Handling Knowledge Uncertainty in Risk-Based Requirements Engineering

Antoine Cailliau and Axel van Lamsweerde

ICTEAM – Institute for Information & Communication Technologies, Electronics and Applied Mathematics
Université catholique de Louvain
Louvain-la-Neuve, Belgium
{antoine.cailliau, axel.vanlamsweerde}@uclouvain.be

***Abstract*—Requirements engineers are faced with multiple sources of uncertainty. In particular, the extent to which the identified software requirements and environment assumptions are adequate and sufficiently complete is uncertain; the extent to which they will be satisfied in the system-to-be is uncertain; and the extent to which obstacles to their satisfaction will occur is uncertain. The resolution of such domain-level uncertainty requires estimations of the likelihood that those different types of situations may or may not occur. However, the extent to which the resulting estimates are accurate is uncertain as well.**

This meta-level uncertainty limits current risk-based methods for requirements engineering. The paper introduces a quantitative approach for managing it. An earlier formal framework for probabilistic goals and obstacles is extended to explicitly cope with uncertainties about estimates of likelihoods of fine-grained obstacles to goal satisfaction. Such estimates are elicited from multiple sources and combined in order to reduce their uncertainty margins. The combined estimates and their uncertainties are up-propagated through obstacle refinement trees and then through the system's goal model. Two metrics are introduced for measuring problematic uncertainties. When applied to the probability distributions obtained by up-propagation to the top-level goals, the metrics allow critical leaf obstacles with most problematic uncertainty margins to be highlighted. The proposed approach is evaluated on excerpts from a real ambulance dispatching system.

***Index Terms*—obstacle analysis; risk assessment; probabilistic goals; requirements completeness; uncertainty management; quantitative reasoning; goal-oriented requirements engineering.**

I. Introduction

Uncertainty is increasingly recognized as a key issue in the engineering of complex software systems such as mobile systems, human-intensive systems, applications embedded to complex environments, smart cyber-physical systems, or self-adaptive systems [22, 37]. The software components in such systems highly depend on their seamless interaction with a wide variety of environment components, including devices and people, whose behavior is uncertain.

Requirements engineers are at the forefront of uncertainty problems. The adequacy and "sufficient" completeness of the identified software requirements, environment assumptions and relevant domain properties is generally uncertain [28]. The satisfaction rate of those requirements and assumptions when the system-to-be will be running is uncertain as well [6, 8, 20, 29]; this is generally due to unexpected events and conditions that may occur in the environment [27]. Moreover, the likelihood of such events and conditions occurring is uncertain.

Uncertainty mitigation techniques are intended to reduce specific types of uncertainty [37]. For requirements engineering (RE), risk analysis cycles are often introduced in the RE process [4, 20, 27, 31]. A *risk* is an uncertain factor preventing the satisfaction of system objectives. Risk analysis can therefore be naturally integrated in goal-oriented RE methods [15, 17, 20, 27, 32].

Obstacle analysis is a goal-oriented form of risk analysis where an *obstacle* to a goal is a precondition for non-satisfaction of this goal. While elaborating a goal model as an AND/OR goal refinement/abstraction graph, the analyst performs obstacle analysis cycles [28]. Each cycle consist of three steps: (a) the *identification* of as many obstacles to leaf goals as possible; (b) the *assessment* of their likelihood and of the severity of their consequences; (c) the *resolution* of likely and severe obstacles through appropriate countermeasures to be integrated as new goals in the goal model. Systematic techniques are available for *identifying* obstacles from requirements and domain properties [2, 27]. For the *assessment* step, likelihoods and criticalities can be determined quantitatively with expert assistance [20] and/or by calculations over probabilistic obstacle/goal models [8, 29]. For obstacle *resolution*, operators are available for exploring alternative countermeasures [27] and for integrating selected countermeasures into the goal model [9].

In traditional risk analysis, uncertainty during risk assessment arises under two forms [23, 40].

- *Physical uncertainty* refers to system phenomena. For example, the chance of a sensor breaking down might be defined by a probability. This domain-level form of uncertainty can be reduced by changing the system under consideration –e.g., by introduction of appropriate countermeasures.
- *Knowledge uncertainty* refers to the assessment of physical uncertainty by experts. In our example, our imperfect knowledge of what the exact probability value is might lead us to estimate it with some uncertainty margin. Such meta-level form of uncertainty can be reduced through further studies, consultation of more experts, increased experience, run-time monitoring of relevant events or data, and so forth.

Risk analysis techniques for RE so far addressed physical uncertainty only [4, 8, 20, 27, 31]. Obstacle analysis, in particular, is intended to anticipate the occurrence of adverse events or conditions by producing a more complete and robust

978-1-4673-6905-3/15 © 2015 IEEE

goal model; the runtime satisfaction rate of requirements and assumptions appearing as leaf goals in the goal model is thereby increased [27, 28].

Current risk analysis techniques for RE do not address knowledge uncertainty. They rely on expert judgement at some point or another to estimate the likelihood of fine-grained events or conditions emerging from their analysis. Such estimates are typically based on experience or historical data [4, 8, 20, 31], sometimes refined through runtime event monitoring [18, 19]. However, the extent to which those estimates are accurate remains uncertain. Expert's judgements might be subjective or biased; relevant data might not be available; accurate data might be too expensive to obtain; collected data might no longer be relevant in view of technology changes; data might be too sparse; and so on [40].

The objective of this paper is to address this current limitation of risk-based RE techniques through quantitative techniques for managing the *knowledge* uncertainty about the estimates being used.

An earlier formal framework for probabilistic goals and obstacles [8] is extended to explicitly capture and reason about uncertainties on estimates of likelihoods of fine-grained obstacles to goal satisfaction. The resulting framework supports specific analyses involving knowledge uncertainty, in particular, the prioritisation of obstacles with respect to the degree and spread of uncertainty they cause on the satisfaction of top-level goals in the goal model. This allows analysts to highlight critical obstacles (in terms of severity of their consequences) whose knowledge uncertainty must be reduced in order to determine whether they are sufficiently likely or not.

The paper makes the following contribution.

- Uncertainties about risk estimates and goal satisfaction rates are integrated in the specification of probabilistic goals and obstacles. In alignment with new-generation reliability databases providing ranges of estimates [1], the extended framework supports both single-point and multi-point value estimates –unlike [4, 8].
- Two metrics are provided for measuring problematic knowledge uncertainties about goal satisfaction.
- A quantitative technique is provided for highlighting the obstacles with most severe consequences on the goal model and most problematic knowledge uncertainties.
- Uncertainty margins on estimates are reduced for increased accuracy by methodical integration of estimates from multiple sources or multiple experts.

The paper is organized as follows. Section II provides necessary rudiments on goal-oriented modeling and probabilistic obstacle analysis. Section III introduces our framework for capturing knowledge uncertainty about goals and obstacles and for measuring problematic uncertainties. Section IV outlines the overall approach for managing knowledge uncertainties. Section V introduces means for reducing knowledge uncertainty about estimates of leaf obstacles by use of multiple sources and/or experts. Section VI details how goal satisfaction rates and uncertainties about them can be computed from those estimates of likelihoods of leaf obstacles and their uncertainty. Section VII briefly discusses how the techniques in the preceding sections may help determine critical obstacles and problematic knowledge uncertainties. Section VIII reports on a preliminary evaluation of our results on a real ambulance dispatching system. Section IX 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 considered system. The word *system* refers to both the software-to-be and its environment, including people, pre-existing software, hardware devices such as sensors and actuators, and so forth. *Domain properties* are descriptive statements about the problem world (such as physical laws).

The paper focuses on *behavioral* goals; unlike softgoals, they can be satisfied in a clear-cut sense [28]. 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 covered by the (formal) goal specification.

A behavioral goal is of type *Achieve* or *Maintain/Avoid* [28]. The specification pattern for an *Achieve* goal is "**if *C* then sooner-or-later *T***", where ***C*** and ***T*** denote a *current* condition and a *target* condition, respectively, 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***"), where ***G*** and ***B*** denote a *good* and *bad* condition, respectively. Linear temporal logic is used for formalizing behavioral goals to enable their analysis. Note that we are only interested in the *non-vacuous satisfaction* of goals, leaving aside the trivial cases where a goal is satisfied because its antecedent is false.

A *goal model* is an AND/OR graph showing how goals contribute positively or negatively to each other [28]. Fig. 1 gives a partial goal model showing how goals are AND-refined for the simple fire alarm system used as a running example. Parent goals are obtained by abstraction, e.g. through *why* questions, whereas child goals are obtained by refinement, e.g. through *how* questions. In a goal model, leaf goals assigned to a software agent are called *requirements* whereas leaf goals assigned to an environment agent are called *expectations*. The latter are prescriptive assumptions on the environment. Descriptive assumptions may be needed as well; they are called *domain hypotheses* –e.g., NoDetectionWhen SmokeDetectorBroken is a domain hypothesis in our example.

Refinement patterns are available for guiding the

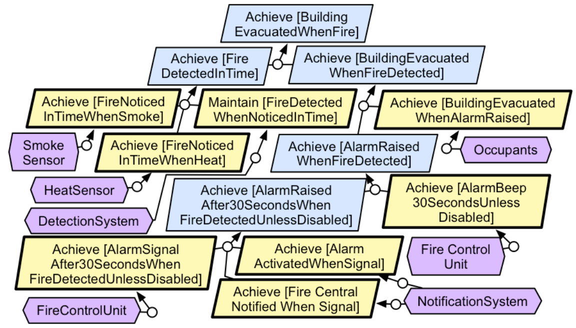
Achieve [Building EvacuatedWhenFire]
Achieve [Fire DetectedInTime]
Achieve [BuildingEvacuated WhenFireDetected]
Achieve [FireNoticed InTimeWhenSmoke]
Maintain [FireDetected WhenNoticedInTime]
Achieve [BuildingEvacuated WhenAlarmRaised]
Smoke Sensor
Achieve [FireNoticed InTimeWhenHeat]
Achieve [AlarmRaised WhenFireDetected]
Occupants
HeatSensor
Achieve [AlarmRaised After30SecondsWhen FireDetectedUnlessDisabled]
Achieve [AlarmBeep 30SecondsUnless Disabled]
DetectionSystem
Achieve [AlarmSignal After30SecondsWhen FireDetectedUnlessDisabled]
Achieve [Alarm ActivatedWhenSignal]
Fire Control Unit
Achieve [Fire Central Notified When Signal]
NotificationSystem
FireControlUnit

Fig. 1. Partial goal model for a fire alarm system

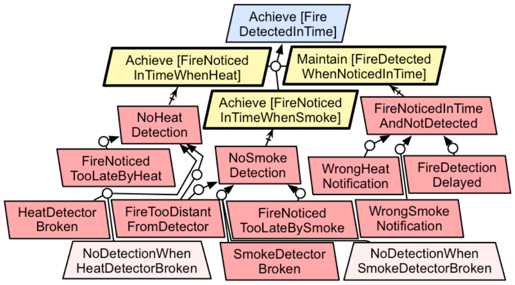
Achieve [Fire DetectedInTime]
Achieve [FireNoticed InTimeWhenHeat]
Maintain [FireDetected WhenNoticedInTime]
NoHeat Detection
Achieve [FireNoticed InTimeWhenSmoke]
FireNoticedInTime AndNotDetected
FireNoticed TooLateByHeat
NoSmoke Detection
WrongHeat Notification
FireDetection Delayed
HeatDetector Broken
FireTooDistant FromDetector
FireNoticed TooLateBySmoke
WrongSmoke Notification
NoDetectionWhen HeatDetectorBroken
SmokeDetector Broken
NoDetectionWhen SmokeDetectorBroken

Fig. 2. Partial obstacle model for leaf goals in Fig. 1

construction of goal models –such as *Milestone-Driven*, *DecompositionByCases*, *Guard-Introduction*, *Divide-And-Conquer*, *Uncontrollability-Driven*, etc. [28]. In Fig. 1, the *Milestone-Driven* refinement pattern is used to refine the top goal Achieve [BuildingEvacuatedWhenFire] into subgoals Achieve [FireDetectedInTime] and Achieve [BuildingEvacuatedWhen FireDetected].

Obstacle analysis. An *obstacle O* to a goal *G* is a satisfiable precondition for the non-satisfaction of this goal [27, 28]:

$$\{O, Dom\} \models \neg G \qquad \text{(obstruction)}$$

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

Like goals, obstacles are AND/OR-refined into sub-obstacles, resulting in a goal-anchored form of risk tree, as Fig. 2 shows. The root obstacle of an obstacle tree 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. The leaf sub-obstacles are fine-grained atomic obstacles whose likelihood can be more easily estimated.

Formal and heuristic techniques are available for the identification of obstacles and for the generation of alternative countermeasures [27, 28]. In particular, domain properties of form "**if *T* then *N***" or "**if *G* then *N***", where ***N*** denotes a *necessary* condition for the target condition ***T*** or good condition ***G***, result in obstacles of form "**sooner-or-later** [***C*** **and**] **never *N***" or "**sooner-or-later** [***C*** **and**] **sooner-or-later not *N***" for *Achieve* and *Maintain/Avoid* goals, respectively. For example, consider the goal Maintain [FireDetectedWhenNoticedInTime] in Fig. 2. Negating this goal yields the root obstacle FireNoticed InTimeAndNotDetected. Regressing through the necessary condition "**if** *FireDetected* **then** *CorrectHeatNotification*" found in the domain results in the obstacle WrongHeatNotification; the derivation is similar for the other sub-obstacles WrongSmokeNotification and FireDetectionDelayed.

Probabilistic goals and obstacles. Behavioral goals might be satisfied only partially. A *probabilistic goal* prescribes some desired condition to be satisfied in at least *X*% of cases [8, 29] –for example, "when a fire is detected the building shall be evacuated within 5 minutes in at least 98% of cases".

The probability of satisfaction of a goal of form "**if *C* then sooner-or-later *T***" is defined as the ratio between (a) the number of possible behaviors satisfying both the goal antecedent ***C*** and consequent **sooner-or-later *T***, and (b) the number of possible behaviors satisfying ***C***.

The *estimated probability of satisfaction* (EPS) of a goal is the probability of its satisfaction in view of its possible obstructions by obstacles.

The *required degree of satisfaction* of goal *G*, denoted by RDS(*G*), is the minimal probability of satisfaction prescribed for this goal. It is imposed by elicited requirements, existing regulations, standards, etc.

The *probability of satisfaction of an obstacle* is defined as the ratio between the number of possible behaviors satisfying the obstacle and the number of possible behaviors.

III. Capturing Knowledge Uncertainties in Goal/Obstacle Models

Probability distributions are commonly used in risk analysis for representing the uncertainty about single-point values of random variables [40]. A *probability distribution* is a function assigning a specific probability to each possible single-point value in some domain. This section extends the probabilistic goal/obstacle specification language in [8] to capture the uncertainty about estimated satisfaction rates of behavioral goals, assumptions, and obstacles in the system-to-be. The domain considered here is the set of possible probability values, that is, the interval [0, 1].

A. Uncertainty about Estimated Satisfaction Rates

Let *A* denote a probabilistic assertion specifying a goal, an assumption (prescriptive or descriptive), or an obstacle in the goal/obstacle models.

Satisfaction uncertainty. The *satisfaction uncertainty* for assertion *A,* denoted by su_A, is defined by a probability distribution over its single-point probabilities of satisfaction.

For example, Fig. 3 shows the probability distribution capturing the satisfaction uncertainty su_G for *G*: Achieve [BuildingEvacuatedWhenFire]. (Section VI describes how such distribution can be computed.) As seen in Fig. 3, the probability of satisfaction of this goal lies between 80% and 90%, with a more likely value around 85%. Note that this multi-value estimate does not meet the goal's RDS, prescribed to be 98% as depicted by the black line on the right.

Satisfaction uncertainties may also refer to domain hypotheses such as NoDetectionWhenSmokeDetectorBroken.

Note that a goal satisfaction uncertainty arises from obstacle satisfaction uncertainties. For example, our certainty about the extent to which the goal Achieve [Fire-NoticedInTimeWhenSmoke] is satisfied depends on our certainty about the extent to which the obstacles FireNoticedToo-LateBySmoke or SmokeDetectorBroken are satisfied.

B. Uncertainty Metrics for Goal Satisfaction

The probability to observe a probability of satisfaction of at least *p* for goal *G*, denoted $SU_G(p)$, is obtained as follows:

$$SU_G(p) = \int_{x<p} su_G(x).$$

This probability corresponds to the area delimited by the su_G curve and value *p*. In Fig. 3, the chance of the goal Achieve

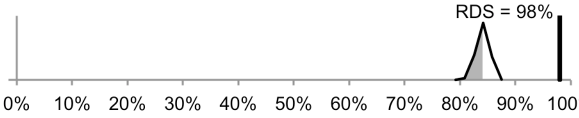

Fig. 3. Satisfaction uncertainty for Achieve [BuildingEvacuatedWhenFire]

[BuildingEvacuatedWhenFire] being satisfied in at least 84% of cases is given by:

$$SU_{Achieve\ [BuildingEvacuatedWhenFire]}(0.84) = 0.78.$$

It corresponds to the grey area in Fig. 3. As shown in Section IV, su_G can be computed by up-propagation from leaf obstacles to their root in obstacle refinement trees and then by up-propagation through the goal model.

Two metrics may be defined to capture (a) our degree of certainty that the goal's RDS will be met; and (b) the spread of satisfaction uncertainty below this RDS.

Goal violation uncertainty. The *violation uncertainty* for a probabilistic goal *G*, denoted by *VU(G)*, is the proportion of satisfaction uncertainty falling below the goal's RDS –that is, the probability of being below this threshold. It is obtained as follows:

$$VU(G) = SU_G(RDS(G)) = \int_{x<RDS(G)} su_G(x).$$

Graphically, this violation uncertainty corresponds to the surface below the satisfaction uncertainty curve up to the goal's RDS. In our example, the violation uncertainty of Achieve [BuildingEvacuatedWhenFire] is *1* (as seen in Fig. 3). This means that it is 100% certain that the goal will not meet its RDS of 98%. In Fig. 3, the curve is completely on the left of the goal's RDS.

Uncertainty spread. Violation uncertainties only capture how much uncertainty lies below the required thresholds. The same surface might take many shapes with more or less spread. Such spread may help making decisions. It might be less problematic to have the uncertainty closer to the RDS than equally spread down to zero. If the uncertainty is close to the goal's RDS, a small change to this RDS might drastically change the violation uncertainty score. The semi-standard deviation of the distribution is used here for measuring spread. It differs from standard deviation as only values below the RDS are taken into account.

The *uncertainty spread* for a goal *G,* denoted by *US(G),* measures the spread of uncertainty below the goal's RDS. It is defined by the semi-standard deviation of its probability distribution with respect to this RDS. For discrete values, this quantity can be computed as follows:

$$US(G) = \sqrt{\frac{\sum_{x_i}\left(x_i - RDS(G)\right)^2}{k}},$$

where x_i are the discrete values of *G*'s satisfaction uncertainty that fall below RDS(*G*)*,* and *k* denotes the number of such values. (This formula can be generalized to continuous satisfaction uncertainties.)

Uncertainty spread values need to be interpreted according to the shape of the curve. Whatever the shape, however, Chebyshev's inequality states that at least 75% of the data are at most at 2 spread values from RDS(*G*) [41]. If the goal's satisfaction uncertainty fits a specific probability distribution, more precise bounds can be obtained.

Back to our example, the goal Achieve [BuildingEvacuated WhenFire] has an uncertainty spread of *0.1645*. According to Chebyshev's inequality, this means that we have at least 75% of the satisfaction uncertainty between 65% and 98%.

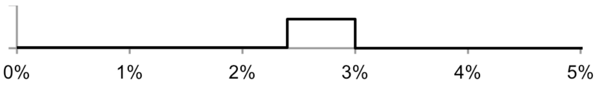

Fig. 4. Satisfaction uncertainty for SmokeDetectorBroken

C. Eliciting Distributions for Leaf Obstacle Satisfaction

To facilitate the elicitation of distribution functions for leaf obstacles from domain experts, discrete points may be used that can be fitted to specific probability distributions –such as Beta, PERT, Triangular, etc. [40]. A *quantile* is a single probability value attached to a cumulative probability. It indicates the cumulative probability to observe a single value of probability of satisfaction [7]. The 50th quantile, called *mode*, is the most likely probability of satisfaction.

To further support such elicitation from domain experts, databases providing estimates for quantiles can be used; they are available in a wide range of domains –e.g., aerospace, health, bank, nuclear, chemical, gas, water pollution, and so forth [14]. Reliable techniques are also available for obtaining accurate single values or distribution estimates from trained experts [23].

For the leaf obstacle SmokeDetectorBroken in Fig. 2, an expert might estimate that there are at least 10% chances of observing at least 24 occurrences of a broken smoke detector out of 1000 fire; at least 50% chances of observing at least 27 such occurrences; and at least 90% chances to observe at least 30 of them. The 10th, 50th and 90th quantiles are 2.4%, 2.7%, and 3% for the probability of satisfaction of this leaf obstacle, respectively. Fig. 4 shows the satisfaction uncertainty for this obstacle. Our estimates were based on reliability databases and published historical data about fire occurrence in various industries [5, 26, 35, 36].

IV. Overall Approach to Uncertainty Management

The satisfaction rate of high-level system goals can be increased by resolving "critical" leaf obstacles through appropriate counter-measures [27, 28] integrated in the goal model [9]. Leaf obstacles with most severe consequences on the goal model should therefore be highlighted while reducing their uncertainty margin.

Our approach for achieving this consists of the following steps:

- The accuracy of estimates for the likelihood of leaf obstacles is increased by combining those elicited from multiple experts and/or data sources (Section V);
- Goal satisfaction rates and their uncertainty margin are computed from those more accurate estimates (Section VI);
- Leaf obstacles are prioritized according to their impact on the satisfaction of top-level goals using the metrics previously introduced (Section VII).

V. Eliciting More Accurate Estimates for Uncertainty Reduction

The first step of our approach consist of eliciting estimates of likelihoods of the leaf obstacles in the obstacle AND/OR refinement trees built during the obstacle identification phase. To reduce uncertainty margins, such estimates should be as adequate and accurate as possible.

The use of multiple sources or multiple experts is generally recognized to increase the accuracy of estimates [13]. Behavioral techniques on multiple experts are often used in practice to reach a consensus towards more accurate estimates [23, 40]. More mathematical approaches are however recognized to produce more accurate results than behavioral ones [12]. The latter are aimed at characterizing expert

TABLE I. EXPERT ESTIMATES FOR CALIBRATION VARIABLES

Variable	Expert	10%	50%	90%	Known value
Temperature SensorBroken	Expert1	2.3%	2.5%	2.8%	4.4%
	Expert2	4.1%	4.7%	5.2%	
FireSprinkler Broken	Expert1	0%	0.1%	0.3%	0.02%
	Expert2	0%	0.1%	0.2%	

judgements by comparative assessment between their estimates and known quantities. The derived characteristics are then used for combining estimates from multiple experts to reduce the uncertainty margin of the value to be estimated. In our context, we want to increase the chance that the satisfaction uncertainty for leaf obstacles is close to the known value.

Obstacles may be annotated with estimated quantiles from multiple experts, e.g.,

Obstacle SmokeDetectorBroken
  Def The smoke detector does not detect fire within 25 sec.
  Probability [Expert1] (2.4%, 2.7%, 3%)
  Probability [Expert2] (2.1%, 2.4%, 2.7%)

Experts are not all equal. Some might systematically over- or under-estimate probabilities of satisfaction; provide very large or very narrow estimates; provide estimate ranges that are accurate or not; and so forth. To get more accurate estimates, we may compare those provided by multiple experts with known values in order to characterize each expert. This is known as *calibration* [7].

A *calibration variable* is a quantity whose exact single-point value is known. In our context, a calibration variable should be closely related to a leaf obstacle –typically, a leaf obstacle whose satisfaction rate is known, an agent/resource failure whose failure rate is known from reliability databases [1], etc. In our running example, three calibration variables might be identified: PushButtonBroken, TemperatureSensorBroken and FireSprinklerBroken. Table I summarizes estimates and known values for two of them. The *X*% columns show the quantiles estimated by the corresponding expert for the calibration variable.

Two techniques are available for characterizing multiple experts and combining their estimated quantiles. We instantiate them to our context in order to combine multiple quantiles on a leaf obstacle into a single satisfaction uncertainty.

Cooke's technique characterizes each expert through a single weight [13]. This weight combines a calibration score and an information score. The *calibration score* measures how close the expert's estimate is to the known value of the calibration variable; the higher the probability estimated for this value, the higher the score. The *information score* measures how precise the estimates are; the narrower the estimates, the greater the score. The satisfaction uncertainty combining the quantiles from multiple experts is obtained as a weighted sum of these quantiles. In our example, Expert1 gets an information score of *0.4216*, a calibration score of *0.9305*, and a weight of *0.4528*; Expert2 gets an information score of *0.4740*, a calibration score of *1*, and a weight of *0.5471*. Figs. 5(a) and 5(b) show the quantile function of both experts; Fig. 5(c) shows the resulting satisfaction uncertainty.

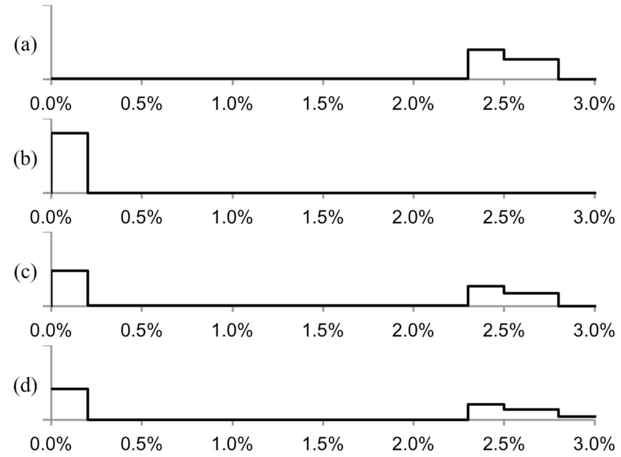

Fig. 5. Satisfaction of BatteryOutOfService: (a) Expert1's estimate, (b) Expert2's estimate, (c) combined estimates using Cooke's technique, (d) combined estimates using Mendel-Sheridan's technique

TABLE II. EXPERT ESTIMATES FOR LEAF OBSTACLES

Expert	Obstacle	10%	50%	90%
Expert1	BatteryOutofService	2.3%	2.5%	2.8%
	BellBroken	1.4%	1.6%	1.8%
	SmokeDetectorBroken	2.4%	2.7%	3%
Expert2	BatteryOutofService	0%	0.1%	0.2%
	BellBroken	1.7%	1.9%	2.1%
	SmokeDetectorBroken	2.1%	2.4%	2.7%

Mendel-Sheridan's technique combines the quantiles of the experts by use of a Bayesian calibrator/estimator [34]. It first computes a minimally informative distribution for the expert's characteristics. This a priori distribution considers each expert to be unbiased; it does not weight any probability more than others. This a priori distribution is then updated with respect to the expert's quantiles and the known value of calibration variables. The resulting distribution is used to combine the quantiles of the experts into a satisfaction uncertainty for our leaf obstacle. Fig. 5(d) shows the resulting satisfaction uncertainty using this technique.

Table II illustrates possible estimates for leaf obstacles and two experts. These estimates are then combined using the two techniques. Which technique performs best remains an open question; it may depend on the application domain, the experts, and the number of calibration variables [13].

VI. COMPUTING GOAL SATISFACTION RATES AND THEIR UNCERTAINTY

The second step in the overall approach outlined in Section IV consists of determining the satisfaction rate of the goals in our goal model, with their uncertainty margins, from the likelihood estimates of leaf obstacles obtained in the previous step.

Computing uncertainty distributions for top-level goals analytically appears unfeasible in practice; it involves heavyweight computing machinery. Monte-Carlo simulation may be used instead. The simulation relies on repeated sampling of parameters to obtain numerical results [40].

The procedure for computing the satisfaction rate and uncertainty for a top-level goal *G* may be outlined as follows:

1. Compute the set of AND-combinations of leaf obstacles obstructing *G*;
2. For each leaf obstacle in this set, select a single-point probability value according to the obstacle's satisfaction rate and uncertainty;
3. Based on this sample, compute the probability of satisfaction for the top-level goal *G*;
4. Repeat the process multiple times and aggregate the single-point probability values obtained for the top-level goal to produce its satisfaction rate and uncertainty.

A. Computing Obstruction Sets

For a given goal G, we first compute all AND-combinations of leaf obstacles and domain hypotheses that may obstruct G. An *obstruction set* for a goal G is a set of obstacles O_i and domain hypotheses DH_j such that:

$$\{O_1, O_2, \ldots DH_1, DH_2, \ldots\} \models \neg G \qquad \text{(obstruction)}$$

$$\{O_1, O_2, \ldots DH_1, DH_2, \ldots\} \not\models \text{false} \qquad \text{(consistency)}$$

An obstruction set OS should be minimal, that is, all its elements are required for falsifying the goal:

$$\text{for all } O_i \text{ in } OS: \ OS \setminus O_i \not\models \neg G$$

$$\text{for all } DH_j \text{ in } OS: \ OS \setminus DH_j \not\models \neg G \qquad \text{(minimality)}$$

For example, the goal Achieve [FireNoticedInTimeWhenHeat] can be obstructed by the following three obstruction sets:

{FireTooDistantFromDetector},
{HeatDetectorBroken, NoDetectionWhenHeatDetectorBroken},
{FireNoticedTooLateByHeat}.

A goal can have multiple alternative obstruction sets. The OR-combination of all alternative obstruction sets for a goal G is called *obstruction superset* for G, denoted by OS (G). The alternatives there should ideally be independent:

$$\text{for all } OS, OS' \text{ in } OS(G): \quad OS \cap OS' = \varnothing \qquad \text{(independence)}$$

Dependences thus arise when obstruction sets share common obstacles or domain hypotheses.

The obstruction superset for a goal is computed by up-propagation from leaf obstacles and domain hypotheses in obstacle trees to the root obstacle; then from the root obstacle to the obstructed leaf goal; and finally from leaf goals in goal trees to the considered top-level goal.

Step 1: From leaf obstacles to root obstacles. Consider the obstacle AND/OR refinement tree anchored on leaf goal LG in the goal model. To obtain the obstruction superset OS(LG), we proceed by structural induction. Let OS (LG|O) denote the obstruction superset for LG obtained by considering all obstacles and domain hypotheses in the obstacle sub-tree rooted on O.

- For a leaf obstacle or domain hypothesis *LO:*

  $$OS(LG|LO) = \{LO\}.$$

- For an *AND*-refinement of O in sub-obstacles O_1 and O_2:

  $$OS(LG|O) = OS(LG|O_1) \times OS(LG|O_2),$$

  where × denotes the cartesian product over sets (the generalization to more sub-obstacles is straightforward). For example, the obstruction set for the refinement of the obstacle NoHeatDetection in Fig. 2 is {HeatDetectorBroken, NoDetectionWhenHeatDetectorBroken}.

- For an *OR*-refinement of O in sub-obstacles O_1 and O_2:

  $$OS(LG|O) = OS(LG|O_1) \cup OS(LG|O_2).$$

  Back to our example, the three obstruction sets for the obstacle FireNoticedInTimeAndNotDetected in Fig. 2 are {WrongHeatNotification}, {WrongSmokeNotification}, and {FireDetectionDelayed}.

Step 2: From root obstacles to leaf goals. The obstruction superset for a leaf goal LG obstructed by a root obstacle RO is:

$$OS(LG) = OS(LG|RO).$$

Step 3: From leaf goals to top goals. The obstruction superset obtained for leaf goal LG is propagated bottom-up along the AND-refinement trees in which LG is involved. For a parent goal PG AND-refined into sub-goals G_1 and G_2 we have:

$$OS(PG) = OS(G_1) \cup OS(G_2).$$

The obstruction sets in the superset thereby obtained are not necessarily independent. This arises from obstacle refinement trees sharing common obstacles, resulting in a non-empty intersection of obstruction sets. Such dependencies must be taken into account when computing the probability of satisfaction of obstruction sets; this can be achieved automatically, see Section VI.B. By construction, independent obstruction sets are minimal.

In our example, the obstruction superset for the goal Achieve [FireDetectedInTime] may be computed from the obstruction supersets obtained for the sub-goals Achieve [FireNoticedInTime WhenSmoke], Achieve [FireNoticedInTimeWhenHeat], and Maintain [FireDetectedWhenNoticedInTime]. Recursively, the obstruction sets for the root goal Achieve [BuildingEvacuatedWhenFire] may then be computed.

The procedure in this section notably differs from the propagation algorithm in [8] as it propagates uncertainties about estimates. Step 3 here is the counterpart of the algorithm in [8]. However, it does not rely on goal refinement patterns and is therefore more general. Moreover, there is only a single propagation here through the obstacle and goal models which makes Step 3 much more efficient.

An obstruction superset somewhat corresponds to the cut set of a fault tree [7, 28]. A first difference is that a cut set yields all combinations of leaf events causing the root event to occur whereas an obstruction superset yields all combinations of leaf obstacles causing the corresponding goal in the goal model to be obstructed. The propagation must therefore continue bottom-up through the goal model with specific propagation rules to assess the severity of obstruction consequences –see Step 2 and Step 3, not found in Fault Tree Analysis. Another difference is that the tree nodes here are formalizable goal/obstacle specifications, linked by entailment relationships among levels, rather than event labels.

B. Computing Single-Value Satisfaction Rates

From the computed obstruction superset for a top-level goal in the goal model and initial single-point values for satisfaction of the leaf obstacles, a single-point probability value is computed for the satisfaction of this goal.

An obstruction set is *satisfied* if all its obstacles and domain hypotheses are satisfied. An obstruction superset is s*atisfied* if at least one of its obstruction sets is satisfied.

The probability of satisfaction of a goal G is given by the probability that its obstruction superset is not satisfied:

$$P(G) = 1 - Pr[OS(G)],$$

where Pr [OS(G)] denotes the probability that the obstruction superset OS(G) is satisfied.

To compute the probability of satisfaction of an obstruction superset, a binary decision diagram (BDD) is built that represents the corresponding Boolean formula where each leaf obstacle and domain hypothesis appears as a variable. This Boolean formula encodes the AND/OR-combination of leaf obstacles and domain hypotheses through the disjunction of the conjunction of elements in each obstruction set. As the ordered BDD is canonical, equivalent formulas result in the same BDD; this makes specific treatments of dependent obstruction sets unnecessary. In our example, for the goal Achieve [FireNoticedInTimeWhenHeat] and its three obstruction sets given before, the corresponding Boolean formula is:

FireTooDistantFromDetector
∨ (HeatDetectorBroken ∧ NoDetectionWhenHeatDetectorBroken)
∨ FireNoticedTooLateByHeat.

Efficient algorithms are available for building compact BDDs [16, 33]. Such BDDs enable fast computation of single-point probability values from single-point probability values for the variables [7]. Fig. 6 shows a BDD corresponding to the preceding formula. Each node represents a leaf obstacle. By following the solid edges if the leaf obstacle is satisfied or the dotted edges otherwise, we can determine whether the corresponding Boolean formula is *true* or *false*. The terminal nodes indicate whether the formula is satisfied (*1*) or not (*0*). For example, if HeatDetectorBroken and NoDetectionWhenDetector Broken are both satisfied, the obstruction set is satisfied since the BDD path ends at node (*1*).

Every edge in the BDD has a probability label. The latter is, for a solid edge, the probability of satisfaction of the obstacle at its source and, for a dotted edge, *1* minus this probability. For a terminal node, the probability is given by its value (*0* or *1*).

The probability of a non-terminal node represents the probability that the formula corresponding to the sub-tree is satisfied. It is given by the product of the probability on the edge and the probability of the target node.

For example, let us assume the following satisfaction probabilities: *0.009* for obstacle FireNoticedTooLateByHeat; *0.004* for obstacle FireTooDistantFromDetector; *0.005* for obstacle HeatDetectorBroken; and *0.98* for the domain hypothesis NoDetectionWhenHeatDetectorBroken. (The next section shows how such single-point values are obtained from their respective probability distributions.) Based on these single-point values, the probability of the node FireNoticedTooLateByHeat is given by:

0.991 × 0 + 0.009 × 1 = 0.009.

The probability of the node FireTooDistantFromDetector is:

0.996 × 0.009 + 0.004 × 1 = 0.013.

We can then compute the probability for the node NoDetectionWhenHeatDetectorBroken:

0.02 × 0.013 + 0.98 × 1 = 0.9802.

We finally obtain the probability for the node HeatDetectorBroken:

0.995 × 0.013 + 0.005 × 0.9802 = 0.0178.

From there we derive the probability of satisfaction for the goal Achieve [FireNoticedInTimeWhenHeat]:

P (Achieve [FireNoticedInTimeWhenHeat]) = 1 – 0.0178 = 98.21%.

C. From Single-Value Satisfaction Rates to Satisfaction Uncertainties

Our procedure so far computes single-point probability values for goal satisfaction from single-point probability values for satisfaction of the obstructing leaf obstacles. To build a complete probability distribution for a top goal in the goal model, we need to sample the satisfaction uncertainty, as defined in Section III.A, for these leaf obstacles. To sample a satisfaction uncertainty, a random number between *0* and *1* is uniformly picked. The inverse of the cumulative distribution function associated with the satisfaction uncertainty is then used to get a corresponding probability of satisfaction. More likely probabilities of satisfaction are thereby picked more often than less likely ones.

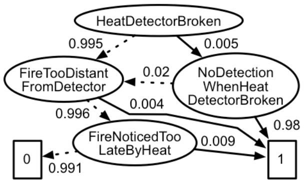
HeatDetectorBroken
0.995
0.005
FireTooDistant FromDetector
0.02
NoDetection WhenHeat DetectorBroken
0.996
0.004
0.98
FireNoticedToo LateByHeat
0.009
0
0.991
1

Fig. 6. BDD for obstruction superset of Achieve [FireNoticedInTimeWhenHeat]

The probability of satisfaction of a top goal is then computed for a given sample (as detailed in Section IV.A and IV.B). The sampling is repeated a large number of times to obtain a set of probabilities of satisfaction for this top goal. The obtained set of probabilities can then be aggregated into a distribution by using their frequency. For example, by repetitive sampling for the leaf obstacles, the following probabilities of satisfaction for the top goal Achieve [BuildingEvacuatedWhenFire] might be obtained:

0.82, 0.84, 0.848, 0.846, 0.821, 0.839, 0.841, …

By counting the number of times each probability occurs, we can build the distribution shown in Fig. 1.

The number of sampling required depends on the satisfaction uncertainties for the leaf obstacles, the numerical precision to achieve, the time available to solve the problem, and so forth.

VII. Finding Critical Obstacles and Uncertainties

The third step in the overall approach outlined in Section IV consists of prioritizing obstacles according to the metrics introduced in Section III.B, that is, the resulting violation uncertainty and uncertainty spread for top-level goals.

The impact of each single leaf obstacle on the goal model must be assessed individually. For a given top goal, the procedure in the previous section is applied for each leaf obstacle with the satisfaction probability of all other leaf goals being set to *0*. The violation uncertainty and the uncertainty spread for the considered top goal are then computed according to their definition in Section III.B. The result may be represented on a scatter plot, called *violation diagram*, to highlight the most critical obstacles to this top goal. When all likely and critical individual obstacles are resolved, the violation uncertainty and uncertainty spread may be computed again with pairs of leaf obstacles in order to build a new violation diagram; then with triples, and so forth.

Our running example contains 13 leaf obstacles. The violation uncertainty and uncertainty spread for the top goal Achieve [BuildingEvacuatedWhenFire] were seen before to be 1 and 0.1529, respectively. Fig. 7 shows the corresponding violation diagram. The following observations can be made from it.

- The obstacle WrongSmokeNotification is clearly a critical one. The violation of the top goal it causes is certain.
- The top goal violation caused by SmokeDetectorBroken and BatteryOutofService is certain; however the uncertainty spread is very low. This indicates that the uncertainty is probably close to the goal's RDS. This obstacle might thus not be that critical as a slight change in the goal's RDS might drastically change the violation uncertainty.
- The leaf obstacles FireCentralNotificationDelayed, FireDetection Delayed, and NetworkDown are not causing the highest

TABLE III. VIOLATION UNCERTAINTY AND UNCERTAINTY SPREAD

Obstacle	Violation Uncertainty	Uncertainty Spread
WrongSmokeNotification	1	0.0196
SmokeDetectorBroken	1	0.0076
BatteryOutofService	1	0.0058
FireDetectionDelayed	0.686	0.0091
FireCentralNotificationDelayed	0.6661	0.0090
NetworkDown	0.6231	0.0119
FireNoticedTooLateBySmoke	0.1621	0.0003

violation uncertainty or uncertainty spread.

- For FireLatelyNoticedBySmoke, there is some good confidence that the obstacle will not prevent the root goal from reaching its RDS as the violation uncertainty is quite low.
- For the remaining 6 obstacles, it is certain that they do not individually prevent the root goal from reaching its RDS.

Table III shows the violation uncertainty and the uncertainty spread for the relevant leaf obstacles.

The obstacle prioritization technique in [8] does not support uncertainties about estimates. Moreover, the technique here proceeds iteratively on the size of obstacle combinations.

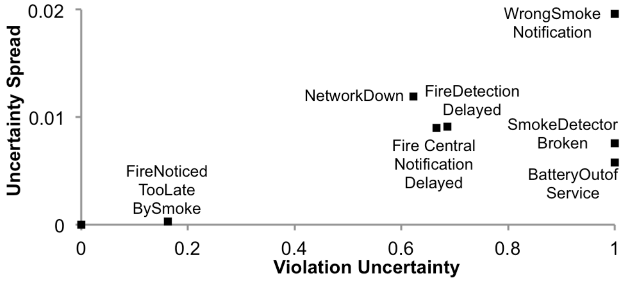

Fig. 7. Violation Diagram for Achieve [BuildingEvacuatedWhenFire]

VIII. EVALUATION

Our approach was applied on a model of a real ambulance dispatching system deployed in the Brussels area. This model is partly based on the one used for evaluating obstacle analysis techniques [2, 27] while building upon the first author's experience as a voluntary paramedic in this type of system. The model contains 71 goals, 19 goal refinements and 8 agents. The obstacle model contains 76 obstacles; among these 47 are leaf obstacles whose likelihood has to be estimated. The quantitative aspects used in our evaluation were based on real data provided by stakeholders. For confidentiality reasons, the presentation here is based on fictitious but realistic data.

The top goal in this model is Achieve [IncidentResolved] with a RDS of 95%. The *milestone-driven* refinement pattern produces three sub goals: Achieve [IncidentReported], Achieve [AmbulanceIntervention], and Achieve [IncidentResolvedByAmbulanceIntervention]. The goal Achieve [AmbulanceIntervention] states that "*an ambulance shall be on the incident scene within 10 minutes*" [39]. At a lower level, the goal Achieve [AllocatedAmbulanceMobilization] is further refined using a *case-driven* refinement pattern into Achieve [AllocatedAmbulanceMobilizationAtStation] and Achieve [AllocatedAmbulanceMobilizationOnRoad]. This refinement yields two domain hypotheses annotated with their estimated probability of satisfaction:

If AmbulanceAllocated **then** AmbOnRoad P = 0.4
If AmbulanceAllocated **then** AmbAtStation P = 0.6

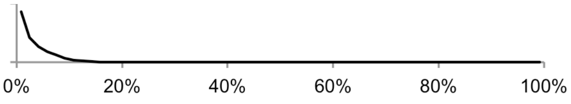

Fig. 8. Satisfaction uncertainty for Achieve [IncidentResolved]

Here are a few obstacle refinements anchored on leaf goals (in textual format for lack of space):

```
NoAmbulanceInterventionWhenMobilized
    ← MobilizationRetracted
        ← MobilizationCancelled
        ← DestinationForgotten
        ← DestinationChanged
    ← ServiceEndsBeforeIntervention
    ← AmbulanceStoppedOrInWrongDirection
        ← AmbulanceInWrongDirection
        ← AmbulanceStopped
```

The satisfaction probabilities for those leaf obstacles in the Brussels area are not publically available although they are partially recorded. We therefore asked 5 experienced paramedics involved in the system to estimate missing or unavailable data. Table IV outlines some of the collected data. The experts provided estimates based on a custom number of interventions; all estimates were then converted into percentages (which explains decimal values in Table IV).

For calibration, statistical data were obtained about the following obstacles: AllocatedAmbulanceNotAtStation (50%), MobilizationCancelled (13%), and MDTturnedOff (33.3%). These leaf obstacles were used as calibration variables.

The results produced by our techniques proved helpful in the following respects.

Managing knowledge uncertainty. Based on the calibration and collected data, the *violation uncertainty* obtained for the top goal Achieve [IncidentResolved] was *100%*, with an uncertainty spread of *0.9189*. Fig. 8 shows the satisfaction uncertainty for this goal. This might seem low; the reason is that the model only captures the ideal case without taking any countermeasure to obstacles into account. The rate of ideal ambulance intervention is actually experienced to be roughly similar to the curve obtained with our technique.

Violation diagrams helped identifying most likely and critical obstacles together with obstacles requiring further elicitation. The black triangles in Fig. 9 show the violation uncertainty and uncertainty spread for the top goal, taking all experts into account. Four obstacles were estimated to cause the top goal not to meet its RDS with more than 90% of certainty, namely, Mobilization Cancelled, MDTTurnedOff, Blackspot, and AvailableBedNotAssigned. Adequate countermeasures to these obstacles should therefore be elaborated and integrated. Among all obstacles, 18 have an uncertainty spread higher than 0.10; they should therefore be further refined or more experts should be asked. The uncertainty spread for the 25 other critical obstacles was low. This indicates that experts roughly

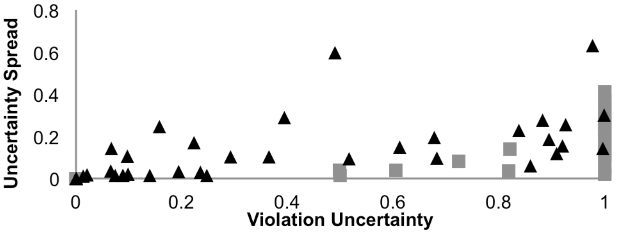

Fig. 9. Violation diagram for Achieve [IncidentResolved]

agreed on their probability of satisfaction. Over the 47 leaf obstacles, 14 are with certainty not causing the probability of satisfaction of the top goal to fall below its RDS.

Capturing uncertainty about risk estimates through multi-point values. Paramedics were asked to estimate a lower bound, the most probable value, and an upper bound for all leaf obstacles. To mitigate the common difficulty of estimating strict and accurate lower and upper bounds [40], the ones we collected were used as 10th and 90th quantiles, with a 10% overshoot being integrated. Eliciting probability distributions would have been impossible in practice as the required statistical background is far too important.

Integrating estimates from multiple experts. The uncertainty spread for the goal Achieve [IncidentResolved], obtained by using the estimates of the first expert only, is *0.9471* which is very high. Had we used the first expert only, 30 leaf obstacles would have been determined to be potentially critical and likely. The uncertainty spread caused by each individual obstacle would have been high on average, as the grey squares in Fig. 9 shows. Using more than one expert helped us reduce the general uncertainty spread and the number of obstacles to be considered as critical and likely.

IX. Related Work

Work on handling uncertainty in RE has generally been devoted to physical uncertainty rather than knowledge uncertainty (as defined in Section I). One exception is Fault Tree Analysis (FTA) [7]. Knowledge uncertainty is supported there through probabilistic distributions on leaf events and Monte-Carlo simulation for propagation to root causes. As seen in Section VI.A, our approach exploits a goal refinement graph to assess the severity of risk consequences. It is integrated in an overall method for risk identification, assessment and resolution. Unlike FTA, the goal/obstacle refinement structure is grounded on a formal framework providing a precise semantics for nodes and enabling the verification of correctness of refinements. FTA does not seem to combine estimates from multiple experts.

Related work on handling physical uncertainty through obstacle analysis was discussed at the beginning of this paper [2, 8, 9, 20, 27, 29]. In particular, the DDP lightweight tool for risk analysis [20] allows goals to be annotated with their degree of importance, obstacles with their likelihood of occurrence, and countermeasures with their effectiveness. DDP does not seem to support uncertainty about estimated quantities and integration of estimates from multiple experts.

In [29], leaf goals are annotated with probability distributions that are analytically combined for obtaining probability distributions for top goals. The approach is extended in [24] for selecting best system design alternatives. The top probability distribution is obtained through a single-value procedure and Monte-Carlo simulation. This technique appears more finer-grained as it supports domain-specific propagation rules and variables. Such rules and variables need however be defined in an ad-hoc manner. No support is provided for multiple expert assessments. Based on a framework for statistical decision making, the technique in [30] evaluates the gain of perfect information and helps identify alternatives maximizing benefits while reducing project risks. This work might be adapted for selecting most critical obstacles.

TABLE IV. EXPERT ESTIMATES FOR LEAF OBSTACLES

Obstacle	Expert	Estimate		
		Min	*Mode*	*Max*
AllocatedAmbulance NotAtStation	Expert1	13.3%	30%	53.3%
	Expert2	15%	40%	50%
	Expert3	10%	15%	25%
	Expert4	30%	40%	50%
	Expert5	2%	4%	10%
ServiceEnd BeforeIntervention	Expert1	10%	12%	15%
	Expert2	0%	5%	15%
	Expert3	4%	8%	12%
	Expert4	10%	15%	20%
	Expert5	10%	10%	20%
PatientCannot ReachAmbulance	Expert1	0%	13.3%	20%
	Expert2	0%	10%	25%
	Expert3	0%	3%	5%
	Expert4	5%	10%	25%
	Expert5	2%	4%	6%

In [38], KAOS goal/obstacle models are extended with a probabilistic layer for technology qualification. The probability of satisfaction for leaf obstacles is estimated by a triangular distribution. The distribution for the root goal is then computed by Monte-Carlo simulation and a single-value propagation algorithm. The probabilistic layer introduced there has no precise semantics in terms of behaviour which limits precise reasoning and questions the correctness of propagations. Here we are not limited to one triangular distribution per obstacle and we support multiple experts. Our propagation procedure appears more efficient as it only requires a single propagation through the obstacle/goal model in Step 1 (see Section VI.A). The notion of required degree of satisfaction (RDS) does not appear relevant to the objectives of [38]; obstacle prioritization is therefore different. TROPOS is another goal-oriented framework, more focussed on soft goals, that supports quantitative reasoning for assessing risks and physical uncertainties [4]. External events are identified and assessed with respect to their positive or negative contributions to goals. The point that different stakeholders may have different perceptions of a risk is also made in [3]. In [25], the focus is on “design-time” uncertainty which enables reasoning about the presence or absence of model elements.

Fuzzy goals may be introduced for coping with satisfaction uncertainty [6, 11, 42]. These are more oriented towards self-adaption at system run-time. The concern there is to reason in terms of proximity of goals being fully satisfied –rather than in terms of probabilities of satisfaction. Our approach might be applied there to elicit membership functions, regulate the fuzziness of goal satisfaction, and involve multiple experts.

CORAS is an UML-based risk analysis methodology supported through various types of diagram [31]. In the assess step, risks are annotated with single-value probabilities to support quantitative reasoning. No support is provided for reasoning about uncertainty over estimated probabilities and for integrating multi-expert estimates.

X. Conclusion

The quantitative technique presented in the paper allows analysts to cope with knowledge uncertainty about satisfaction rates of system goals and their obstructing obstacles. An earlier formal probabilistic framework for goals and obstacles [8, 9] is extended to explicitly reason about uncertain estimates. The impact of estimation uncertainty is measured on top-level goals through two metrics: goal violation uncertainty and uncertainty spread. Satisfaction rates and their uncertainties for top goals are computed by up-propagation through obstacle and goal refinement trees, from leaf obstacles whose satisfaction rate and uncertainty need be estimated. As a result, more critical leaf obstacles may be highlighted for resolution. To reduce their uncertainty margin, our approach allows estimates from multiple sources to be combined.

All techniques in the paper are supported by a tool [10]. The satisfaction probabilities can be specified as distributions or quantiles. All pictures and tables shown in the paper were generated by this tool. The techniques and tool were applied to two non-trivial case studies: a fire detection system and a real ambulance dispatching system deployed in the Brussels area.

Future work should consider a cost/benefit analysis of uncertainty reduction. The calibration and adaptation of satisfaction uncertainties at runtime, à la [18], appears another promising direction. Connections between our approach and others based on fuzzy logics might be worth investigating too.

Beyond frameworks for goal-oriented RE, other RE and risk analysis frameworks might benefit from our techniques. In particular, FTA might integrate similar metrics for problematic uncertainties together with means for reasoning about risk consequences and combinations of multiple expert estimates. Other software engineering areas are faced with the problem of handling uncertainties about estimated quantities [21]. The application of similar techniques appears worth considering in other contexts beyond RE as well.

ACKNOWLEDGMENTS

This work was supported by the RICOSORE project (FRS/FNRS, T.0134.14). Thanks are due to Simon Busard for inspiring discussions and to the reviewers for their comments.

REFERENCES

[1] F. Akhmedjanov, “Reliability databases: State-of-the-art and perspectives”, Denmark. Forskningscenter Risoe. Risoe-R, 2001.

[2] D. Alrajeh, J. Kramer, A. van Lamsweerde, A. Russo and S. Uchitel, “Generating Obstacle Conditions for Requirements Completeness”, *Proc. ICSE'2012: 34th Intl. Conf. Softw. Eng.*, Zürich, May 2012.

[3] Y. Asnar, N. Zanone, “Perceived Risk Assessment”, *Proc. QoP’08: 4th ACM workshop on Quality of Protection*, 2008, 59-64.

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

[5] B. Ayyub, *Risk analysis in engineering and economics*. Chapmann & Hall, 2003.

[6] L. Baresi, L. Pasquale, P. Spoletini. "Fuzzy goals for requirements-driven adaptation." *18th IEEE International Requirements Engineering Conference* (RE), IEEE, 2010, 125-134.

[7] T. Bedford, R. Cooke, *Probabilistic Risk Analysis: Foundations and methods*. Cambridge, 2001.

[8] A. Cailliau and A. van Lamsweerde, “Assessing requirements-related risks through probabilistic goals and obstacles”, *Req. Eng. Journal,* 18(2), 2013, 129-146.

[9] A. Cailliau and A. van Lamsweerde, “Integrating Exception Handling in Goal Models”, *Proc. RE 2014: 22th IEEE Intl. Req. Eng. Conf.,* 2014.

[10] A. Cailliau, KAOS Tools, https://github.com/ancailliau/KAOSTools.

[11] B. Cheng, P. Sawyer, N. Bencomo, J. Whittle, "A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty", *Model Driven Engineering Languages & Systems*, Springer, 2009, 468-483.

[12] R. T. Clemen, R. L. Winkler, “Combining probability distributions from experts in risk analysis”. *Risk Analysis*, 19, 1999, 187-203.

[13] R. M. Cooke, *Experts in Uncertainty*. Oxford, 1991.

[14] R. M. Cooke, L. Goossens, “TU Delft expert judgment data base”, *Reliability Eng. & System Safety 93 (5)*, 2008, 657-674.

[15] R. Darimont and M. Lemoine, *“*Security Requirements for Civil Aviation with UML and Goal Orientation*”*, *Proc. REFSQ’07 – Intl. Conf. Foundations for Software Quality*, LNCS 4542, Springer-Verlag, 2007.

[16] R. Ebendt, G. Fey, R. Drechsler, *Advanced BDD Optimization*, Springer, 2005.

[17] S.J. Ellis, E.R. Henderson, T.H. Klinge, J.I. Lathrop, J.H. Lutz, R. Lutz, D. Mathur, and A.S. Miner, “Automated Requirements Analysis for a Molecular Watchdog Timer”, *Proc. ASE’2014*: *Automated Software Engineering Conference*, Vasteras (Sweden), ACM, September 2014.

[18] I. Epifani, C. Ghezzi, R. Mirandola, G. Tamburrelli. "Model evolution by run-time parameter adaptation." *IEEE 31st International Conference on Software Engineering*, IEEE, 2009, 111-121.

[19] M. Feather, S. Fickas, A. van Lamsweerde, and C. Ponsard, “Reconciling System Requirements and Runtime Behaviour”, *Proc. IWSSD’98: 9th Intl. Workshop on Software Specification and Design*, IEEE, April 1998.

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

[21] N. Fenton, M. Neil. "Software metrics: roadmap." *Proceedings of the Conference on the Future of Software Engineering*, ACM, 2000.

[22] D. Garlan, “Software Engineering in an Uncertain World”, *Proc. FSE/SDP Workshop on Future of SE research*, ACM, 2010, 125-128.

[23] A. O'Hagan et al*, Uncertain judgements: eliciting experts' probabilities*, Wiley, 2006.

[24] W. Heaven, E. Letier, “Simulating and Optimising Design Decisions in Quantitative Goal Models”, *Proc. RE’2011: 19th IEEE Intl. Requirements Engineering Conference*, Trento, Italy, September 2011.

[25] J. Horkoff et al. "Supporting early decision-making in the presence of uncertainty." *Proc. RE’2014: 22nd IEEE Intl. Requirements Engineering Conference*, Karlskrona, Sweden, 2014.

[26] L. M. Krasner et al., *Evaluation of Available Data for Probabilistic Risk Assessments (PRA) of Fire Events at Nuclear Power Plants*, Office of Nuclear Reactor Regulation, 1985.

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

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

[29] 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*, 2004, 53-62.

[30] E. Letier, D. Stefan, E. T. Barr, “Uncertainty, Risk, and Information Value in Software Requirements and Architecture”, *Proc. 36th International Conference on Software Engineering 2014*, 883-894.

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

[32] 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”, *Req. Eng. Journal*, 12(1), 2007, 41-54.

[33] C. Meindel, T. Theobald, *Algorithms and Data Structures in VLSI Design*, Springer-Verlag, 1998.

[34] M. Mendel, T. Sheridan, “Filtering Information from Human Experts”, *IEEE Trans. On Syst., Man, and Cyber.,* Vol 26, No 1, 1989.

[35] Military Handbook, *Reliability Prediction of Electronic Equipment*, MIL-HDBK-217F, 1991.

[36] Naval Surface Warfare Center, *Handbook of Reliability Prediction Procedures for Mechanical Equipment*, 2010.

[37] A. Ramirez, A. Jensen, B. Cheng, “A Taxonomy of Uncertainty for Dynamically Adaptive Systems”, Proc SEAMS’2012, 99-108.

[38] M. Sabetzadeh, et. al, “Combining Goal Models, Expert Elicitation, and Probabilistic Simulation for Qualification of New Technology, *Proc. IEEE 13th Intl. Symp. on High-Assurance Syst. Eng.*, Nov. 2011, 10-12.

[39] Service Public Fédéral, “Vade-mecum de l’aide médicale urgente”, Brussels (Belgium), 2007.

[40] D. Vose, *Risk Analysis: A quantitative guide.* Wiley, 2008.

[41] D. D. Wackerly, W. Mendnhall, Richard L. Scheaffer, *Mathematical Statistics with Applications.* 6th edition, Duxbury, 2002.

[42] J. Whittle, P. Sawyer, N., B. HC Cheng, J.-M. Bruel, "Relax: Incorporating uncertainty into the specification of self-adaptive systems.", *Proc. RE’09: 17th IEEE Intl. Req. Eng. Conf.*, 2009.