Modeling and Reasoning with Decision-Theoretic Goals

Sotirios Liaskos[1], Shakil M. Khan[2], Mikhail Soutchanski[3], and John Mylopoulos[4]

[1] School of Information Technology, York University, Toronto, Canada
liaskos@yorku.ca,
[2] Department of Computer Science and Engineering, York University, Toronto, Canada
skhan@cse.yorku.ca,
[3] Department of Computer Science, Ryerson University, Toronto, Canada
mes@cs.ryerson.ca,
[4] Department of Information Engineering and Computer Science, University of Trento, Italy
jm@disi.unitn.it

Abstract. Goal models have found important applications in Requirements Engineering as models that relate stakeholder requirements with system or human tasks needed to fulfill them. Often, such task specifications constitute rather idealized plans for requirements fulfillment, where task execution always succeeds. In reality, however, there is always uncertainty as to whether a specification can/will actually be executed as planned. In this paper, we introduce the concept of decision-theoretic goals in order to represent and reason about both uncertainty and preferential utility in goal models. Thus, goal models are extended to express probabilistic effects of actions and also capture the utility of each effect with respect to stakeholder priorities. Further, using a state-of-the-art reasoning tool, analysts can find optimal courses of actions/plans for fulfilling stakeholder goals while investigating the risks of those plans. The technique is applied in a real-world meeting scheduling problem, as well as the London Ambulance Service case study.

Keywords: Information Systems Engineering, Goal Modeling, DT-Golog, Decision Theory

1 Introduction

Goal-Oriented Requirements Engineering is founded on the premise that functional requirements for information systems can be derived from stakeholder goals through a systematic process [1]. For example, the goal *Schedule a Meeting* in a university setting might be fulfilled by a system that supports a set of functions (gather constraints automatically, find free slots, send out reminders, etc.) as well as actions carried out by external actors (participants, a meeting initiator etc.). When the designer selects an alternative for fulfilling top-level stakeholder goals and generates a design, the implicit claim is that the design will fulfill every instance of the goal (e.g., successfully schedule and hold every requested meeting).

Unfortunately, the world is not that simple. The design – which implements a generic plan for fulfilling the goal – may actually fail for a number of reasons, including limited resources, bad scheduling, unexpected obstacles, and more. For instance, participants

may provide inaccurate constraints or maintain incomplete on-line calendars. Or, the email with the meeting invitation may include the wrong time or room. Even sending the email per se often does not guarantee its receipt, especially when mail servers or anti-spam filters are not appropriately maintained. Hence, actions – carried out by the system or actors in its environment – produce effects that vary in uncontrollable ways. In other words, a design may fail to fulfill instances of a goal due to violation of implicit domain assumptions and axioms [1], such as those pertaining to the expected effects of system or user actions.

To address such uncertainties, stakeholders may want to posit probabilistic requirements, such as *Meeting scheduling requests will be fulfilled 95% of the time* [2]. Given such requirements, it is the task of the designer to come up with a plan that will succeed within the probabilistic constraints of the requirement. At the same time, however, stakeholders wish to maintain the multi-objective nature of alternatives analysis. Thus, potentially conflicting goals such as *Quick Scheduling* vs. *Maximize Attendance* or *Keep Secretary Unburdened* vs. *Quality of Schedule* may each be served better by different designs. Stakeholders may be willing to exchange an increased probability of failure with an increased value in one or more of those objectives in case of success. In these circumstances, searching for a suitable design is a process of finding designs that offer the best combination of quality, based on stakeholder preferences, and likelihood of success, i.e. the best expected value.

In this paper, we introduce the concept of *decision-theoretic goals*, which combines the merits of their probabilistic [2] and their preferential [3] cousins in order to capture both probabilistic uncertainty and preferential utility in goal models. To achieve this, we extend the preference and priority-enabled goal modeling language we proposed in [3] to allow representation of probabilistic actions, that is, actions that do not have just one unique effect but a probability distribution over possible effects/outcomes. Utility functions, on the other hand, assign different desirability measures to different such action outcomes. The extended goal model is then translated into *DT-Golog*, a formal specification language that combines ideas from dynamic domain specification languages and Markov Decision Processes (MDPs) [4, 5]. A DT-Golog reasoning tool is then used to evaluate alternative designs by which the specified goals are fulfilled with optimal expected value. This way, both likelihood and value are considered when searching for good solutions to the given requirements problem.

We organize the paper as follows. In Section 2 we present our goal modeling notation and in Section 3 we show how we extend it to allow for decision-theoretic analysis. In Section 4 we show what kinds of automated reasoning the technique enables. Then, in Section 5 we report on an application to a real-world meeting scheduling problem as well as the London Ambulance Service (LAS) case and discuss tool performance. Finally, we survey related work in Section 6 and conclude in Section 7.

2 Goal Models

Goal models ([1, 6]) have been found to be effective in concisely capturing large numbers of alternative sets of low-level tasks, operations, and configurations that can fulfill high-level stakeholder goals. In Figure 1, a (simplified) goal model for scheduling meet-

ings is depicted. The model shows how the high-level goal of a meeting organizer to *Have a Meeting Scheduled* is analyzed into the particular subgoals and actions that are needed for the goal to be attained. The model primarily consists of *goals* (also: *hard-goals*) and *tasks*. Goals – the ovals in the diagram – are generally defined as states of affairs or conditions that one or more actors of interest would like to achieve [6]. Tasks, on the other hand, – the hexagonal shapes – describe particular activities that the actors perform in order to fulfill their goals.

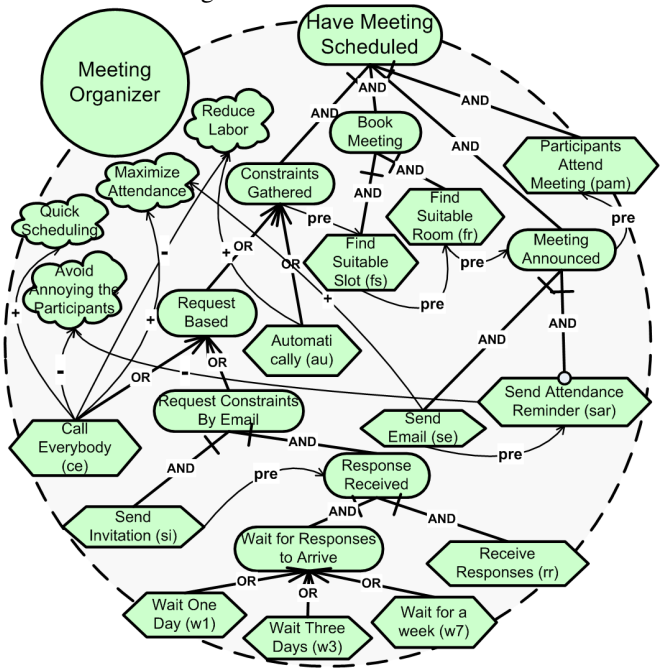

Fig. 1. A goal model

Goals and tasks are connected with each other via AND- and OR-decompositions. By AND-decomposing a goal into other subgoals or tasks, we indicate that the satisfaction of each of its children is necessary for the decomposed goal to be fulfilled. However, children of AND-decompositions can be designated as *optional* through a circular annotation added on their top, such as *Send Attendance Reminder* in the figure. On the other hand, if the goal is OR-decomposed into other goals or tasks, then the satisfaction of one of these goals or tasks suffices for the satisfaction of the parent goal.

The *order* in which goals and tasks are satisfied and performed respectively is relevant. To express constraints over satisfaction ordering we use a *precedence link* ($\xrightarrow{pre}$). A precedence link drawn from a goal/task to another goal/task, indicates that satisfaction/performance of the target of the link cannot begin unless the origin is satisfied or performed. Thus, the precedence link from *Find Suitable Room* to *Meeting Announced* indicates that unless the former is performed, none of the tasks below the latter can be performed. Furthermore, the *negative precedence* link ($\xrightarrow{npr}$) indicates that performance of the link target cannot start if the element at the origin of the link has been satisfied.

Moreover, *soft-goals* (the cloud-shaped elements) represent goals whose fulfillment does not have a clear-cut satisfaction criterion. Since satisfaction of soft-goals cannot be established in a crisp manner, the degree by which they are satisfied is assessed through evidence of satisfaction of other goals. In the goal model, this is represented through positive *helps* ($\xrightarrow{+}$) and negative *hurts* ($\xrightarrow{-}$) *contribution links* drawn from goals and tasks to soft-goals.

The AND/OR decomposition implies a number of sequences of leaf-level tasks that can satisfy the top level hard goal. We call such sequences *plans*. The variability of such plans emerges both due to the existence of OR-decompositions and optional sub-goals in the AND/OR tree, allowing for different subsets of tasks that can fulfill the root goal, and due to the fact that a given subset (i.e., a solution to the AND/OR tree) can be ordered in different ways subject to $\xrightarrow{pre}$ and $\xrightarrow{npr}$ constraints. Furthermore each plan has a different impact to high-level soft-goals. Back in Figure 1, a plan that includes calling everybody to acquire constraints has a negative impact to the soft-goal *Reduce Labour* and should be avoided if that soft-goal is important. It would be a good plan, however, if *Quick Scheduling* were a high priority goal.

3 Goals, Probabilities and Utilities

3.1 Decision-Theoretic Goals

In the standard notation we described above, performance of tasks is assumed to bring about the desired result with certainty. In reality, however, tasks have multiple intended or unintended outcomes, each with different likelihood. As such, task performance does not guarantee goal achievement. To model and reason about this uncertainty, the traditional concept of a goal has been extended to include a probability of success to it [2, 7]. Hence, probabilistic goals describe a desired state of affairs as well as a minimum probability value for this state to be successfully reached.

To this, however, we wish to add here another dimension: that of *utility* as measured by the impact that solutions of the goal have to high-level qualities and stakeholder priorities thereof. Thus, *decision-theoretic goals* require maximization of *expected utility*, which combines probability of success and utility. Thus:

"Have Meeting Scheduled", optimally

requires that a meeting is scheduled while maximizing expected utility. Nevertheless, since expected utility combines probability and utility, it is possible that the plan with the optimal expected utility score has a forbiddingly low success probability. Thus goal:

"Have Meeting Scheduled", optimally, prob 0.7

demands that we wish to schedule the meeting optimally but also ensure that the probability of success of the optimal plan exceeds 0.7. Hence, decision-theoretic goals prescribe both the quality that plans to achieve them must meet, in terms of satisfying high-level preferences, and our risk tolerance with respect to those plans.

To allow reasoning about decision-theoretic goals we extend the standard goal modeling formalism with the following elements: *domain predicates*, which model the state features of the domain, *effect tables*, which model possible effects of tasks and their

Task: Receive Responses			
Affects (decision variables): *excellentResponsesReceived, adequateResponsesReceived, tooFewResponsesReceived*			
Depends on (condition variables): *waited1Day, waited3Days, waited1Week*			
Domain Predicates			p
1.	*waited1Day*	*excellentResponsesReceived*	0.0
2.		*adequateResponsesReceived*	0.33
3.		*tooFewResponsesReceived*	0.67
4.	*waited3Days*	*excellentResponsesReceived*	0.17
5.		*adequateResponsesReceived*	0.5
6.		*tooFewResponsesReceived*	0.33
7.	*waited1Week*	*excellentResponsesReceived*	0.83
8.		*adequateResponsesReceived*	0.17
9.		*tooFewResponsesReceived*	0.0
Attainment Formula: (*adequateResponsesReceived* ∨ *excellentResponsesReceived)*			

(a) An Effect Table

Softgoal: Avoid Annoying the Participants		
Depends on: *calledEverybody, reminderArrived*		
Domain Predicates		*u*
calledEverybody	*reminderArrived*	0.0
	not *reminderArrived*	0.3
not *calledEverybody*	*reminderArrived*	0.7
	not *reminderArrived*	1.0

(b) A Utility Table

Avoid Annoying the Participants	0.1
Quick Scheduling	0.2
Reduce Labor	0.7

(c) A Priority Table

Table 1. Effect, Utility, and Priority Tables.

probabilities, *attainment functions* and *utility tables*, which connect the state of the domain with the achievement of goals and soft-goals, respectively, and *priority tables*, whereby we prioritize goals. We discuss each of these extensions below.

3.2 Representing State and Probabilistic Effects

The first step in our extension is to explicate what is true in the environment before, while and after the tasks of the goal model are performed. To do so we use *domain predicates*. Domain predicates represent fixed facts about the domain – e.g. *available(secretary)* or *has(projector, meetingRoom)* – or facts that may vary due to the performance of tasks or exogenous reasons – e.g. *invitationsSent* or *requested(meetingRoom)*. Each combination of truth values of the domain predicates determines the state in which the process for fulfilling the goals is; let S denote the set of all such combinations. Note that we do not model states explicitly in our proposal; rather we only model state features representing individual changeable properties domain.

Let us now focus on tasks. Ideally, performance of a task by an agent implies that certain facts in the domain change in a deterministic way, leading the system to a new state with certainty. In reality, as we claimed, this cannot always be assumed. Firstly, the outcomes of some tasks rely on chance due to their nature. For example the task *Find Suitable Time Slot* may or may not lead to a situation where *slotFound* holds, depending on the scheduling constraints at hand. Secondly, there are tasks that have one expected and/or desired outcome, but they always run a probability of failure. Thus, the task *Send Invitations* will most probably lead to the fact *invitationsReceived* being true, but there is always a probability of the same fact being false due to a number of factors, such as infrastructure error (server down, anti-spam false positives) or human error (accidental deletion or mishandling of email). Human actors in particular may have their own unidentified and conflicting goals that prevent them from acting as prescribed. Thus, we are interested in representing *probabilistic effects* of tasks – that is effects that, for a variety of reasons, may lead to different outcomes with different likelihood.

To do this we first associate each task with a number of effects that can potentially be brought about upon the task's performance. Effects are represented using do-

main predicates. Thus, performance of the task *Receive Responses* may or may not have an effect that *adequateResponsesReceived*, meaning that important participants responded but not many others, versus the competing (mutually-exclusive in our case) effects *tooFewResponsesReceived*, meaning that too few or none of the important participants responded and *excellentResponsesReceived*, meaning that a very satisfactory amount of responses has arrived, including the important participants. Each of those effects occurs with a certain probability given different conditions.

We can represent these probabilities using a decision table such as that of Table 1a; we call it the *effect table* for the task. The table actually represents the probability distribution over possible effects of the task *Receive Responses*. It contains one or more *decision variables*, which represent possible configurations of effects that the task can bring about, as well as one or more *condition variables*, which are the variables which the probabilities of various value configurations of the decision variables depend on. Both decision and condition variables are drawn from the set of domain predicates. Each combination of condition and effect configurations occurs with a certain probability.

In the example of Table 1a, there are three decision variables (domain predicates: *excellentResponsesReceived*, *adequateResponsesReceived* and *tooFewResponsesReceived*) and the probability that a certain combination of truth values occurs depends on three condition variables that have to do with how long the initiator waited after the original invitation (domain predicates: *waited1Day*, *waited3Days* and *waited1Week*). Thus the probability that adequate responses will arrive within the first three days is 0.5. Note that, in the particular example, both decision and condition variables are mutually exclusive. In the general case, arbitrary combinations of values can be considered.

3.3 Redefining Goal Satisfaction

Hard-goals and Probabilistic Effects. The probabilistic interpretation of task effects necessitates certain refinements to the goal model of Figure 1. Firstly, satisfaction of hard-goals does not exactly reflect the AND/OR structure of the underlying subtree anymore, because it is now measured by the effect of the underlying tasks and not by the mere fact that the tasks are performed. Hence, we define satisfaction of the goal based on the desirable effects of the task.

In the case of multiple effects, as in *Receive Responses* of Table 1a, we construct the *attainment formula* of each task and hard-goal exclusively based on domain predicates, which signifies what effects must be brought about to consider a goal or task satisfied or performed. In our example, the attainment formula of task *Receive Responses* could be *excellentResponsesReceived* $\vee$ *adequateResponsesReceived*. Satisfaction of higher level hard-goals is defined via conjunctions or disjunctions of attainment formulae of tasks depending on the corresponding AND/OR structure. Thus, the attainment formula of *Book Meeting* is *slotFound* $\wedge$ *roomBooked*, each being, in turn, predicates describing probabilistic effects of tasks *Find Suitable Slot* and *Find Suitable Room*, respectively. Note that it is in the discretion of the modeller to redefine satisfaction conditions, by, for example, setting the attainment formula of *Receive Responses* to be just *excellentResponsesReceived* – hence stricter than the previous one.

Assessing Soft-goal Satisfaction. As with hard-goals, in light of probabilistic effects of tasks, a refined model of the satisfaction of soft-goals should also depend on

the actual outcome of task performance, rather than the mere fact that a task was performed. For example, the claim that the task *Send Attendance Reminder* contributes negatively to the soft-goal *Avoid Annoying the Participants* can be supported only if the reminder actually went through – if not, no annoyance can reasonably be assumed. Thus, contribution links are refined into relationships between domain predicates and soft-goals. More specifically, similarly to the attainment formula we saw above, each soft-goal g of the goal graph is assigned an *attainment function* u_g that maps the set S of all possible truth assignments of domain predicates to an interval of real numbers: $u_g : S \mapsto [0, 1]$. Thus, different configurations of truth values for the domain predicates imply a potentially different value for the attainment function of the soft-goal at hand. The higher the attainment value, the more the soft-goal is believed to be satisfied. In the interval [0,1], 1.0 represents full satisfaction of the soft-goal and 0.0 its full denial. Thus, in effect, we quantify the originally qualitative contributions of the goal model – we suggest how below.

We found that representation of attainment functions is also possible using a tabular format such as that of Table 1b – we call it the *utility table*. The variables used in the table represent the domain predicates that influence the satisfaction of the soft-goal. Each combination of truth values of those domain predicates is associated with the actual attainment value (seen as a utility value) of the soft-goal at hand. Thus, attainment values express utility (with respect to the soft-goal at hand) of the situation(s) that is/are described by each truth value combination. In Table 1b, a possible attainment function for the soft-goal *Avoid Annoying the Participants* is shown. Attainment of that goal largely depends on whether the meeting organizer has called all participants on the phone to gather constraints, expressed through domain predicate *calledEverybody* as well as whether s/he has (successfully) sent them reminders to attend the meeting, modeled through the domain predicate *reminderArrived* – other predicates are irrelevant. In the utility table, different combinations of truth values of these domain predicates imply a different attainment value for the soft-goal, shown in the last column. Thus, according to the table, in any state $s \in S$ in which predicate *calledEverybody* is true and predicate *reminderArrived* is false, the attainment value for goal *Avoid Annoying the Participants* (for short: *AvoidAnP*) is $u_{AvoidAnP}(s) = 0.3$.

Aggregating utilities through priority profiles. Utility tables show how each state of the domain implies a different attainment value for a particular soft-goal. To assign to each state a universal "goodness" value which combines all soft-goals of interest we use *priority tables* [3]. A priority table is a representation of the relative importance of soft-goals, in form of a weighted numeric combination. They can include any subset of soft-goals from the goal model. Table 1c shows a priority table with three soft-goals. In the case of an hierarchical organization of soft-goals, we can elicit priority tables for each decomposition of the hierarchy, combine them in a larger profile containing only leaf-level soft-goals and continue our analysis with those (cf. [8]).

Given a priority profile and its weights we can construct a linear combination of attainment formulae of individual soft-goals expressing a measure of global utility U, which we call *total utility*. This measure is used for optimization as we describe below. More formally, let $w_1, w_2, \ldots, w_i$ be the priority values for soft-goals of interest $g_1, g_2, \ldots, g_i$. Then, the total utility U for the goal model in state $s \in S$ is

$U(s) = \sum_i w_i \times u_{g_i}(s)$. Thus, as per Table 1c, 0.1, 0.2 and 0.7 are the priority values for soft-goals *Avoid Annoying the Participants* (for short: *AvoidAnP*), *Quick Scheduling* and *Reduce Labor* respectively. In a state s where soft-goal *Avoid Annoying the Participants* is satisfied by, e.g. 0.3 and *Quick Scheduling* and *Reduce Labor* are satisfied by 0.9 and 0.5 respectively, the total utility value U for that state will be:

$$0.1 \times u_{AvoidAnP}(s) + 0.2 \times u_{QuickScheduling}(s) + 0.7 \times u_{ReduceLabor}(s)$$
$$= 0.1 \times 0.3 + 0.2 \times 0.9 + 0.7 \times 0.5 = \mathbf{0.56}$$

Getting the numbers. The quantitative measures we discuss above occur both in the form of probabilities and in the form of utility/priority values. Overall, while we focus in this paper on the technical representation and reasoning aspects, we believe that there are solid methods and experience in terms of eliciting probabilities and utility measures [9]. As we demonstrate below, probability numbers can come from either simple measurements in the domain or, in the absence of such, subjective judgement by the modellers. Further, there is a variety of ways by which utility and priority numbers can be found, including prominent requirements prioritization techniques such as AHP [10, 8]; both Tables 1b and 1c can be results of AHP's pairwise comparisons. Even subjective ad-hoc assessment is a realistic possibility: it has been found that even if the numbers are not exact, they may be good enough to make correct informed decisions [11]. Otherwise, numerical attainment values are expressions of utility and as such can be obtained through more systematic techniques such as reward elicitation [12].

4 Reasoning about Decision-Theoretic Goals

4.1 Integrating Decision-Theoretic Planning

The above extensions are useful for performing automated reasoning about goal satisfaction under probabilistic effects, utilities and soft-goal priorities. To enable this, the extended goal model is translated into *DT-Golog* [4, 5]. DT-Golog is a formal language for modeling and reasoning about dynamic domains under uncertainty, through combining logical and procedural action theory specification and Markov Decision Processes (MDPs). A DT-Golog specification consists of constructs that represent state features, called *fluents*, as well as agent *actions* that bring the world from one *situation*, where some fluents are true, to another, where the same fluents or different ones might be true. In the action theory specification, DT-Golog programs "glue" actions together in a procedural manner in order to describe ways to achieve goals. Moreover, precondition and successor state axioms define what needs to be true in order for an action to be performed and how exactly fluents are affected by each action, respectively. The effects of actions are multiple and each with a different probability. Further, both actions and fluents are used to define utility functions. This way, DT-Golog can search for *policies* (i.e., nested branching statements prescribing what action to take depending on a condition) within constraints imposed by the program. A returned policy maximizes the *total accumulated expected utility* defined as the (gradually discounted) sum of the products of total utility values and the probability that each such value is obtained when following a remaining portion of the policy. In addition to a policy, DT-Golog also returns a probability of successful termination that sums probabilities of all branches in which a given program runs to completion.

The generation of a DT-Golog specification from the extended goal model in order to allow such reasoning, is based on translating tasks into actions, and domain predicates into fluents. Further, precedence links inform the formation of precondition axioms and effect tables translate into successor state axioms and probabilistic effects. DT-Golog procedures mimic the hard-goal structure, while the soft-goal structure is translated into a utility function. For the interest of space, the formal translation details appear in our longer technical report.

4.2 Querying for Optimal Solutions

Let us return to the example of Figure 1 and discuss different kinds of decision-theoretic goals we can reason about using DT-Golog with the generated specification.

Optimizing expected utility. Decision-theoretic goals of the form *"Schedule Meeting", optimally* are satisfied by a policy p of the translated goal model G, iff p brings about the maximum accumulated expected utility in G. The necessary probability and utility measures are drawn from the appropriately translated effect, utility and priority tables we saw above. Thus, in Figure 1 and assuming we have introduced effect and utility tables for each of the involved tasks and soft-goals accordingly (which we do not present due to space constraints), by setting all soft-goals to be of equal priority we find a policy with total accumulated expected utility 2.7. The policy includes success plans (i.e. branches of the policy in which all tasks are successfully performed) such as `[si,w7,rr,fs,fr,se,sar,pam]` (referring to abbreviations in the parentheses inside the task symbols). DT-Golog informs us also that the total probability of successful termination of the policy is 0.4.

The result is, of course, sensitive to probability and utility values. Thus, if we assume that soft-goals follow the priority values of Table 1c, instead of having equal priority as we assumed above, the resulting policy, with accumulated expected utility 3.1 and probability of success 0.34, includes success plans such as `[au,fs,fr,se,sar,pam]`. Clearly, the increased importance of soft-goal *Reduce Labour* in the priority table favours the choice of automated constraint gathering `au`.

Testing Probability Thresholds. The other kind of decision-theoretic goals that we saw has the form *"Schedule Meeting", optimally, prob* c, where c is a probability value. Such a decision-theoretic goal is satisfied by a policy p of the translated goal model G iff p has the maximum accumulated expected utility in G and p has a probability of success greater or equal to c. Thus, DT-Golog simply tests if the optimal policy has a probability of success above c. For example, the second of the above optimal policy has a success probability of 0.34, meaning that, if we also had a probability threshold c of, say, 0.7, DT-Golog would report failure to find suitable policy. Note that optimality is defined in a global sense and independent of the probability threshold.

5 In Practice

5.1 A Meeting Scheduling Study

As a preliminary test of the feasibility of our modeling technique, we applied it to a meeting scheduling problem that occurs in our workplace. Our *SE@York* seminars are

events that we organize at York University and feature regular talks by visiting or resident software engineering scholars and PhD students. The first author is the meeting initiator of the SE@York meetings and has access to relevant data sources. Potential participants are professors and graduate students of the IT and CS departments. The standard request-based constraint acquisition method is performed by the initiator as seen in the model of Figure 1. In terms of quality goals, the real concern of the organizers is to have good attendance. To a lesser extend they would like to have the meeting scheduled as quickly as possible, for varying reasons including that e.g. a visitor speaker is leaving the country or running out of patience.

Getting the numbers. In our domain, probabilistic data comes from the initiator's email archives (constraint requests and responses) as well as the paper-based room booking logs. The numbers presented in the effect table of task *Receive Responses* in Table 1a are actual values coming out of our data. The email archive data also allow us to calculate the probability that a slot will eventually be found (0.83, in our case). The room booking logs, on the other hand, allow us to calculate the probabilities that the meeting room will be available. For our study, we simply looked at the probability that the room is available at any workday from 9am to 5pm in January. A successful plan for having a meeting properly scheduled is defined to be one in which at least half of the responses have arrived prior to deciding on a time slot (so this is our semantics of the effect *adequateResponses*) and a time slot as well as a meeting room is found immediately. These success conditions are defined accordingly through attainment formulae.

To elicit utilities we make use of the Analytic Hierarchy Process (AHP) [8], focussing on soft-goals *Maximize Attendance* and *Quick Scheduling*. Thus we set $U(s) = 0.75 \times u_{MaximizeAttendance}(s) + 0.25 \times u_{QuickScheduling}(s)$. These two attainment formulae are defined also through pair-wise comparisons on the three probabilistic effects of the task *Receive Responses* as well as on the effects representing the three children of the goal *Wait for Responses to Arrive*.

Reasoning. We focus on the problem of how long the organizer must wait before deciding a slot. For the above utilities, which represent our actual preferences, the optimal solution is to wait for seven days. The probability of success in that case is 0.5. Should *Quick Scheduling* be more important than *Maximize Attendance* – and in our SE@York meetings there have been such cases – after swapping the priority weights, the optimal solution is to try waiting for 1 day. This is due to the fact that waiting less (e.g. *waited1Day*) has much higher utility now. But this solution has lower probability of success, 0.17, since within 1 day adequate responses may have not been received, leading to higher failure probability of the task *Receive Responses*. To see why these probability numbers appear to be low compared to our intuition one must remember that we define ourselves through attainment formulae what constitutes a successful plan.

5.2 Adding Detail

Our use of DT-Golog with the specification that is generated from the semi-formal goal model, exploits only a subset of DT-Golog's expressive power. To further study how DT-Golog's expressive capabilities are applicable to the requirements analysis problem, an application to the well known London Ambulance Service (LAS) [13] case was also performed. The application is described in detail elsewhere [14] – here we focus on key

features. The particular case concerns the problem of managing a fleet of ambulances to respond to emergency incidents in the city of London, UK. What makes the case particularly interesting for our purposes is the explicit performance requirements that can be imposed in the form of an exact probability distribution of allowable ambulance response times. More specifically, concrete performance and reliability requirements can be set for candidate dispatch strategies. Thus, we can demand that a request is responded to within 14 minutes of the time a call is placed, that activation time (call receipt and decision) should always be made in less than 3 minutes, or that travel time to the incident should be 11 minutes 95% of the time and 8 minutes 50% of the time.

To search for designs that meet these performance objectives, extension of the initial DT-Golog specification needs to be performed by adding detail in a number of ways. Firstly, domain information is added in the form of particular instances of objects, agents and contexts that are involved in the LAS operations. Thus, the geography of three city regions is modeled using 10x10 grids. Each hospital, ambulance, incident etc. is represented as a DT-Golog fact and occupies at a given point in time a particular cell in the grid, representing its geographical position. Actions and fluents are relativised to particular objects through parameters. Thus, a fluent of the type *carLocation(c,l,t,s)* is used to represent that an ambulance c is at location l at time t in situation s – the location is represented through a term *loc(x, y)*, where x and y are co-ordinates in the grid. Actions also have a temporal argument, with which their duration is encoded.

To allow analysis of different dispatch strategies, each expected to have different performance characteristics, Golog procedures describing those strategies are written. These are more complex than translations of AND/OR structures that the framework we described above produces. Furthermore, the utility functions are an essential part of each strategy, as they describe the chosen optimization approach. Thus, aspects such as the familiarity of an ambulance driver in an area or the effect of personnel fatigue are modeled through appropriately structured utility tables.

Moreover, *simulation* is necessary when there is a need to model random variables representing exogenous events. In the LAS case, these are the occurrence of emergency incidents. A Poisson distribution of incidents is assumed with various arrival frequency scenarios. Different dispatch strategies are then repeatedly tried for a large number of requests. The response times are counted/averaged and compared with the set requirements, allowing better understanding of the behaviour of different response strategies.

5.3 Tool Performance

DT-Golog has been found to perform reasonably well compared to plain MDP solving. But how does it perform with our goal models? To explore this we tried it with different sizes of goal models, which we constructed by randomly combining smaller models we have developed for real domains (meeting scheduler, automatic teller machine, on-line bookstore and nursing). This way, the resulting artificial models preserved some degree of structural naturalness. Random numbers were entered for the probability values.

We had DT-Golog compute optimal policies for each root goal. The search horizon was set to the maximum plan length the goal model can yield. We used an Intel(R) Core(TM)2 CPU T5500 1.67 GHz with 4.00 GB RAM under Windows 7 to perform the experiments. In Table 2, the time to get the result is given in seconds with respect

Nodes	Bound	Time	Nodes	Bound	Time	Nodes	Bound	Time	Nodes	Bound	Time
10	4	0.0	30	8	0.07	45	19	95.3	60	16	4395
20	8	0.08	40	12	3.7	50	14	75.6	65	21	(*)

Table 2. Time (in sec) to find optimal solution.

to the size of the goal model, (*) signifying non termination within an hour – the bound also indicates the maximum plan length the model can yield. For design time analysis, the tool seems to perform adequately well for sizes up to about fifty nodes. Note also the dependency of the performance on the maximum plan length. We are optimistic that these times will improve as more research is taking place on the matter of reasoning performance (e.g. [15]). It is important to point that the presence of a DT-Golog program restricts the state space to a subset that is meaningful for the domain at hand. This allows DT-Golog to reason much more efficiently than e.g. a plain MDP-based approach would. In the LAS case we described, for instance, the overwhelming space of $30^{300} \cdot 2^{300}$ possible states did not prevent DT-Golog from doing useful analysis.

6 Related Work

Probabilistic analysis of requirements has been a subject for some investigation the past few years. Notable is the work by Letier and van Lamsweerde [2], in which goal structures offer the basis for structuring probability density functions that constitute a measure of achievement of non-functional objectives. Genetic-algorithm based reasoning was further proposed to allow for selecting static solutions that optimize such measures [16]. Recently, these ideas were applied for supporting obstacle analysis [7]. Our framework is different in a number of ways including that it focuses on agent action and dynamic aspects of the solutions (policies/plans) in addition to choices in the goal hierarchy and that it systematically integrates separate measures of priority, utility and probability in a semi-formal manner.

Probabilistic model checking with MDPs has been proposed in PRISM [17] and successfully used in a variety of applications – albeit not yet in the context of goal modeling. One fundamental difference between the model checker and DT-Golog that makes the later more suitable for our particular purpose is the fact that DT-Golog readily allows us to specify complex actions as *programs* and evaluate alterative designs, which is crucial for requirements analysis. Thus, DT-Golog goes beyond the classic MDP approach, where only primitive stochastic actions are allowed and not programs composed from such actions. Other approaches for dealing with uncertainty in requirements engineering have focussed on self-adaptive systems and follow a fuzzy logic based approach [18, 19]. In comparison, we model probability and utility as separate measures, and focus on automated reasoning about optimal behaviours, in terms of both those measures. In addition, a wealth of proposals exist for reasoning about goal models [20]. In that line of work, however, whenever dynamic aspects of the domain are considered, analysis is deterministic and does not take uncertainty of action into account.

7 Concluding Remarks

We presented a decision-theoretic framework for modeling and reasoning about stakeholder goals and priorities in the presence of uncertainty. The framework is based on

the recognition that optimal solutions for fulfilling stakeholder goals will not necessarily be executed as planned, but may fail due to human or system error or other unknown factors. Therefore, to allow for pragmatic design-time analysis, we must take uncertainty into account. This calls for rethinking the semantics of standard goal models that is used for reasoning about alternatives. The main contributions of this paper towards those directions are an approach to probabilistically extend goal models to allow for modeling agent actions with uncertain effects together with stakeholder utilities and priorities, as well as a way to translate them into a formal specification language that allows for evaluating alternative designs based on utility optimization. We also show how detailed analysis can be performed using this toolset. Differences of our proposal from the work already done in the area include a strong focus on dynamic/behavioural aspects of solutions (i.e. sequences of tasks) and allowing exploration of the interplay between priority, utility and probability.

For the future, we wish to work on the core of the DT-Golog reasoner to also allow searching for local optima with respect to probability thresholds, effectively allowing trade-offs between probability and expected utility. Further, empirical assessment of the reliability and accuracy of precise DT-Golog analysis (and the effort investment it takes) is a priority. Scalability is an issue to be investigated in such a context. In terms of scalability of the modeling process, our current sense is that, due to the modularity of the probability and utility specification process (each task and soft-goal has its own table), larger goal models should easily accommodate definition of effects and utilities. In terms of scalability of the automated reasoning, our early results are encouraging for small-to-medium practical models. Nevertheless, we still need to explore solutions with larger models, such as breaking the problem into sub-problems [21].

Acknowledgements. This work has been partially supported by the ERC advanced grant 267856 for a project titled “Lucretius: Foundations for Software Evolution”, April 2011–March 2016 (http://www.lucretius.eu.)

References

1. Anne Dardenne, Axel van Lamsweerde, and Stephen Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1-2):3–50, 1993.
2. Emmanuel Letier and Axel van Lamsweerde. Reasoning about partial goal satisfaction for requirements and design engineering. In *Proceedings of the 12th International Symposium on the Foundation of Software Engineering (FSE-04)*, pages 53–62, Newport Beach, CA, 2004.
3. Sotirios Liaskos, Sheila McIlraith, Shirin Sohrabi, and John Mylopoulos. Representing and reasoning about preferences in requirements engineering. *Requirements Engineering Journal (REJ)*, 16:227–249, 2011.
4. Mikhail Soutchanski. *High-Level Robot Programming in Dynamic and Incompletely Known Environments*. PhD thesis, Department of Computer Science, University of Toronto, 2003.
5. Craig Boutilier, Ray Reiter, Mikhail Soutchanski, and Sebastian Thrun. Decision-theoretic, high-level agent programming in the situation calculus. In *Proceedings of the 17th Conference on Artificial Intelligence (AAAI-00)*, pages 355–362, Austin, TX, July 30–3 2000.
6. Eric S. K. Yu and John Mylopoulos. Understanding “why” in software process modelling, analysis, and design. In *Proceedings of the 16th International Conference on Software Engineering (ICSE’94)*, pages 159–168, Sorrento, Italy, 1994.

7. Antoine Cailliau and Axel van Lamsweerde. A probabilistic framework for goal-oriented risk analysis. In *Proceedings of the 20th IEEE International Requirements Engineering Conference (RE'12)*, pages 201–210, Chicago, IL, 2012.
8. Sotirios Liaskos, Rina Jalman, and Jorge Aranda. On eliciting preference and contribution measures in goal models. In *Proceedings of the 20th International Requirements Engineering Conference (RE'12)*, pages 221–230, Chicago, IL, 2012.
9. Philip J. Boland. *Statistical and Probabilistic Methods in Actuarial Science*. Chapman & Hall – CRC Interdisciplinary Statistics, 2007.
10. Joachim Karlsson and Kevin Ryan. A cost-value approach for prioritizing requirements. *IEEE Software*, 14(5):67–74, 1997.
11. Kevin Regan and Craig Boutilier. Robust policy computation in reward-uncertain MDPs using nondominated policies. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI–10)*, Atlanta GA, 2010.
12. Kevin Regan and Craig Boutilier. Regret-based reward elicitation for Markov Decision Processes. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI'09)*, pages 444–451, Montreal, QC, Canada, 2009.
13. Anthony Finkelstein and John Dowell. A comedy of errors: the london ambulance service case study. In *Proceedings of the 8th International Workshop on Software Specification and Design (IWSSD-8)*, pages 2–4, Schloss Velen, Germany, September 1996.
14. Huy Pham, Mikhail Soutchanski, and John Mylopoulos. A simulator and a Golog implementation of the London Ambulance Service (LAS) Computer-Aided Despatch (CAD) system. Technical report, Department of Computer Science, Ryerson University, http://www.cs.toronto.edu/˜mes/papers/LAS/index.html, 2006.
15. Lutz Böhnstedt, Alexander Ferrein, and Gerhard Lakemeyer. Options in Readylog reloaded – generating decision-theoretic plan libraries in Golog. In *Advances in Artificial Intelligence (KI 2007)*, volume 4667 of *Lect. Notes in Computer Science (LNCS)*, pages 352–366. 2007.
16. William Heaven and Emmanuel Letier. Simulating and optimising design decisions in quantitative goal models. In *Proceedings of the 19th IEEE International Requirements Engineering Conference (RE'11)*, pages 79–88, Trento, Italy, 2011.
17. Andrew Hinton, Marta Kwiatkowska, Gethin Norman, and David Parker. Prism: A tool for automatic verification of probabilistic systems. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 3920 of *Lecture Notes in Computer Science (LNCS)*, pages 441–444. 2006.
18. Jon Whittle, Peter Sawyer, Nelly Bencomo, Betty H. C. Cheng, and Jean-Michel Bruel. Relax: a language to address uncertainty in self-adaptive systems requirement. *Requirements Engineering*, 15(2):177–196, 2010.
19. Luciano Baresi, Liliana Pasquale, and Paola Spoletini. Fuzzy goals for requirements-driven adaptation. In *Proceedings of the 18th IEEE International Requirements Engineering (RE'10)*, pages 125–134, Sydney, Australia, 2010.
20. Jennifer Horkoff and Eric Yu. Analyzing goal models: different approaches and how to choose among them. In *Proceedings of the 2011 ACM Symposium on Applied Computing (SAC'11)*, pages 675–682, Taichung, Taiwan, 2011. ACM.
21. Sotirios Liaskos, Shakil M. Khan, Marin Litoiu, Marina Daoud Jungblut, Vyacheslav Rogozhkin, and John Mylopoulos. Behavioral adaptation of information systems through goal models. *Informations Systems (IS)*, 37(8):767–783, 2012.