambulance dispatching system

III. A Model-Based Framework for Capturing Probabilistic Goals and Obstacles

The probability of satisfaction of a goal depends on the probability of occurrence of obstacles obstructing it. The severity of the consequences of an obstacle depends on the difference between the *prescribed* degree of satisfaction for the obstructed goals and the *estimated* probability of satisfaction of these goals in view of their obstruction. This section defines these various notions more precisely.

A. Probabilistic Goals

As seen before, a (non-probabilistic) goal defines a maximal set of intended behaviors. The probability of goal

satisfaction is defined in terms of the probability of observing one of those behaviors.

For a behavioral goal $C \Rightarrow \Theta T$, we are obviously interested in *non-vacuous satisfaction*, leaving aside those trivial cases where the goal is satisfied because C = false. We therefore focus our attention on behaviors where the goal antecedent C is satisfied.

Definition 1. The *probability of satisfaction of a goal* is the proportion between (a) the number of possible behaviors satisfying the goal's antecedent C and consequent ΘT and (b) the number of possible behaviors satisfying the condition C.

A goal is thus *fully satisfied* if its probability of satisfaction is equal to 1.

Consider the goal Achieve [AmbulanceMobilizedWhenAllocated] in Fig. 1. Its probability of satisfaction is defined as:

$$\frac{\text{Nr. of behaviors where allocated ambulance is mobilized}}{\text{Nr. of behaviors where ambulance is allocated}}$$

Assuming there are 3 possible behaviors where an allocated ambulance is mobilized out of 4 possible behaviors where the ambulance is allocated, the probability of satisfaction for that goal is 75%.

Note that the set of behaviors satisfying this goal does not necessarily satisfy or deny the goal Achieve [Ambulance AllocatedWhenIncidentReported]; whether an allocated ambulance is mobilized or not does not depend on whether an ambulance is allocated or not. On another hand, the goals Achieve [AmbulanceMobilizedWhenAllocated] and Achieve [AllocatedAmbulanceMobilizedWhenOnRoad] are not independent as the set of behaviors satisfying the latter also satisfies the former.

Definition 2. Two goals are *dependent* if the set of behaviors non-vacuously satisfying one of them is also non-vacuously satisfying or denying the other.

Two goals are independent if they are not dependent. In terms of conditional probabilities, this amounts to saying that:

$$P(G_1 \mid G_2) = P(G_1 \mid \neg G_2) = P(G_1),$$
$$P(G_2 \mid G_1) = P(G_2 \mid \neg G_1) = P(G_2),$$

where $P(G)$ denotes the probability of satisfaction of G and $P(G|H)$ denotes the probability of satisfaction of G over all behaviors satisfying property H.

Our behavioral goals are structured in an AND/OR refinement graph [23]. It can then be shown that:

- two goals are dependent if one of them is a child or descendant of the other, or if it conflicts directly or indirectly with the other;
- in a minimal and consistent goal refinement, the subgoals are independent.

In our probabilistic framework, goals will be annotated with an *estimated* probability of satisfaction and a *required* degree of satisfaction.

Definition 3. The *estimated probability of satisfaction* (*EPS*) for a goal is the probability of satisfaction of this goal in view of its possible obstructions. It is computed from the goal/obstacle models.

In our running example, an allocated ambulance might not be mobilized for various reasons, e.g., the ambulance crew being not responsive, the ambulance being not ready for the next mission, communication failing, etc. Due to all such obstacles, there is a probability of this goal not being satisfied.

The *EPS* for a goal G will be denoted $P(G)$ in the sequel; $P(G_1, G_2)$ will denote the *EPS* of G_1 and G_2 in combination.

Definition 4. The *required degree of satisfaction* (*RDS*) for a goal is the minimal probability of satisfaction admissible for this goal. It is imposed by elicited requirements, existing regulations or standards, and the like.

For example, ORCON standards require ambulances to be on the incident scene within 14 minutes in 95% of cases [24]. This will be captured by annotating the goal Achieve[Ambulance OnSceneInTimeWhenIncidentReported] with a *RDS* of 0.95.

Note that the previous situation of (non-probabilistic) goals recalled in Section II is generalized here; for such goals we have $RDS(G) = 1$.

Annotating a behavioral goal $C \Rightarrow \Theta T$ in a goal model with its RDS amounts to specifying it in a probabilistic temporal logic [18] through an assertion of form $C \Rightarrow Pr_{\geq RDS}[\Theta T]$.

Definition 5. A goal G is *probabilistic* if $0 < RDS(G) < 1$.

Based on a goal's *EPS* and *RDS*, we can measure the gap between its estimated and prescribed probabilities. If EPS $\geq$ RDS, the goal's required satisfaction threshold is reached; if EPS < RDS, it is not and we have a problem. This gap should be as low as possible. The difference allows us to measure how severe the goal violation is.

Definition 6. The *severity of violation* of a goal G is defined by:

$$SV(G) = RDS(G) - P(G).$$

The *domain-consistency* condition introduced in Section II is generalized accordingly; it now states that there is a chance to observe a behavior at least that satisfies the goal and the domain properties:

$$P(G \mid Dom) > 0$$

In our generalized setting for goals with a partial degree of satisfaction, we need to state what desirable goal refinements are. The completeness, consistency and minimality conditions in Section II are therefore generalized accordingly.

A refinement of goal G into subgoals $SG_1, ..., SG_n$ is now said to be *complete* if:

$$P(G \mid SG_1, \ldots, SG_n, Dom) > 0 \quad \text{(complete refinement)}$$

Note that this condition is weaker than the completeness condition in Section II as it accounts for partial satisfaction; it covers in particular the case of full satisfaction, equivalent to the completeness condition in Section II:

$$P(G \mid SG_1, \ldots, SG_n, Dom) = 1$$

The refinement is *consistent* if:

$$P(SG_1, \ldots, SG_n \mid Dom) > 0 \quad \text{(consistent refinement)}$$

The refinement is *minimal* if for all i:

$$P(G \mid SG_1, \ldots, SG_{i-1}, SG_{i+1}, \ldots, SG_n, Dom) < P(G \mid SG_1, \ldots, SG_n, Dom) \quad \text{(minimal refinement)}$$

B. Probabilistic Obstacles

A goal is partially satisfied because obstacles to it can occur. Consider the goal Achieve [AmbulanceMobilizedWhenAllocated] in Fig. 1 whose precise specification is:

$$\text{AmbulanceAllocated} \Rightarrow \Diamond_{\leq 2\ \text{min}}\ \text{AmbulanceMobilized}$$

There is a domain property stating that a *necessary condition* for ambulances to be mobilized is that their ambulance crew must be responsive:

$$\text{AmbulanceMobilized} \Rightarrow \text{CrewResponsive}$$

By regression [21] of the goal negation through this domain property, we obtain the obstacle AmbulanceCrewNot Responsive:

$$\Diamond\ (\text{AmbulanceAllocated} \ni \Box_{\geq 2\ \text{min}} \neg\ \text{CrewResponsive})$$

This condition captures the situation of an ambulance being sooner or later allocated without subsequent crew response for 2 minutes. It will be called *obstacle condition*. Such conditions should hopefully not be satisfied too often.

Definition 7. The *probability of an obstacle* is the probability of satisfaction of the obstacle condition, that is, the proportion between (a) the number of possible behaviors satisfying the obstacle condition and (b) the number of possible system behaviors.

The probability of an obstacle *O* will be denoted by *P(O)*. The probability of *O* over all behaviors satisfying some property *H* will be denoted by *P(O|H)*.

The obstruction and domain-consistency conditions recalled in Section II must be generalized in this probabilistic setting.

The *obstruction* condition now states that there is a chance that the obstacle will violate the goal:

$$P(\neg G \mid O, Dom) > 0 \qquad \text{(obstruction)}$$

Note again that this condition is weaker than the obstruction condition in Section II as it accounts for partial obstruction; it covers in particular the case of full obstruction, equivalent to the obstruction condition in Section II:

$$P(\neg G \mid O, Dom) = 1$$

The *domain-consistency* condition states that there is a chance for the obstacle to occur:

$$P(O \mid Dom) > 0 \qquad \text{(domain consistency)}$$

In our generalized setting, we need to characterize what obstacle refinements are. The conditions on obstacle refinement introduced in Section II are therefore generalized accordingly.

For an AND-refinement, the *completeness*, *consistency* and *minimality* conditions are similar to those introduced in Section III.A for probabilistic goals.

For an OR-refinement, the counterpart of the *entailment* condition in Section II now states that if one of the sub-obstacles occurs then the parent obstacle may occur:

$$P(O \mid SO_i) > 0 \quad \text{for all } SO_i \qquad \text{(entailment)}$$

This condition is again weaker than the entailment condition in Section II; it covers the particular case of full satisfaction, equivalent to the entailment condition in Section II:

$$P(O \mid SO_i) = 1$$

For example, for the top OR-Refinement in Fig. 2 we have:

$$P(\text{MobilizedAmbulance}\mathbf{Not}\text{OnSceneInTime} \mid \text{AmbulanceLost}) > 0,$$

as a mobilized ambulance lost might not be on the incident scene within 11 minutes.

The generalized condition for an OR-refinement to be *domain-complete* now states that the parent obstacle cannot be satisfied through further sub-obstacles:

$$P(O \mid \neg\ SO_1, \ldots, \neg\ SO_n, Dom) = 0 \qquad \text{(domain completeness)}$$

In our running example, our domain knowledge might allow us to state "if a mobilized ambulance is not stuck in traffic jam nor lost nor broken down, it will reach the incident scene within 11 minutes"; in such case we would have:

$$P(\text{MobilizedAmbulance}\mathbf{Not}\text{OnScene} \mid \neg\ \text{StuckInTrafficJam}, \neg\ \text{AmbulanceLost}, \neg\ \text{AmbulanceBrokenDown}) = 0.$$

The disjointness condition on sub-obstacles in Section II is generalized into an independence condition:

$$P(SO_i \mid SO_j) = P(SO_i \mid \neg\ SO_j) = P(SO_i),$$
$$P(SO_j \mid SO_i) = P(SO_j \mid \neg\ SO_i) = P(SO_j)$$

In our example, the probability of an ambulance being broken down does not depend on, e.g., the probability of the ambulance being lost or stuck in a traffic jam.

Note that two dependent obstacles can be captured through three independent obstacles: one where the first obstacle condition holds but not the second, one where the second obstacle condition holds but not the first, and one where both hold. Each of these can have a different probability.

IV. Evaluating Obstacles and Their Consequences

This section shows how obstacle probabilities are computed from the obstacle refinement model and how the probabilities of their consequences are computed from the goal refinement model.

The estimated probabilities of leaf obstacles are to be obtained first. Such estimates are up-propagated in obstacle refinement trees (Section IV.A); the results are propagated from root obstacles to leaf goals in the goal model (Section IV.B); the results are in turn up-propagated in the goal model to obtain probabilities of consequences in terms of goal obstructions at various levels of abstraction.

A. From Leaf Obstacles to Root Obstacles

We first need to rely on domain knowledge to obtain estimated probabilities of leaf obstacles in refinement trees – typically, through statistical data about past system behaviors (cf. Definition 7). For the leaf obstacle

$$\Diamond\ (\text{AmbulanceMobilized} \ni \Box\ \neg\ \text{AmbulanceInFamiliarArea})$$

in Fig. 2, such data might reveal that the situation of mobilized ambulances being in unfamiliar areas occurs in 20% of the cases; for the leaf obstacle

$$\Diamond\ (\text{AmbulanceMobilized} \ni \Box\ \neg\ \text{GPSWorking})$$

we might get from such data that the situation of the GPS not working inside mobilized ambulances occurs in 10% of the cases.

Such estimates are to be up-propagated in obstacle trees. In an *AND-refinement*, a parent obstacle may occur if all its sub-obstacles occur. The probability of the parent obstacle is therefore the probability that each sub-obstacle occurs and

their combined occurrence leads to the satisfaction of the parent obstacle:

$$P(O) = P(SO_1) \times P(SO_2) \times ... \times P(O \mid SO_1, SO_2, ...)$$

Back to our example, we thus also need to know from statistical data how often does an ambulance in unfamiliar area with non-working GPS get lost – e.g., in 95% of the cases.

In an *OR-refinement*, a parent obstacle may occur if any of the sub-obstacles occurs. The probability of the parent obstacle is therefore the probability that any of the child obstacle occurs and leads to the satisfaction of the parent obstacle. In this case we cannot simply sum the probabilities of each sub-obstacle occurring and leading to the satisfaction of the parent obstacle; we would then need to remove the probability of different sub-obstacles occurring in combination. To overcome this problem, we consider the probability of the parent obstacle *not* occurring, which equals the probability of no child obstacle occurring that would lead to the satisfaction of the parent obstacle. For a complete and disjoint refinement, this leads to:

$$P(O) = 1 - (1 - P(SO_1) \times P(O \mid SO_1)) \times (1 - P(SO_2) \times P(O \mid SO_2)) \times \ldots$$

The preceding formulas for AND- and OR-refinements are recursively applied bottom-up through the refinement tree until the probability of the root obstacle is obtained.

Consider our obstacle model in Fig. 2 with the above statistical data about leaf obstacles, namely, 20% of mobilized ambulances are sent to unfamiliar areas, 10% of GPS inside mobilized ambulances are not working, and 95% of mobilized ambulances in unfamiliar areas with non-working GPS get lost. The propagation rule for AND-refinements yields the following probability for the parent obstacle AmbulanceLost:

$$P(\text{AmbulanceLost}) = (0.2 \times 0.1) \times 0.95 = 0.019$$

Assume now that statistical data tell us that 2% of mobilized ambulances get stuck in traffic jam, 0.5% of mobilized ambulances break down, and the proportion of lost, stuck or broken ambulances not reaching the incident scene within 11 minutes is 99%, 98%, and 100%, respectively. The propagation rule for OR-refinements yields the following probability for the root obstacle MobilizedAmbulance**Not**OnSceneInTime:

$$P(\text{MobilizedAmbulance}\mathbf{Not}\text{OnSceneInTime}) = 1 - (1 - 0.019 \times 0.99) \times (1 - 0.02 \times 0.98) \times (1 - 0.005 \times 1) = 0.0429,$$

which means that a mobilized ambulance will not arrive on the incident scene within 11 minutes in 4.29% of cases.

B. From Root Obstacles to Obstructed Leaf Goals

In standard risk analysis, a risk consequence is expressed in terms of degree of loss of satisfaction of the associated objective. This is translated in our framework by saying that the *consequence* of an obstacle is the lower degree of satisfaction of the obstructed leaf goal and, recursively, of its parent and ancestor goals.

The probability of non-satisfaction of the leaf goal LG is given by the probability that the root obstacle RO occurs and that such occurrence actually leads to the non-satisfaction of the leaf goal (see the *obstruction* condition in Section III.B):

$$1 - P(LG) = P(RO) \times P(\neg LG \mid RO)$$

Back to our running example, we can thereby compute the reduced probability of satisfaction for the leaf goal Achieve [AmbulanceOnSceneWhenMobilized] in Fig. 1. As the obstacle MobilizedAmbulance**Not**OnSceneInTime always obstructs the goal, we obtain:

$$P(\text{Achieve[AmbulanceOnSceneWhenMobilized]}) = 1 - 0.0428 \times 1 = 0.957$$

This means that in more than 95% of cases, a mobilized ambulance will arrive on scene within the prescribed 11 minutes.

If the leaf goal can be obstructed by more than one obstacle, it will be satisfied when none of these occurs:

$$P(LG) = (1 - P(O_1) \times P(\neg LG \mid O_1)) \times (1 - P(O_2) \times P(\neg LG \mid O_2)) \times \ldots$$

C. From Obstructed Leaf Goals To Higher-Level Goals

The decreased degree of satisfaction of the obstructed leaf goal must be up-propagated in the goal refinement graph in order to determine all obstacle consequences. The probability of satisfaction of a parent goal depends on the probabilities of its subgoals. Without loss of generality, the presentation will consider refinements in two subgoals for sake of clarity.

As introduced in Section III.A, in the most general case the parent goal is satisfied if the two subgoals are satisfied, or the satisfaction of the first is sufficient for satisfying the parent, or the satisfaction of the second is sufficient for satisfying the parent. This leads to the following general propagation rule for AND-refinements:

$$\begin{aligned} P(G) = {} & P(SG_1, SG_2) \times P(G \mid SG_1, SG_2) \\ & + P(SG_1, \neg SG_2) \times P(G \mid SG_1, \neg SG_2) \\ & + P(SG_2, \neg SG_1) \times P(G \mid SG_2, \neg SG_1) \\ & + P(\neg SG_1, \neg SG_2) \times P(G \mid \neg SG_1, \neg SG_2) \end{aligned}$$

In case we focus our attention on a *single* system, no alternative OR-refinements are to be considered; the probability of satisfying *G* given that none of the subgoals is satisfied is then equal to zero, and the last term disappears. Moreover, in case where the refinement meets the non-probabilistic completeness condition in Section II, we have that $P(G \mid SG_1, SG_2) = 1$. The AND-propagation rule then reduces to:

$$\begin{aligned} P(G) = {} & P(SG_1, SG_2) \\ & + P(SG_1, \neg SG_2) \times P(G \mid SG_1, \neg SG_2) \\ & + P(SG_2, \neg SG_1) \times P(G \mid SG_2, \neg SG_1) \end{aligned} \qquad \text{(and-propag)}$$

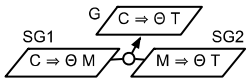
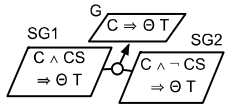
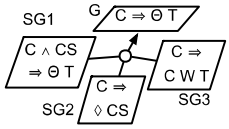
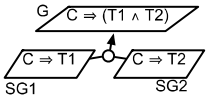
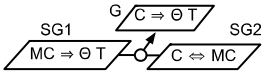
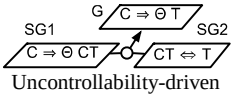
Depending on the type of refinement and goal, this propagation rule can be made further specific. Table 1 gives propagation rules for a sample of common refinement patterns known to be complete, consistent and minimal [10]; the subgoals there are therefore independent.

For a *milestone-driven* refinement, for example, the satisfaction of a single milestone-based subgoal is not sufficient for satisfying the parent goal. The propagation rule therefore reduces to

$$P(G) = P(SG_1) \times P(SG_2)$$

For a *case-driven* refinement, the parent goal is satisfied when one of the subgoals is satisfied. If *P(CS)* denotes the

TABLE I. PROPAGATION RULES FOR COMMON REFINEMENTS

Pattern	Equation
 Milestone-driven	P(G) = $P(SG_1) \times P(SG_2)$
 Case-driven	P(G) = $P(CS) \times P(SG_1)$ + (1 - P (CS) × $P(SG_2)$ Refinements are annotated with P(CS)
 Guard-introduction	P(G) = $P(SG_1) \times P(SG_2) \times P(SG_3)$
 Divide-and-conquer	P(G) = $P(SG_1) \times P(SG_2)$
 Unmonitoribility-driven	P(G) = $P(SG_1) \times P(SG_2)$
 Uncontrollability-driven	P(G) = $P(SG_1) \times P(SG_2)$

probability of satisfying the case-condition *CS*, assuming two disjoint cases, the propagation rule becomes:

$$P(G) = P(CS) \times P(SG_1) + (1 - P(CS)) \times P(SG_2)$$

To evaluate obstacles consequences, we can proceed in two ways:

- *Global impact analysis:* the computed probabilities for all obstructed leaf goals are together propagated bottom-up in the goal graph to see how much the resulting *EPS* of higher-level goals deviates from their required *RDS*.
- *Local impact analysis:* the consequence of a single leaf goal obstruction is evaluated by up-propagation of the computed probability for this leaf goal, all other leaf goals being assigned a probability of 1 (meaning that they are all assumed to be fully satisfied).

Let us illustrate such global impact analysis on the model in Fig. 1. (For lack of space we will use precise goal names instead of their formal specification.) We want to know whether this model satisfies the threshold imposed by the ORCON standard; the latter requires the goal Achieve [AmbulanceOnSceneInTimeWhenIncidentReported] to be satisfied in at least 95% of cases.

For the leaf goal Achieve[AmbulanceOnSceneWhenMobilized], the probability of satisfaction computed in Section IV.B is *0.956*. Similar computations for the other leaf goals in Fig.1 yield:

Achieve [AmbulanceAllocatedWhenIncidentReported]: *0.98*,
Achieve [AllocatedAmbulanceMobilizedWhenOnRoad]: *0.98*,
Achieve [MobilizedByFaxWhenAllocated]: *0.90*,
Achieve [MobilizedByPhoneWhenAllocated]: *0.95*,
Achieve [MobilizedByFaxOrByPhoneWhenAllocated]: *1*
(no obstacle found in the obstacle model).

To obtain the probability for the parent goal Achieve [AllocatedAmbulanceMobilizedWhenAtStation], we use the general *and-propag* rule as the refinement does not fit any pattern. In this rule, we have here $P(G \mid \neg SG_i, SG_j) = 1$; for example, MobilizedWhenAtStation is satisfied given that MobilizedByFax is satisfied. The simplified rule then yields:

P (Achieve [AllocatedAmbulanceMobilizedWhenAtStation]) =
$0.9 \times 0.95 + 0.10 \times 0.95 + 0.05 \times 0.9 = 0.995$

We can now compute the probability of satisfying the goal Achieve [AmbulanceMobilizedWhenAllocated]. Its refinement in Fig. 1 is a case-driven refinement; the corresponding simplified propagation rule can therefore be used. The case condition *CS* is *AllocatedAmbulanceAtStation*; statistical data tell us that this condition holds in 60% of cases. We therefore obtain:

P (Achieve [AmbulanceMobilizedWhenAllocated]) =
$= 0.60 \times 0.995 + 0.40 \times 0.98 = 0.984$

We can then continue the up-propagation and compute the probability of satisfying the goal Achieve [AmbulanceOnSceneWhenAllocated] in Fig. 1. Its refinement is a milestone-driven one; the associated propagation rule is therefore used. This leads to:

P (Achieve [AmbulanceOnSceneWhenAllocated]) =
$0.984 \times 0.957 = 0.9466$

Finally, we reach the top goal in Fig. 1. Its refinement is a milestone-driven one as well. The same propagation rule yields:

P (Achieve[AmbulanceOnSceneInTimeWhenIncidentReported]) =
$0.98 \times 0.9466 = 0.9277$

The resulting EPS for this goal is thus 92.77%; the system as modelled is thus not able to satisfy the ORCON standard prescribing 95%. The next section discusses how the critical obstacles can be identified for higher-priority resolution in a new version of the model.

V. IDENTIFYING CRITICAL OBSTACLE COMBINATIONS

Countermeasures must be deployed at RE time or at system runtime in order to resolve probabilistic goal violations. Such countermeasures can be explored according to risk reduction tactics [21, 23]. To select most appropriate ones at modeling time or at runtime, we need to identify the most problematic leaf obstacles.

There is a multi-criteria optimization problem here as we are looking for minimal sets of leaf obstacles that maximize the severity of goal violations, where $SV(G) = RDS(G) - P(G)$.

To achieve this, we can generate all possible leaf obstacle combinations. The violation severity $SV(G)$ is then computed for each obstructed goal *G*. If these goals have different priorities, we can weight differently the computed $SV(G)$ according to their respective priority. The most critical combinations are identified by sorting the leaf obstacle combinations by violation severity (possibly weighted).

Consider the obstacles in Fig. 2. There are 4 leaf obstacles and 8 possible combinations (as two leaf obstacles

are involved in an AND-Refinement). Table II shows the computed SVs for all these combinations; a value "1" (resp. "0") indicates that the corresponding obstacle is (resp. is not) in the combination. Fig. 3 visualizes the computed violation severity *SV* corresponding to each combination. The squares represent leaf obstacle combinations that differ in size; the black ones indicate the most critical combinations for a given size. As we can see there, two combined leaf obstacles are sufficient for falling under the goal's RDS; a single obstacle cannot obstruct the goal enough. From Table 2 we can make the following further observations.

- The possibility of an ambulance being lost or stuck in traffic jam is sufficient for severe obstruction of the goal; this is the pair to resolve first.
- The two other pairs cause a significantly smaller loss in satisfaction of our top-level goal.

The set of black squares in Fig. 3 defines a Pareto front; efficient algorithms for generating them are available [19, 8]. Our generation of leaf obstacle combinations and their ranking by severity can thereby be optimized in order to scale up for larger systems.

VI. VALIDATION

The techniques in this paper were also used for risk analysis of a carpooling support system. A brief description follows.

> The system should act as a marketplace for drivers to offer empty seats in real time and travellers to use them under agreed conditions. A driver is matched in real time with anyone searching for a ride along a common route. Effective carpooling may critically depend on marketplace size; the system should therefore be attractive to drivers, in particular by not overconstraining them. Drivers are assumed to have a GPS-based navigation device and a PDA/iPhone-like touch screen.

Our goal model for this system includes 32 goals, 15 refinements up to 7 levels. A variety of refinements patterns were used [23]. The obstacle model includes 75 obstacles, among which 42 leaf obstacles. The complete report for the case study can be found at [9]. For lack of space, we only present fragments of our study together with some lessons learnt; precise goal names are used here again instead of their formal specification.

A top behavioral goal for this system is Achieve [NeedForRideServed]; it prescribes that at "least 95% of passengers making a request for a ride will arrive at destination within the specified time constraints". This high-level goal is refined in three subgoals: Achieve[RideRequestEncoded], Achieve[AdequateDriverFoundWhenRequestEncoded], and Achieve [PassengerAtDestinationWhenDriverFound]. Each of these is refined towards assignable requirements or assumptions. For example, the leaf goal Achieve [DropPointReachedWhen PassengerInCar] states that "a passenger of a planned ride inside the driver's car shall arrive at the drop point"; the leaf goal Achieve [RidePlannedWhenProposalSelected] states that "a ride shall eventually be planned when passenger has selected a proposal".
Obstacles to such leaf goals were generated and refined. Here is a sample of refinements in textual format together with their probability - estimated for leaf goals and computed for non-leaf goals.

Drop Point Not Reached When Passenger In Car *12,4%*

TABLE II. Violation severity for Achieve [AmbulanceOnSceneInTimeWhenIncidentReported]

Amb. Lost	Amb. Stuck In Traffic	Amb. Broken Down	EPS	RDS	SV
1	1	1	92,77%	95%	**2,23%**
1	1	0	93,20%		**1,80%**
0	1	1	94,54%		**0,46%**
1	0	1	94,61%		**0,39%**
0	1	0	95,02%		-0,02%
1	0	0	95,10%		-0,10%
0	0	1	96,44%		-1,44%
0	0	0	96,92%		-1,92%

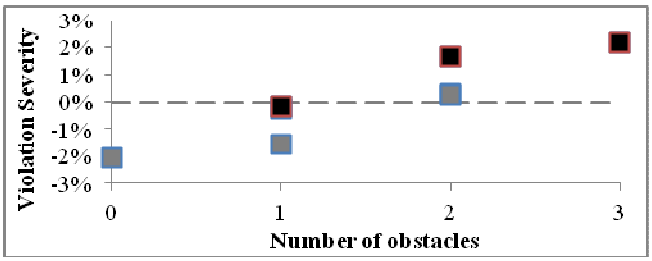


Fig. 3. Obstacle combination ranking by violation severity

Drop Point Not Reached In Time	*3,8%*
Car Stuck In Traffic Jam	*3,4%*
Driver Has Wrong Instructions	*0,1%*
Detour From Planned Road	*0,2%*
Other Passenger Late	*0,1%*
Drop Point Never Reached	*8,6%*
Car Broken Down	0,5%
Driver Gets Lost	3%
Driver Has Wrong Instructions	0,1%
Driver At Other Drop Point	2%
Drop Point Inaccessible	3%
Ride Not Planned While Proposal Selected	4,55%
Ride Canceled	1,95%
Cancelled By Driver	1%
Cancelled By Passenger	0,95%
No Pickup Or Drop Point Found	2,67%
No Pickup Point Found	0,7%
No Pickup Point Accessible To Riders	0,7%
No Drop Point Found	0,7%
No Drop Point Accessible To Riders	0,7%
No Pickup Or Drop Time Found	1,2%
No Pickup Time Found	0,6%
Pickup Time Incompatible With Other Pickups	0,2%
Journey Longer Than Proposed Time	0,2%
No Drop Time Found	0,6%
…	

The estimates for leaf obstacles are grounded on the domain; they can be elicited from experienced users, statistical data or runtime measures from existing software applications. For example, the probability of the leaf obstacle DriverForgetsToGoToPickupPoint was estimated to 1% (10 rides out of 1000). The probability of the leaf obstacle RidersDoNotRecognizeEachOther was estimated to 0,9% whereas the probability of the more frequent leaf obstacle WrongContactInformation was estimated to 6%**.**

Combinations involving a large number of obstacles causing the high-level goal to fall under its RDS were not

felt really helpful in identifying the obstacles to focus on. Some of the singletons and obstacle pairs we found were potentially causing significant goal violations already. For example, the obstacle WrongContactInformation with probability 6% produced a SV of 1% (meaning that only 94% of passengers would have a successful ride). The obstacle DriverGetsLost with probability 3% produced a SV of 0.91%. The obstacle PassengerLateAtPickupPoint with probability 5,5% produced a SV of 0.5%.

Among the 904 pairs of leaf obstacles identified as critical (out of 1722 pairs), the 3 most critical pairs not including the previous obstacles were:

- (PassengerGetsLost, RideCancelledByPassenger), SV = 4.78%;
- (PassengerGetsLost, DriverLateAtPickupPoint), SV = 4.75%;
- (DriverLateAtPickupPoint, RideCancelledByPassenger), SV = 4.74%

These three obstacle combinations should thus have higher priority for resolution.

Some of the leaf obstacles appeared to have more importance than others, even if they have a small probability. For example, the obstacle PassengerCancelsRide, with a probability of 0,85%, was seen to potentially obstruct many leaf goals, e.g., PassengerAtPickupPointWhenInstructionsKnown, RideInstructionsReadWhenSent, ProposalSelectedFromSuggestions, ProposalRemainsMatchingIfSelected, etc. Even if the estimated probability is low, the obstacle might be critical. For example, if the probability of PassengerCancelsRide is changed from 0,85% to 0.95%, the violation severity increases from -0.10% to 0.57%.

To sum up, the large number of obstacles made it quite difficult to identify what most critical obstacles should be considered first for selecting resolutions. The prioritized list of obstacles produced by our technique helped significantly in that direction.

VII. Related Work

Probabilistic fault trees are sometimes used for analyzing undesirable events in safety-critical systems [26, 6]. Low-level events are annotated with probabilities that are propagated by use of ad hoc rules. As they are informal and not model-based, such trees provide little support for identifying root events, refining trees incrementally and reasoning in terms of higher-level objectives.

CORAS is a UML-based risk modeling methodology that relates assets and risks annotated with likelihoods to support quantitative reasoning in the *identify-assess-control* cycle of risk analysis [28]. Such likelihoods and their contributions however remain fairly vague from a semantic standpoint.

DDP is a lightweight, tool-supported technique for quantitative risk analysis [13]. Goals, obstacles and countermeasures are called requirements, failure modes and PACTs, respectively. Each goal is decorated with a weight representing its importance. Failure modes are annotated with likelihoods. Countermeasures are decorated with an effectiveness defined as the proportion of risk reduction. Criticality and loss of objective are then characterized as arithmetic combinations on these annotations. Our technique builds on DDP by adding formality to specifications, a more precise semantics for probabilities grounded on application-specific phenomena, and a model-based refinement structure for propagating probabilities through risks and consequences at various levels of granularity.

The TROPOS goal-oriented framework is also closely related to our efforts. It puts more focus on modeling soft goals and reasoning qualitatively about their contributions. TROPOS has been extended to support some form of quantitative reasoning [15], risk assessment [4], and evaluation of system performance indicators [5]. In [4], goals are called assets and are related to external events that influence positively or negatively goal satisfaction or denial, respectively. Influences and degrees of satisfaction are assessed quantitatively or qualitatively. The quantitative approach in [15] on which [4] and [5] are based also relies on model-based propagation rules. However, the considered goals and risks have no precise semantics in terms of system behaviors; they are not measurable. The probabilities therefore cannot be grounded on behavioral semantics. Moreover, the propagations do not take advantage of specific types of refinement. There seems to be no risk AND/OR-refinement structure for propagating probabilities from fine-grained risks that are easier to estimate in terms of application-specific phenomena. Lastly, probabilistic goals are not supported in terms of estimated vs. required probabilities of satisfaction.

In [31], KAOS goal models are extended with probabilities and propagation rules for technology qualification. Their rules appear different; they do not take advantage of different refinement types. For example, their AND-propagation rule does not apply to case-driven or non-minimal refinements; their OR-propagation rule for obstacles can be made simpler thanks to obstacle disjointness. Probabilistic goals as defined in our work are not introduced in [31]; they seem not relevant to their context.

In [25], goals are annotated with random variables they refer to and associated objective functions; such variables are bound by equations that are tailored to corresponding refinements. Probability density functions are propagated bottom-up to assess alternative goal refinements. This technique is more precise and finer-grained, but more heavyweight. It is targeted at selecting alternative options rather than prioritizing obstacles by criticality.

VIII. Conclusion

The quantitative risk assessment technique presented in the paper is model-based and anchored on an existing goal-oriented framework for requirements engineering. The framework is extended with a probabilistic layer allowing behavioral goals to be characterized in terms of their *estimated* and *required* degrees of satisfaction. The specification of such goals and their obstacles has a formal semantics in terms of system behaviors, allowing probabilities to be grounded on measurable, application-specific phenomena. The severity of obstacle consequences in terms of degree of goal violation is determined quantitatively and systematically by probability propagations through the obstacle and goal models. The most critical obstacle combinations are then determined in order to

prioritize obstacles and guide the exploration of appropriate countermeasures against the more critical obstacles, using available techniques [21], to increase requirements completeness.

Our technique was successfully applied to two non-trivial mission-critical systems for ambulance dispatching and carpooling, respectively.

The use of Markov chains as semantic models of goal/obstacle specifications is currently being investigated to improve the accuracy of probability estimations. Dedicated tool support is also under development to replace our current spreadsheet calculations and integrate them in semi-formal [23] and formal [1] goal-oriented RE environments. The transposition of our framework from behavioral goals to measurable soft goals is also worth considering.

A next step concerns the assessment of the cost-effectiveness of countermeasures and their integration in the goal model. The handling of uncertainty over probabilities is another issue. Domain experts tend to provide ranges for estimating probabilities; measurements can contains errors; knowledge about certain probabilities might be missing. Such uncertainty needs to be integrated as well.

ACKNOWLEDGMENT

This work was supported by the European Fund for Regional Development and the Walloon Region. Thanks are due to Bernard Lambeau, Christophe Damas and Simon Busard for inspiring discussions, and to the reviewers for their comments calling for clarifications.

REFERENCES

[1] D. Alrajeh, J. Kramer, A. van Lamsweerde, A. Russo and S. Uchitel, "Generating Obstacle Conditions for Requirements Completeness", *Proc. ICSE'2012: 34th Intl. Conf. on Software Engineering*, Zürich, May 2012.

[2] E.J. Amoroso, *Fundamentals of Computer Security*. Prentice Hall, 1994.

[3] A. Anton and C. Potts, "The Use of Goals to Surface Requirements for Evolving Systems", *Proc. ICSE'1998: Intl. Conf. on Software Engineering*, Kyoto, May 1998, 157-166.

[4] Y. Asnar, P. Giorgini and John Mylopoulos, "Goal-driven Risk Assessment in Requirements Engineering", *Req. Eng. Journal* 16(2), June 2011, 101-116.

[5] D. Barone, L. Jiang, D. Amyot, J. Mylopoulos, "Reasoning with Key Performance Indicators", PoEM 2011, LNBIP Vol. 92, 2011, 82-96.

[6] T. Bedford and R. Cooke, *Probabilistic Risk Assessment - Foundations and Methods*. Cambridge University Press, 2001.

[7] B.W. Boehm, "Software Risk Management: Principles and Practices," *IEEE Software*, Jan./Feb. 1991, 32-41.

[8] S. Börzsönyi, D. Kossmann and K. Stocker, "The Skyline Operator", Proc. IEEE 17th Intl. Conf. on Data Engineering., Washington, 2001, 421-430.

[9] www.info.ucl.ac.be/~acaillia/publications/carpoolingsystem.html.

[10] R. Darimont and A. van Lamsweerde, "Formal Refinement Patterns for Goal-Driven Requirements Elaboration", *Proc. FSE'4 - Fourth ACM SIGSOFT Symp. on the Foundations of Software Engineering*, San Francisco, October, 1996,179-190.

[11] R. Darimont and M. Lemoine, "Security Requirements for Civil Aviation with UML and Goal Orientation", *Proc. REFSQ'07 – International Working Conference on Foundations for Software Quality*, Trondheim (Norway), LNCS 4542, Springer-Verlag, 2007.

[12] US Department of Defense, *Procedures for Performing a Failure Mode Effect and Criticality Analysis*, Standard MIL-STD-1629A, November 1980.

[13] M.S. Feather and S.L. Cornford, "Quantitative Risk-Based Requirements Reasoning", *Req. Eng. Journal*, 8(4), 2003, 248-265.

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

[15] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani, "Formal Reasoning Techniques for Goal Models", J. Data Semantics 1(1), 2003, 1-20.

[16] W. Heaven and E. Letier, "Simulating and Optimising Design Decisions in Quantitative Goal Models", *Proc. RE 2011: 19th IEEE Intl. Requirements Engineering Conf.*, Trento, Italy, September 2011.

[17] C. Jones, *Assessment and Control of Software Risks*. Yourdon Press, 1994.

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

[19] H. T. Kung, F. Luccio and F. P. Preparata, "On Finding the Maxima of a Set of Vectors", J. ACM 22(4), Oct. 1975, 469-476.

[20] A. van Lamsweerde and E. Letier, "Integrating Obstacles in Goal-Driven Requirements Engineering", *Proc. ICSE-98: 20th International Conference on Software Enginering*, Kyoto, April 1998.

[21] A. van Lamsweerde and Emmanuel Letier, "Handling Obstacles in Goal-Oriented Requirements Engineering", *IEEE Trans. Softw. Eng.* 26(10), October 2000, 978-1005.

[22] A. van Lamsweerde, "Elaborating Security Requirements by Construction of Intentional Anti-Models", *Proc. ICSE'04, 26th Intl. Conf. on Software Engineering*, ACM-IEEE, May 2004, 148-157.

[23] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*, Wiley, January 2009.

[24] *Report of the Inquiry Into the London Ambulance Service*. The Communications Directorate, SW Thames Regional Authority, 1993.

[25] E. Letier and A. van Lamsweerde, "Reasoning about Partial Goal Satisfaction for Requirements and Design Engineering", *Proc. FSE 2004: 12th ACM Symp. on Foundation of Software Engineering*, Newport Beach, CA, November 2004, 53-62.

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

[27] N. Leveson, "An Approach to Designing Safe Embedded Software", *Proc. EMSOFT 2002 – Embedded Software: 2nd International Conference*, Grenoble, LNCS 2491, Springer-Verlag, Oct. 2002, 15-29.

[28] M.S. Lund, B. Solhaug and K. Stølen, Model-Driven Risk Analysis: the CORAS approach. Springer-Verlag, 2011.

[29] R. Lutz, A. Patterson-Hine, S. Nelson, C.R. Frost, D. Tal and R. Harris, "Using Obstacle Analysis to Identify Contingency Requirements on an Unpiloted Aerial Vehicle", *Requirements Engineering Journal* 12(1), 2007, 41-54.

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

[31] M. Sabetzadeh, D. Falessi, L. Briand, S. Di Alesio, D. McGeorge, V. Ahjem and J. Borg, "Combining Goal Models, Expert Elicitation, and Probabilistic Simulation for Qualification of New Technology," *IEEE 13th Intl. Symp. on High-Assurance Systems Engineering (HASE)*, Nov. 2011, 10-12.