Requirements Eng
DOI 10.1007/s00766-013-0168-5

RE 2012

Assessing requirements-related risks through probabilistic goals and obstacles

Antoine Cailliau · Axel van Lamsweerde

Received: 16 November 2012 / Accepted: 20 March 2013

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 existing 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 toward a more complete and robust goal model. A detailed evaluation of our framework on a non-trivial carpooling support system is also reported.

Keywords Obstacle analysis · Risk assessment · Probabilistic goals · Requirements completeness · Goal-oriented requirements engineering · Risk analysis · Quantitative reasoning

1 Introduction

Missing requirements and assumptions are reported as one of the major causes of software failure [22]. Incompleteness often results from unanticipated conditions under which the software should behave adequately. A natural inclination to conceive systems that are too ideal 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 [4, 13, 20, 22, 27]. 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. The likelihood of a risk should not be confused with the likelihood of a consequence of the risk. For example, the

A. Cailliau (✉) · A. van Lamsweerde
Département d'Ingénierie Informatique,
Université catholique de Louvain, Louvain-la-Neuve, Belgium
e-mail: antoine.cailliau@uclouvain.be

A. van Lamsweerde
e-mail: axel.vanlamsweerde@uclouvain.be

Published online: 25 May 2013

likelihood of the risk of the GPS device not working inside an ambulance is not the same as the likelihood of the consequence that the ambulance might get lost when this risk occurs.

Depending on the category of objective being obstructed, risks may correspond to safety hazards [25, 26], security threats [2, 21], inaccuracy conditions on software input/output variables with respect to their environment counterpart [20], and so forth.

Risks must be identified, assessed against likelihood and severity, and controlled through appropriate countermeasures [7, 16, 22, 27].

- At requirements engineering time, risks can be systematically *identified* from prescriptive requirements and descriptive domain properties [20].
- For risk *assessment*, qualitative scales can be used for quick but rough estimation of likelihood and severity [12], possibly in relation with a requirements model [4]—e.g., from '*unlikely*' to '*very likely*' and from '*low*' to '*highly critical,'* respectively. Alternatively, quantitative scales can be used to capture such estimates more precisely [6, 13], possibly in relation to a requirements model [30].
- For risk *control*, we may explore alternative countermeasures and select the 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* [22].

In goal-oriented modeling frameworks, obstacles were introduced as a natural abstraction for risk analysis [3, 19]. An *obstacle* to a goal is a precondition for the non-satisfaction of this goal. Obstacle analysis includes three steps [20]:

(a) *Identification:* as many obstacles as possible to every leaf goal in the goal refinement graph should be identified from relevant domain properties.
(b) *Assessment:* the likelihood and severity of each obstacle should be determined.
(c) *Resolution:* likely and critical obstacles should be resolved by systematic model transformations in order to integrate appropriate countermeasures in the goal model. Alternative resolutions typically encode the aforementioned risk-reduction tactics.

Obstacle analysis has been successfully used in a variety of mission-critical systems, see e.g., [11, 28].

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 [4, 13, 27], 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 [1, 20], countermeasure derivation [20], and goal model analysis, including goal refinement checking, operationalization checking, and behavior model synthesis [22].
- *Measurable statements*: Unlike [4, 13], 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 [29].
- *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 the existing techniques, requirements that prescribe some property to hold in at least X % of the cases should be supported and integrated within the assessment framework.

Partial degrees of goal satisfaction are introduced in [24] for evaluating alternative system options. They are specified through objective functions on random variables associated with the corresponding goals. These variables are refined according to the goal refinement structure provided by the goal model. The degrees of goal satisfaction are determined bottom-up by computing the probability density function of higher-level variables from the probability density functions of lower-level ones. This results in accurate estimations at the price of extra variables to be identified and fairly complex computations on them. Bayesian networks might also be used for making predictions about partially satisfied assertions [14]. Their construction and validation does not, however, take advantage of the available goal structure and turn to be very difficult for complex systems.

The technique presented in this paper 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 can be sorted by degree of criticality. Such ordering can then be used for guiding the

selection among alternative countermeasures identified through risk-reduction tactics [20, 22]. 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 min *in* 95 *% of cases*'?

The paper is organized as follows. Section 2 introduces some necessary background on goal-oriented modeling and obstacle analysis. Section 3 introduces our model-based probabilistic framework for goals, obstacles, and their refinements. Section 4 shows how obstacle probabilities are up-propagated through obstacle refinement trees and how obstacle consequences are up-propagated through the goal model. Section 5 discusses the identification of critical obstacle combinations to be resolved, based on a prioritization of obstacles according to the severity of their consequences. Section 6 reports on our evaluation of the technique on a carpooling support system. Section 7 discusses related work.

2 Background

Some necessary rudiments on goal-oriented requirements engineering are first recalled (Sect. 2.1) before known results on obstacle analysis are briefly summarized (Sect. 2.2).

2.1 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 goal 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 [22].

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: o (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* [22]. The specification pattern for an *Achieve* goal is:

$$\textbf{if } C \textbf{ then sooner-or-later } T, \quad \text{that is, } C \Rightarrow \Diamond T, \qquad (\textit{Achieve})$$

where C denotes a current condition and T a target condition, with obvious particularizations to *Immediate Achieve*, *Bounded Achieve,* and *Unbounded Achieve* goals [22].

The specification pattern for a *Maintain* (resp. *Avoid*) goal is:

$$[\textbf{if } C \textbf{ then}] \textbf{ always } G \text{ (resp. } [\textbf{if } C \textbf{ then}] \textbf{ never } B), \quad \text{that is, } [C \Rightarrow] \Box G (\text{resp. } [C \Rightarrow] \Box \neg B), \qquad (\textit{Maintain/Avoid})$$

where G and B denote a good condition and a bad condition, respectively. (Brackets are used here to delimit optional parts in those patterns.)

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

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

A *goal model* is an AND/OR graph showing how goals contribute positively or negatively to each other [10, 15, 21]. Parent goals are obtained by abstraction, whereas child goals are obtained by refinement. *Refinement paths* connect goal nodes in this graph to their ancestor nodes. Leaf goals are assignable 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. Graphically, goals are represented by parallelograms, domain properties by 'home' shapes and agents by hexagons, see Fig. 1.

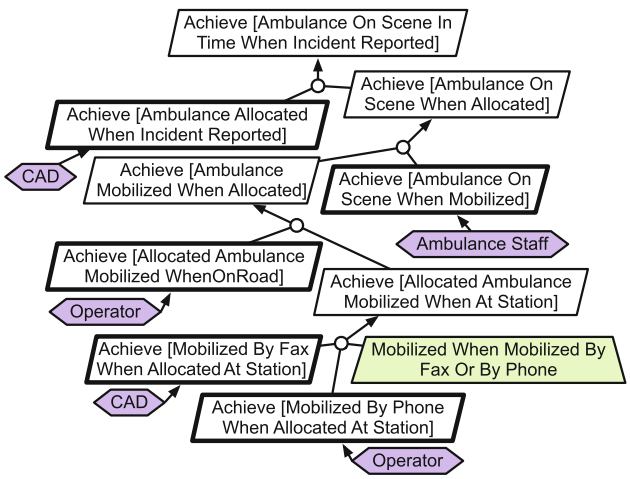


Fig. 1 Partial goal model for an ambulance dispatching system

Refinement patterns are available to help building goal models, e.g., the *Milestone-Driven*, *Case-Driven*, *Guard-Introduction*, *Unmonitorability-Driven*, *Uncontrollability-Driven*, or *Divide-and-Conquer* patterns [10, 21].

In Fig. 1, the two top refinements are milestone-driven (with *AmbulanceAllocated* and *AmbulanceMobilized* as milestone conditions, respectively), the refinement in the middle is case-driven (with *AmbulanceOnRoad* and *AmbulanceAtStation* as case conditions), and the bottom refinement fits no specific pattern. For example, the top goal is more precisely specified as follows:

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

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

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

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

AND-refinements in a goal model should ideally be correct, that is, complete, consistent, and minimal [10, 21].

- 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:

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

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

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

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

$$\{\wedge_{j\neq i}\, SG_j,\ \text{Dom}\} \not\models G \quad \text{for all } i\ (1 \leq i \leq n) \qquad (\textit{minimal refinement})$$

A goal refinement obtained by instantiation of a refinement pattern is formally guaranteed to be complete, consistent, and minimal [10]; the correctness proof is done once for all on the temporal logic formalization of the pattern. The partial goal model in Fig. 1 shows three complete, consistent, and minimal AND-refinements; the bottom refinement is complete and consistent.

In a goal model, goals may be connected by *conflict links* if the goals are potentially conflicting. Goals are *potentially conflicting* (or *divergent*) if there exists a satisfiable and non-trivial boundary condition making them logically inconsistent in the domain [21]:

$$\{B, G_1, \ldots, G_n, \text{Dom}\} \models \textbf{false},\ \{B, \text{Dom}\} \not\models \textbf{false} \quad (\textit{conflict})$$

2.2 Obstacle analysis

An *obstacle* to a goal is a domain-satisfiable precondition for non-satisfaction of this goal [19]:

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

Similar to goals, obstacles can be AND/OR refined into sub-obstacles, resulting in a goal-anchored form of risk tree [21]. In an obstacle 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:

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

OR-refinements should ideally be domain complete and disjoint:

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

Formal and heuristic techniques are available for the identification of obstacles [1, 19] and for the generation of alternative countermeasures [19]. In particular, for the *Achieve* and *Maintain/Avoid* goal specification patterns introduced in Sect. 2.1, specific domain properties are worth eliciting. They take the form:

always if *CSQ* **then** *N*,
that is, $CSQ \Rightarrow N$,

where *N* denotes a necessary condition for the consequent *CSQ* of a leaf goal; *CSQ* is the target condition *T* of an *Achieve* goal $C \Rightarrow \Diamond T$ or the good condition *G* of a *Maintain* goal $C \Rightarrow \Box G$. Such domain properties result in obstacles taking the form:

sooner-or-later *C* **and never** *N*,
that is, $\Diamond(C \wedge \Box\neg N)$

for an *Achieve* goal, or

sooner-or-later(*C* **and sooner-or-later not** *N*),
that is, $\Diamond(C \wedge \Diamond\neg N)$

for a *Maintain* goal.

Consider the goal *Achieve* [*AmbulanceOnScene-WhenMobilized*] in Fig. 1 whose target condition is *AmbulanceOnScene*:

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

Negating this goal yields the root obstacle:

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

The necessary conditions for the target include the following:

$$AmbulanceOnScene \Rightarrow \neg AmbulanceInTrafficJam$$

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

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

3 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 possible obstruction by obstacles. This section defines these various notions more precisely.

3.1 Probabilistic goals

As seen before, a 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. The paper considers finite-behavior systems only. To simplify the presentation, the goal formalizations are propositional; our probabilistic framework is, however, applicable to first-order formalizations as well.

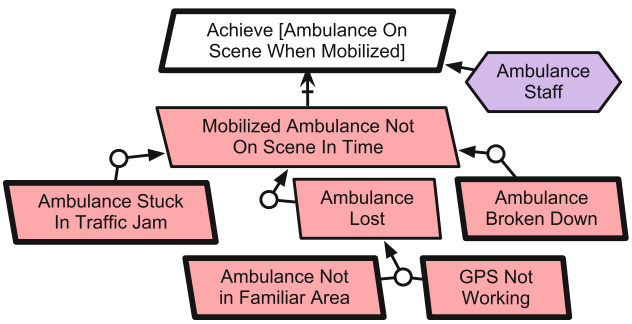


Fig. 2 Partial obstacle model for ambulance dispatching system

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 due to C being *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

- the number of possible behaviors satisfying the goal's antecedent C and consequent ΘT and
- 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* [*AmbulanceMobilized-WhenAllocated*] 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 0.75.

The set of behaviors satisfying the goal *Achieve* [*AmbulanceMobilizedWhenAllocated*] does not necessarily satisfy or deny the goal *Achieve* [*AmbulanceAllocated-WhenIncidentReported*] in Fig. 1; whether an allocated ambulance is mobilized or not does not depend on whether an ambulance is allocated or not. On the other hand, the goals *Achieve* [*AmbulanceMobilizedWhenAllocated*] and *Achieve* [*Allocated AmbulanceMobilizedWhenOnRoad*] are not independent; every behavior satisfying the latter also satisfies the former. Goal dependence is defined more precisely as follows.

Definition 2 Two goals are *dependent* if the set of behaviors that non-vacuously satisfies one of them non-vacuously satisfies or denies the other. Two goals are *independent* if they are not dependent.

In terms of conditional probabilities, the independence of goals G_1 and G_2 is characterized by the following conditions:

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

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.

The AND/OR-refinement structure in a correct goal model can be exploited to syntactically determine whether two goals are independent.

Proposition 1 *In a goal graph whose AND-refinements are complete, two goals are dependent if they are connected through a refinement path or a conflict link.*

Proof (a) In a complete refinement, any behavior satisfying a child goal is a behavior satisfying the parent goal (see the entailment relation for complete refinements in Sect. 2.1). The independence of the parent and child goals would, by definition, require at least one behavior satisfying the child goal to not satisfy the parent goal, which contradicts the completeness assumption. In view of the transitivity of entailments, the argument can be recursively applied to ancestor goals. (b) If a goal G_1 conflicts with a goal G_2, we can find a boundary condition B such that $\{G_1, B, \text{Dom}\} \models \neg\, G_2$ (see the conflict relation in Sect. 2.1). The set of behaviors satisfying G_1 under such circumstances necessarily denies G_2, and the goals are by definition dependent.

Proposition 2 *In a minimal, complete and consistent goal refinement, the subgoals are independent.*

Proof Assume a minimal, complete, and consistent refinement of goal G into two dependent subgoals G_1, G_2, with the set of behaviors satisfying G_1 (say) satisfying or denying G_2. If these behaviors satisfy G_2, we have: $\{G_1, \text{Dom}\} \models G_2$. As the refinement is complete, we have: $\{G_1, G_2, \text{Dom}\} \models G$ which reduces to $\{G_2, \text{Dom}\} \models G$; the refinement is thus not minimal which contradicts our assumption. If those behaviors deny G_2, a similar argument leads us to conclude that the refinement is not consistent as no behavior can be found that satisfies both subgoals. The extension to n subgoals is straightforward.

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

Definition 3 The *estimated probability of satisfaction* (EPS) of 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 is unresponsive, the ambulance is not ready for its next mission, communication failure, etc. Due to such obstacles, there is a probability of this goal not being satisfied.

The EPS of a goal G is denoted by $P(G)$. The EPS of goals G_1 and G_2 in combination is denoted by $P(G_1, G_2)$.

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

For example, ORCON standards require ambulances to be on the incident scene within 14 min in 95 % of cases [22]. This is captured by annotating the goal *Achieve* [*AmbulanceOnSceneInTimeWhenIncident Reported*] with a RDS of 0.95.

Note that the previous situation of (non-probabilistic) goals recalled in Sect. 2.1 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 such as PCTL [16] through an assertion taking the form:

$$C \Rightarrow \Pr_{\geq \text{RDS}}\,[\Theta T]$$

where $\Pr_{\geq \text{RDS}}$ is the probabilistic operator expressing that the probability of paths satisfying ΘT is at least equal to RDS.

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

Given the EPS and RDS of a goal, we can measure the gap between these estimated and prescribed probabilities. If EPS $\geq$ RDS, the goal's required satisfaction threshold is reached; if EPS $<$ RDS, it is not and 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 (SV) of a goal G is defined by:

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

The domain-consistency condition introduced in Sect. 2.1. is generalized accordingly; it now states that there is a chance to observe one behavior at least that satisfies the goal and the domain properties:

$$P(G|\text{Dom}) > 0 \qquad (\textit{domain-consistency})$$

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 Sect. 2.1 are therefore generalized accordingly.

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

$$P(G|SG_1, \ldots, SG_n, \text{Dom}) > 0 \qquad (\textit{complete refinement})$$

Note that this condition is weaker than the completeness condition in Sect. 2.1 as it accounts for partial satisfaction; it covers in particular the case of full satisfaction, equivalent to the completeness condition in Sect. 2.1:

$$P(G|SG_1, \ldots, SG_n, \text{Dom}) = 1$$

The refinement is *consistent* if:

$$P(SG_1, \ldots, SG_n|\text{Dom}) > 0 \qquad (\textit{consistent refinement})$$

The refinement is *minimal* if:

$$P(G|\wedge_{j\neq i} SG_j, \text{Dom}) < P(G|\wedge_j SG_j, \text{Dom})$$
$$\text{for all } i\ (1 \leq i \leq n) \qquad (\textit{minimal refinement})$$

3.2 Probabilistic obstacles

A goal can be partially satisfied because of obstacles to it. Consider the goal *Achieve* [*AmbulanceMobilized-WhenAllocated*] in Fig. 1. Its precise specification is:

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

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

$$AmbulanceMobilized \Rightarrow CrewResponsive$$

By regression [19] of the goal negation through this domain property, we obtain the obstacle *AmbulanceCrew**Not**Responsive*:

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

This condition captures the situation of an ambulance being sooner-or-later allocated without subsequent crew response for 2 min. It is 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 its obstacle condition, that is, the proportion between

- the number of possible behaviors satisfying the obstacle condition and
- the number of possible system behaviors.

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

The obstruction and domain-consistency conditions recalled in Sect. 2.2 must be generalized in this probabilistic setting.

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

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

Note again that this condition is weaker than the obstruction condition in Sect. 2.2 as it accounts for partial obstruction; it covers in particular the case of full obstruction, equivalent to the obstruction condition in Sect. 2.2:

$$P(\neg G|O, \text{Dom}) = 1$$

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

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

In our generalized setting, we need to characterize what obstacle refinements are. The conditions on obstacle refinement introduced in Sect. 2.2 are therefore generalized accordingly.

For an AND-refinement, the *completeness*, *consistency*, and *minimality* conditions are similar to those introduced in Sect. 3.1 for probabilistic goals.

For an OR-refinement, the counterpart of the *entailment* condition in Sect. 2.2 now states that if one of the sub-obstacles occurs then the parent obstacle may occur:

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

This condition is again weaker than the entailment condition in Sect. 2.2; it covers the particular case of full satisfaction, equivalent to the entailment condition in Sect. 2.2:

$$P(O|SO_i) = 1$$

For example, for the top OR-refinement in Fig. 2, we do not have:

$$P(MobilizedAmbulanceNotOnSceneInTime|AmbulanceLost) = 1,$$

as a lost mobilized ambulance might not be on the incident scene within 11 min, but this is not always necessarily the case.

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

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

In our running example, our domain knowledge might allow us to state that '*if a mobilized ambulance is not stuck in traffic jam or lost or broken down, then it will reach the incident scene within* 11 min.' We would then have:

$$P(MobilizedAmbulanceNotOnScene|\neg StuckInTrafficJam, \neg AmbulanceLost, \neg AmbulanceBrokenDown) = 0$$

The *disjointness* condition on sub-obstacles in Sect. 2.2 is generalized into an *independence* condition:

$$P(SO_i|SO_j) = P(SO_i|\neg SO_j) = P(SO_i) \quad \text{for all } SO_i \neq SO_j$$
$$P(SO_j|SO_i) = P(SO_j|\neg \text{SO}_i) = P(SO_j) \quad \text{for all } SO_j \neq SO_i$$

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.

4 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 (Sect. 4.1); the results are propagated from root obstacles to leaf goals in the goal model (Sect. 4.2); the results are in turn up-propagated in the goal model to obtain probabilities of the obstacle consequences in terms of goal obstructions at various levels of abstraction (Sect. 4.3.).

4.1 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

$$\Diamond(AmbulanceMobilized \wedge \Box\neg AmbulanceInFamiliarArea)$$

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

$$\Diamond(AmbulanceMobilized \wedge \Box\neg GPSWorking)$$

we might get from statistical data that the situation of the GPS device 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:

$$P(O) = P(SO_1) \times P(SO_2) \times \cdots \times P(O|SO_1, SO_2, \ldots)$$

Back to our example, we thus also need to know from statistical data how often an ambulance in unfamiliar area with non-working GPS gets 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:

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

The preceding formulas for AND- and OR-refinements are recursively applied bottom-up through the obstacle 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*:

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

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 min is 99, 98, and 100 %, respectively. The propagation rule for OR-refinements yields the following probability for the root obstacle *MobilizedAmbulanceNotOnSceneInTime*:

$$P(MobilizedAmbulanceNotOnSceneInTime) = 1 - (1 - 0.019 \times 0.99) \times (1 - 0.02 \times 0.98) \times (1 - 0.005 \times 1) = 0.0429,$$

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

4.2 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 Sect. 3.2):

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

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:

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

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

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

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

4.3 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. (The presentation hereafter considers refinements consisting of two subgoals only, without any loss of generality.)

As introduced in Sect. 3.1, 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:

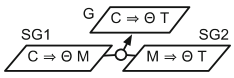
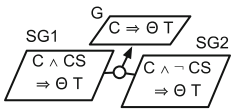
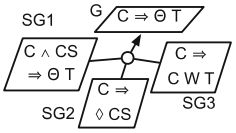
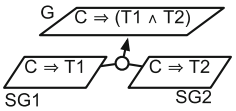
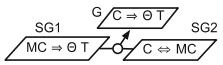
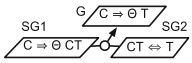
$$\begin{aligned} P(G) = {} & P(SG_1, SG_2) \times P(G|SG_1, SG_2) \\ & + P(SG_1, \neg SG_2) \times P(G|SG_1, \neg SG_2) \\ & + P(SG_2, \neg SG_1) \times P(G|SG_2, \neg SG_1) \\ & + P(\neg SG_1, \neg SG_2) \times P(G|\neg SG_1, \neg SG_2) \end{aligned}$$

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 the refinement meets the non-probabilistic completeness condition in Sect. 2.1, we have that $P(G|SG_1, SG_2) = 1$. The AND-propagation rule then reduces to:

$$\begin{aligned} P(G) = {} & P(SG_1, SG_2) \\ & + P(SG_1, \neg SG_2) \times P(G|SG_1, \neg SG_2) \\ & + P(SG_2, \neg SG_1) \times P(G|SG_2, \neg SG_1) \quad (\textit{and-propag}) \end{aligned}$$

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 (see Proposition 2 in Sect. 3.1).

Table 1 Propagation rules for common goal refinement patterns

Refinement pattern	Propagation equation
 *Milestone-driven*	$P(G) = P(SG_1) \times P(SG_2)$
 *Case-driven*	$P(G) = \mathrm{P}(CS) \times P(SG_1) + (1–P(\mathrm{CS}) \times P(SG_2)$
 *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*	$P(\mathrm{G}) = P(SG_1) \times P(SG_2)$

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

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

The specific simplification of the (*and-propag*) rule thus depends on the goal refinement pattern/tactic used; this information is available in the annotation of the refinement node [21].

To evaluate obstacles consequences, we may 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 the global impact analysis on the model in Fig. 1. 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 Sect. 4.2 is 0.956. Similar computations for the other leaf goals in Fig. 1 yield:

Achieve[*AmbulanceAllocatedWhenIncidentReported*] 0.98,
Achieve[*AllocatedAmbulanceMobilizedWhenOnRoad*] 0.98,
Achieve[*MobilizedByFaxWhenAllocatedAtStation*] 0.90,
Achieve[*MobilizedByPhoneWhenAllocatedAtStation*] 0.95.

The probability for the parent goal *Achieve* [*AllocatedAmbulanceMobilizedWhenAtStation*] is obtained by using the general *and-propag* rule as the refinement does not fit any pattern. In this rule, we have here $P(G|\neg SG_i, SG_j) = 1$; for example, *Achieve* [*AllocatedAmbulanceMobilizedWhenAtStation*] is satisfied given that *Achieve* [*MobilizedByFaxWhenAllocatedAtStation*] 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:

$$P(Achieve\,[AmbulanceOnSceneWhenAllocated]) = 0.984 \times 0.957 = 0.947$$

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

$$P(Achieve[AmbulanceOnSceneInTimeWhenIncidentReported]) = 0.98 \times 0.947 = 0.928$$

The resulting EPS for this goal is 92.8 %; 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.

5 Identifying critical obstacle combinations

Countermeasures should be deployed at requirements engineering time or at system runtime in order to resolve probabilistic goal violations. Such countermeasures can be explored according to risk-reduction tactics [19, 21]. To select the most appropriate ones at modeling time or at runtime, we should identify the most problematic leaf obstacles. These are the ones involved in critical combinations.

An obstacle combination is said to be *critical with respect to a goal G* if it results in $SV(G) > 0$, where $SV(G) = RDS(G) - P(G)$. It is *critical with respect to a set of goals* if it is critical with respect to each of them.

The problem of determining critical combinations can be seen as a multi-criteria optimization problem; we are looking for minimal sets of leaf obstacles that maximize the violation severity of high-priority goals.

A brute-force approach for generating critical combinations consists of generating all possible leaf obstacle combinations. The violation severity $SV(G)$ is then computed for each high-priority goal G considered. If these goals have different priorities, their computed $SV(G)$ can be weighted differently according to their respective priority. The most critical combinations are identified by sorting the leaf obstacle combinations by decreasing, possibly weighted $SV(G)$'s.

Table 2 Violation severity for *Achieve* [*AmbulanceOnSceneInTimeWhenIncidentReported*]

Ambulance lost	Ambulance stuck in traffic	Ambulance broken down	EPS (%)	RDS	SV (%)
1	1	1	92.77		**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	95 %	−0.02
1	0	0	95.10		−0.10
0	0	1	96.44		−1.44
0	0	0	96.92		−1.92

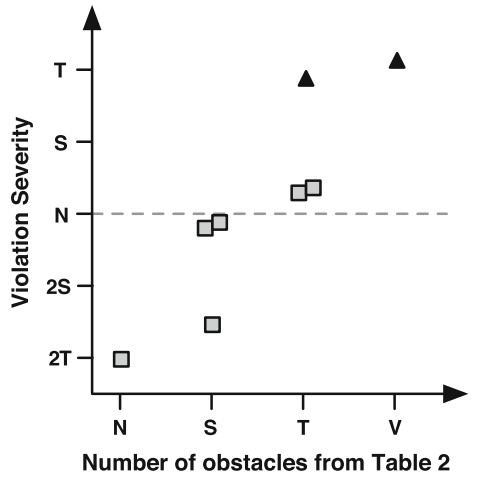


Fig. 3 Ranking of obstacle combinations by violation severity

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 2 shows the computed $SV(G)$ of all these combinations, taking G as the root goal from Fig. 1. The values 1 or 0 indicate that the corresponding obstacle is or is not in the combination, respectively. (The two leaf obstacles at the bottom of Fig. 2 are aggregated in Table 2 into their parent obstacle *AmbulanceLost* for lack of space). As we can see bold in Table 2, there are 4 critical obstacle combinations with respect to the root goal.

Figure 3 visualizes those $SV(rootGoal)$ computed for different combinations. The squares represent leaf obstacle combinations that differ in size along the x-axis. Those above the 0 % threshold, indicated by the dotted line, are critical. The black triangles indicate the most critical combinations for a given size. As we can see in bold in Table 2, two combined leaf obstacles are sufficient for falling under the RDS of the root goal; a single obstacle cannot obstruct this goal enough. From Table 2, we can make the following further observations about critical pairs.

- 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.

$$\frac{\text{Nr. of COOL behaviors satisfying } (RequestEncoded \wedge \Diamond\, AtDestination)}{\text{Nr. of behaviors satisfying } RequestEncoded} \geq 95\,\%.$$

The set of black triangles in Fig. 3 defines a Pareto front. Efficient algorithms for generating Pareto fronts are available [8, 18]. Our generation of leaf obstacle combinations and their ranking by severity can thereby be optimized in order to scale up for larger systems. This optimization is not applied in the next section as it is subjected to future work.

6 Validation

The techniques in this paper were used for risk analysis of a CarpOOLing support system (COOL) [9]. A brief preliminary 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 the COOL carpooling system includes 39 goals and 18 refinements up to 7 levels. A variety of refinements patterns were used [21]. The obstacle model includes 98 obstacles, among which 63 are leaf obstacles.

This section reports on our risk analysis for the COOL system. Section 6.1 summarizes our elaboration of the goal and obstacle models. Section 6.2 discusses our probability estimations for the leaf obstacles in the COOL obstacle model. Section 6.3 presents our global impact analysis of the COOL obstacles, whereas Sect. 6.4 illustrates our local impact analysis on selected obstacles. Section 6.5 provides some concluding remarks from our experience.

6.1 Modeling COOL goals and obstacles

We started with the goal *Achieve* [*RideRequestServed*] corresponding to the passenger needs we wanted to focus on. For the carpooling system to fulfill its mission, we expect 95 % of encoded ride requests to be served, that is,

Hence, the following specification draft (with time constraints being ignored for simplicity):

Goal *Achieve[RideRequestServed]*
Def *For every encoded request for a ride, the passenger shall arrive at her destination.*
RDS 95%
FormalSpec $RequestEncoded \Rightarrow \Diamond\, AtDestination$

Using the milestone-driven pattern, this high-level goal is refined in two subgoals: *Achieve* [*RidePlannedWhenRequestEncoded*] and *Achieve* [*PassengerAtDestinationWhenRidePlanned*]. Each of these is refined toward assignable

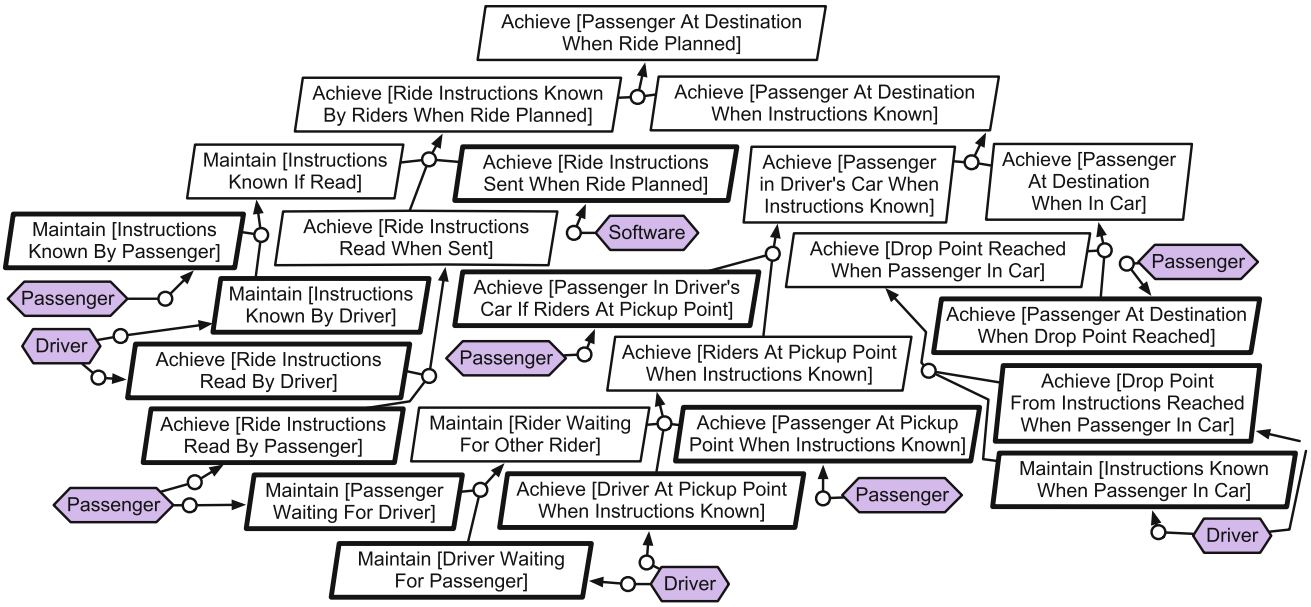


Fig. 4 Goal refinements for *Achieve* [*PassengerAtDestinationWhenRidePlanned*]

requirements or assumptions. Figures 4 and 5 show refinements for these two goals.

When applicable, customized propagation rules were derived from the general *and-propag* rule in Sect. 4.3 to simplify it. The corresponding refinement was then annotated with the simplified rule.

For example, consider the following goal:

Goal *Achieve [OkRidesSelectedFromProposedList]*
Def *A set of convenient possible rides shall be selected from the list proposed by the software*
FormalSpec $RideListProposed \Rightarrow \Diamond\ OkRidesSelectedFromList$

This goal was refined in the three following leaf goals:

Goal *Achieve [OkRidesSelectedByDriver]*
Def *A set of convenient possible rides shall be selected by the driver from the list proposed by the software*
FormalSpec $RideListProposed \Rightarrow \Diamond\ OkRidesSelectedByDriver$

Goal *Achieve [OkRidesSelectedByPassenger]*
Def *A set of convenient possible rides shall be selected by the passenger from the list proposed by the software*
FormalSpec $RideListProposed \Rightarrow \Diamond\ OkRidesSelectedByPassenger$

Goal *Maintain [OkRidesRemainSelected]*
Def *A selected set of convenient possible rides remains selected*
FormalSpec
$OkRidesSelectedByPassenger \Rightarrow \Box\ OkRidesSelectedByPassenger$
$\wedge\ OkRidesSelectedByDriver \Rightarrow \Box\ OkRidesSelectedByDriver$

This refinement did not match available refinement patterns. In this specific case, there is no way the parent goal can be satisfied with one or more subgoals not being satisfied. In addition, the subgoals are independent. The following simplified rule can therefore be applied:

$$P(G) = P(SG_1) \times P(SG_2) \times P(SG_3)$$

Root obstacles were generated by negating corresponding leaf goals. For example, consider the following leaf goal:

Goal *Achieve [RideListProposed]*
Def *For any accurately encoded ride request and ride offer, a list of possible rides shall be proposed by the software*
FormalSpec $RequestEncoded \wedge RideReqAccurate \wedge RideOfferAccurate \Rightarrow \Diamond\ RideListProposed$

This leaf goal is obstructed by the following root obstacle:

Obstacle *NoRideListProposed*
Def *For some accurately encoded ride request and ride offer, no list of possible rides is ever proposed by the software*
FormalSpec
$\Diamond\ (RequestEncoded \wedge RideReqAccurate \wedge RideOfferAccurate \wedge \Box\neg\ RideListProposed)$

All root obstacles were then refined using available techniques [20]. Figure 6 shows a refinement tree thereby obtained.

6.2 Estimating leaf obstacles to the COOL system

The 63 leaf obstacles were then annotated with estimates of their probability of occurrence. As they are grounded on the domain, such estimates can be elicited from user experience, statistical data, or runtime measures from existing software applications. As the system provides specific features for ride evaluation by the riders, the evaluation questionnaire might be designed for non-

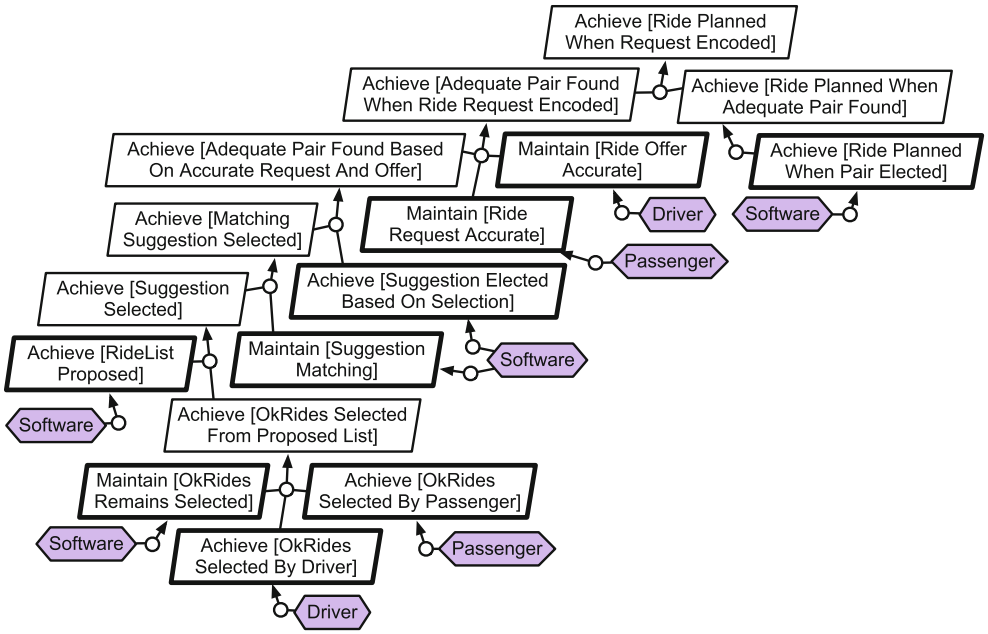


Fig. 5 Goal refinements for *Achieve* [*RidePlannedWhenEncoded*]

positive evaluations to reflect the obstacle model so as to acquire relevant data.

Table 3 provides leaf obstacle estimates based on personal carpooling experience by colleagues and us. These estimates should be refined from statistical data when available.

For example, 15 possible behaviors were estimated to satisfy the following obstacle condition out of 1,000 possible behaviors:

Obstacle *RideOfferCancelledWhenInstructionsSent*
Def *The Instructions are known by the driver and the passenger and the driver cancels her ride offer*
FormalSpec
$\Diamond$ *(InstructionsKnownByPassenger* $\wedge$ *InstructionsKnownByDriver* $\wedge$ *RideOfferCancelled)*

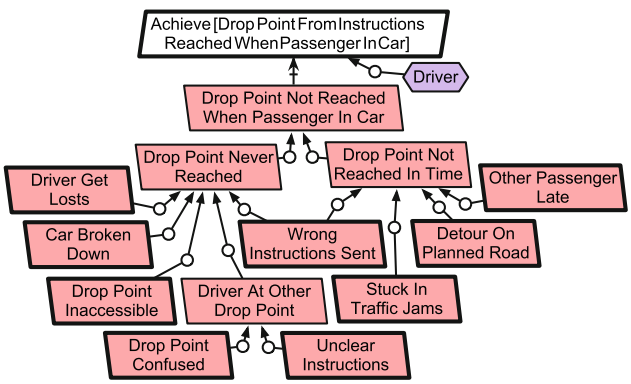


Fig. 6 Obstacle tree against Achieve *[DropPointFromInstructions ReachedWhenPassengerInCar]*

6.3 Global impact analysis for the COOL obstacles

A global analysis was first performed to measure the impact of all obstacles on the system being modeled. The various probabilities were up-propagated from leaf obstacles to root goal using the propagation equations from Sect. 4.

For example, consider the obstacle model fragment depicted in Fig. 6 with probability estimates for the leaf obstacles given in Table 3. We may propagate probabilities according to the rules in Sect. 4.1 to compute the probability of the parent obstacle *DriverAtOtherDropPoint*:

$$P(DriverAtOtherDropPoint) = 1 - (1 - 0.01) \times (1 - 0.005) = 0.015$$

We may then compute the probabilities for the parent obstacles *DropPointNeverReached* and *DropPointNotReachedInTime*:

$$P(DropPointNeverReached) = 1 - (1 - 0.06) \times (1 - 0.005) \times (1 - 0.001) \times (1 - 0.015) \times (1 - 0.001) = 0.08$$

$$P(DropPointNeverReachedInTime) = 1 - (1 - -0.001) \times (1 - 0.1) \times (1 - 0.015) \times (1 - 0.005) = 0.12$$

The latter two probabilities are propagated to the parent obstacle:

$$P(DropPointNotReachedWhenPassengerInCar) = 1 - (1 - 0.08) \times (1 - 0.12) = 0.19$$

Table 3 Estimated probabilities for leaf obstacles

Leaf obstacle	Proba (%)	Leaf obstacle	Proba (%)
Fake ride offer	0.1	Instructions in wrong language	0.5
Ride offer outdated	1	Printing failed	1
Departure date or time confused	2	Ride request cancelled when instructions sent	1.5
Arrival date or time confused	2	Ride offer cancelled when instructions sent	1.5
Departure point confused	3	Instructions not read by passenger	0.5
Arrival point confused	1.5	Instructions not read by driver	0.5
Fake ride request	0.1	Too many luggage	0.5
Ride request outdated	1	Too many riders	0.5
No request matching for that time	2	Not enough seats	0.5
No request matching for that position	1	Riders do not recognize	1
No offer matching for that time	2	Location imprecise	2
No offer matching for that position	1	Ride request cancelled	3
Ride request cancelled and RideList proposed	2	Ride offer cancelled	2
No RideList convenient	1	Passenger late at pickup point	6
Ride offer cancelled and RideList proposed	2	Passenger forgot	1
Ride request cancelled and pair elected	1	Passenger got lost	2
Ride offer cancelled and pair elected	1	Instructions received too late	2
No pickup point found near departure point	0.5	Pickup point not accessible to passenger	0.5
Detour too important for reaching pickup point	1	Driver late at pickup point	2
No pickup point accessible to passenger	0.5	Driver forgot	2
No pickup point accessible to driver	0.5	Driver got lost	6
No drop point found near arrival point	1	Pickup point not accessible to driver	0.5
Detour too important for reaching drop point	0.5	Pickup point confused	1
No drop point accessible to passenger	0.5	Unclear instructions	0.5
No drop point accessible to driver	0.5	Other passenger late	0.5
Pickup time incompatible with road and other point	0.5	Detour on planned road	1.5
Pickup time incompatible with other pickups or drops	0.5	Stuck in traffic jam	10
Journey longer than expected	0.5	Wrong instructions sent	0.1
Drop time incompatible with road and other point	0.5	Car broken down	0.5
Drop time incompatible with other pickups or drops	0.5	Drop point inaccessible	0.1
Wrong contact information	7	Drop point confused	1
Communication failed	3		

This probability may then be propagated to the corresponding obstructed leaf goal using the rule in Sect. 4.2:

$$P(Achieve\,[DropPointFromInstructionsReached\ WhenPassengerInCar]) = 1 - 0.19 = 0.81$$

The result may now be propagated from the leaf goal to the root goal using the propagation rules presented in Sect. 4.3. For the refinement of the goal *Achieve* [*DropPointReachedWhenPassengerInCar*], we use the propagation rule for unmonitorability-driven refinements:

$$P(Achieve\,[DropPointReachedWhenPassengerInCar]) = 1 \times 0.81 = 0.81$$

No obstacle was identified in our model for the goal *Maintain [InstructionsKnownWhenPassengerInCar]*. We can thus propagate one level up using the propagation rule for milestone refinements:

$$P(Achieve\,[PassengerAtDestinationWhenInCar]) = 1 \times 0.81 = 0.81$$

No obstacle was identified against *Achieve* [*PassengerAtDestinationWhenDropPointReached*]. Using the rule for milestone-driven refinements, we continue the up-propagation:

$$P(Achieve\,[PassengerAtDestinationWhenInstructionsKnown]) = 0.65 \times 0.81 = 0.526$$

$$P(Achieve[PassengerAtDestinationWhenRidePlanned]) = 0.808 \times 0.526 = 0.425$$

The probabilities for the goals *Achieve* [*PassengerInDriverCarWhenInstructionsKnown*] and *Achieve* [*RideInstructionsKnownByRidersWhenRidePlanned*] were

Table 4 Critical obstacles

Leaf obstacle	SV for root goal (RDS–EPS) (%)
DriverGotLosts	6.6
StuckInTrafficJams	5.0
RideRequestCancelled	3.7
WrongContactInformation	2.0
PassengerLateAtPickupPoint	1.0
DeparturePointConfused	0.9

similarly obtained by probability propagation from their leaf obstacles. The estimated probability of satisfaction for the root goal *Achieve* [*RideRequestServed*] is then finally obtained:

$$P(Achieve\,[RideRequestServed]) = 0.425 \times 0.66 = 0.28$$

This means that the probability of serving a ride request is only about 28 % if all leaf obstacles were correctly estimated. *Using the ideal model where requirements on countermeasures to likely problems are missing, there is approximately 1 chance out of 3 to serve a ride request.* This is far from the RDS of 95 % prescribed on this main goal.

We thus need to carefully resolve critical obstacles. Due to the large number of leaf obstacles (63), we need to prioritize them, so that we may focus our resolutions on the most critical obstacles.

6.4 Local impact analysis of single COOL obstacles

To complement our global analysis, we may therefore consider one single leaf obstacle at a time, setting the probabilies of all other leaf obstacles to 0. By similar up-propagation of these probabilities, we obtain the violation severity for the root goal. Among the 63 single leaf obstacles, only six revealed to cause severe violation, as Table 4 depicts.

As seen in Table 4, the leaf obstacle *DriverGotLosts*, with estimated probability of 6 %, causes a violation severity of 6.6 % for the root goal; this means that only 88.4 % of requested rides would then be served. The leaf obstacle *StuckInTrafficJam*, with estimated probability of 10 %, causes a violation severity of 5 % for the root goal. The leaf obstacle *WrongContactInformation*, with estimated probability of 7 %, yields a violation severity of 2 % with respect to the root goal's prescribed RDS of 95 %.

Ranking leaf obstacles by their probability of satisfaction is thus not the same as ranking them by the resulting violation severity for the root goal; the obstacle *DriverGotLost* has a lower probability compared with *StuckInTrafficJam* but causes a more severe violation of the root goal.

Table 5 Critical obstacle pairs

Leaf obstacle 1	Leaf obstacle 2	SV for root goal (%)
Arrival date or time confused	Instructions received too late	2.8
Departure date or time confused	Instructions received too late	2.8
Departure date or time confused	Arrival date or time confused	2.8
Communication failed	Instructions received too late	1.8
Arrival date or time confused	Communication failed	1.8
Departure date or time confused	Communication failed	1.8
Arrival date or time confused	Arrival point confused	1.8
Departure date or time confused	Arrival point confused	1.8
Ride offer cancelled when instructions sent	Instructions received too late	1.8
Ride request cancelled when instructions sent	Instructions received too late	1.8
Arrival point confused	Instructions received too late	1.8
Arrival date or time confused	Ride offer cancelled when instructions sent	1.8
Arrival date or time confused	Ride request cancelled when instructions sent	1.8
Departure date or time confused	Ride offer cancelled when instructions sent	1.8
Departure date or time confused	Ride request cancelled when instructions sent	1.8
Communication failed	Ride offer cancelled when instructions sent	0.9
Communication failed	Ride request cancelled when instructions sent	0.9
Arrival point confused	Communication failed	0.9
Instructions received too late	Driver forgot	0.9
Instructions received too late	Driver late at pickup point	0.9

After having found the single leaf obstacles that are critical, we need to turn our attention to *critical pairs* (as defined in Sect. 5). Assuming those critical singletons are selected for resolution, the critical pairs should not redundantly include them. Using the brute-force approach discussed in Sect. 5, we found 470 pairs of leaf obstacles that are critical with respect to the root goal, out of the 1953 generated pairs. Among those 470 pairs, 63 do not include any of the critical singletons in Table 4. The 20

most critical pairs among these are shown in Table 5. Together with the critical singletons, they are the ones having higher priority for resolution. No triple combination excluding the preceding critical singletons and pairs was revealed to be critical.

6.5 Discussion

Global impact analysis revealed that our ideal model is not compliant with our probabilitic requirement of serving 95 % of encoded ride requests. In order to handle the large number of leaf obstacles obtained, some prioritization was required. Local impact analysis revealed critical single obstacles and critical obstacle pairs to be resolved first in the next phase of risk control.

Some of the leaf obstacles appeared to have more importance than others, even if they have a small probability, especially combined with other obstacles. For example, the obstacle *DriverGotLost*, with a probability of 0.06, was seen to potentially obstruct the leaf goals *Achieve* [*DropPointFromInstructionsReached When PassengerInCar*] and *Achieve* [*DriverAtPickupPoint When InstructionsKnown*]. Even if the estimated probability is low, the obstacle might be critical. Leaf obstacles with lower probability may thus be more important than other ones with higher probability when they obstruct more goals; an increase in their probability might have a major impact on root goal satisfaction.

Critical combinations with one or two leaf obstacles appeared to include most of the critical obstacles. Combinations with more obstacles were often supersets of those critical combinations. The resolution of critical singletons and pairs is therefore expected to substantially reduce the number of critical combinations of larger size.

In short, the large number of obstacles made it quite difficult to identify the most critical obstacles to be considered first for resolution. The prioritized list of obstacles produced by our technique helped significantly in that direction.

7 Related work

Probabilistic fault trees are sometimes used for identifying undesirable events in safety–critical systems [6, 24]. Low-level events are annotated with probabilities that are up-propagated from causing events to their consequences in the fault tree. Unlike our refinement structures on probabilistic goal/obstacle assertions, such trees are not grounded on a formal framework; their nodes are just event names and their causal links have no precise semantics. As a consequence, the correctness of event decompositions cannot be established and the propagation rules may appear ad hoc. Moreover, the lack of connection with a goal model may make it hard to identify the root events for starting backward causal analysis.

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 [26]. Such likelihoods and their contributions have no precise semantics, grounded on a formal framework, that would enable formal reasoning.

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 a 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* versus *required* probabilities of satisfaction.

In [29], KAOS goal models are extended with probabilities and propagation rules for technology qualification. The rules there appear different; they do not take advantage of different refinement types. For example, the AND-propagation rule in [29] does not apply to case-driven or non-minimal refinements; the OR-propagation rule for obstacles can be made simpler thanks to obstacle

disjointness. Probabilistic goals as defined in our work are not introduced in [29]; they seem not relevant to the context of that work.

In [23], goals are annotated with objective functions on random variables. The latter are bound by equations tailored to corresponding refinements. Probability density functions are propagated bottom-up to assess alternative goal refinements. This technique enables finer-grained analysis; it, however, requires quality variables to be identified from the model, application-specific propagation equations to be defined on the model, and fairly complex computations to be performed on probability density functions. It is targeted at selecting alternative options rather than prioritizing obstacles by criticality.

8 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 [19], 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 or Markov decision processes as semantic models of probabilistic goal/obstacle specifications is currently being investigated to set our framework on firmer grounds and to improve the accuracy of probability estimations. The required estimates of leaf obstacle probabilities, based on measurable phenomena, is obviously essential and should be better supported.

A dedicated tool is also under development to replace our current spreadsheet calculations and integrate them in semi-formal [21] 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 contain errors; knowledge about certain probabilities might be missing. Such uncertainty should be integrated as well.

Acknowledgments This work was supported by the European Fund for Regional Development and the Walloon Region (TIC-FEDER Grant CE-IQS Project). Bernard Lambeau and Christophe Damas contributed to the elaboration of the goal and obstacle models for the carpooling system. Thanks also to them and to Simon Busard for inspiring discussions on our approach and to the reviewers for comments calling for clarifications.

References

1. Alrajeh D, Kramer J, van Lamsweerde A, Russo A, Uchitel S (2012) Generating obstacle conditions for requirements completeness, Proceedings of ICSE'2012: 34th international conference on software engineering, Zürich
2. Amoroso EJ (1994) Fundamentals of computer security. Prentice Hall, USA
3. Anton A, Potts C (1998) The use of goals to surface requirements for evolving systems. In: proceedings of ICSE'1998: international conference on software engineering, Kyoto, pp 157–166
4. Asnar Y, Giorgini P, Mylopoulos J (2011) Goal-driven risk assessment in requirements engineering. Req Eng J 16(2):101–116
5. Barone D, Jiang L, Amyot D, Mylopoulos J (2011) Reasoning with key performance indicators. In: Proceedings PoEM 2011, LNBIP 92: 82–96
6. Bedford T, Cooke R (2001) Probabilistic risk assessment-foundations and methods. Cambridge University Press, Cambridge
7. Boehm BW (1991) Software risk management: principles and practices. IEEE Softw 8:32–41
8. Börzsönyi S, Kossmann D, Stocker K (2001) The skyline operator. In: Proceedings IEEE 17th international conference on data engineering, Washington, pp 421–430
9. Cailliau A (2012) Risk analysis for a carpooling support system, UCL/INGI Report, September 2012, www.info.ucl.ac.be/~acaillia/publications/carpoolingsystem.html
10. Darimont R, van Lamsweerde A (1996) Formal refinement patterns for goal-driven requirements elaboration. In: Proceedings FSE'4—fourth ACM SIGSOFT symposium on the foundations of software engineering, San Francisco, pp 179–190
11. Darimont R, Lemoine M (2007) Security requirements for civil aviation with UML and goal orientation. In: Proceedings REFSQ'07—international working conference on foundations for software quality, Trondheim (Norway), LNCS 4542, Springer-Verlag, Berlin
12. US Department of Defense (1980) Procedures for performing a failure mode effect and criticality analysis, Standard MIL-STD-1629A
13. Feather MS, Cornford SL (2003) Quantitative risk-based requirements reasoning. Requir Eng J 8(4):248–265
14. Fenton N, Neil M (2001) Making decisions: using Bayesian nets and MCDA. Knowl-Based Syst 14:307–325
15. Giorgini P, Mylopoulos J, Nicchiarelli E, Sebastiani R (2003) Formal reasoning techniques for goal models. J Data Semant 1(1):1–20

16. Jones C (1994) Assessment and control of software risks. Yourdon Press, Upper saddle river, NJ, USA
17. Kwiatkowska M, Norman G, Parker D (2002) Probabilistic symbolic model checking with PRISM: a hybrid approach. In: Proceedings TACAS'02, LNCS 2280, Springer-Verlag, pp 52–66
18. Kung HT, Luccio F, Preparata FP (1975) On finding the maxima of a set of vectors. J ACM 22(4):469–476
19. van Lamsweerde A, Letier E (1998) Integrating obstacles in goal-driven requirements engineering. In: Proceedings ICSE-98: 20th International Conference on Software Engineering, Kyoto
20. van Lamsweerde A, Letier Emmanuel (2000) Handling obstacles in goal-oriented requirements engineering. IEEE Trans Softw Eng 26(10):978–1005
21. van Lamsweerde A (2004) Elaborating security requirements by construction of intentional anti-models. In: Proceedings ICSE'04, 26th international conference on software engineering, ACM-IEEE pp 148–157
22. van Lamsweerde A (2009) Requirements engineering: from system goals to UML models to software specifications. Wiley, NY
23. Report of the inquiry into the London ambulance service. The communications directorate, SW Thames Regional Authority, 1993
24. Letier E, van Lamsweerde A (2004) Reasoning about partial goal satisfaction for requirements and design engineering. In: Proceedings FSE 2004: 12th ACM symposium on foundation of software engineering, Newport Beach, CA, pp 53–62
25. Leveson NG (1995) Safeware: system safety and computers. Addison-Wesley, Wokingham
26. Leveson NG (2002) An approach to designing safe embedded software. In: Proceedings of EMSOFT 2002—embedded software: 2nd international conference, Grenoble, LNCS 2491, Springer-Verlag, pp 15–29
27. Lund MS, Solhaug B, Stølen K (2011) Model-driven risk analysis: the CORAS approach. Springer-Verlag, Berlin
28. Lutz R, Patterson-Hine A, Nelson S, Frost CR, Tal D, Harris R (2007) Using obstacle analysis to identify contingency requirements on an unpiloted aerial vehicle. Requir Eng J 12(1):41–54
29. Robertson S, Robertson J (1999) Mastering the Requirements Process. Addison-Wesley, Wokingham
30. Sabetzadeh M, Falessi D, Briand L, Di Alesio S, McGeorge D, Ahjem V, Borg J (2011) Combining goal models, expert elicitation, and probabilistic simulation for qualification of new technology, IEEE 13th international symposium on high-assurance systems engineering (HASE), pp 10–12