Supporting Decision-Making for Self-Adaptive Systems: From Goal Models to Dynamic Decision Networks

Nelly Bencomo and Amel Belaggoun

INRIA Paris - Rocquencourt, France
nelly@acm.org, amel.belaggoun@inria.fr

Abstract. [**Context/ Motivation**] Different modeling techniques have been used to model requirements and decision-making of self-adaptive systems (SASs). Specifically, goal models have been prolific in supporting decision-making depending on partial and total fulfilment of functional (goals) and non-functional requirements (softgoals). Different goal-realization strategies can have different effects on softgoals which are specified with weighted contribution-links. The final decision about what strategy to use is based, among other reasons, on a utility function that takes into account the weighted sum of the different effects on softgoals. [**Questions/Problems**] One of the main challenges about decision-making in self-adaptive systems is to deal with uncertainty during runtime. New techniques are needed to systematically revise the current model when empirical evidence becomes available from the deployment. [**Principal ideas/results**] In this paper we enrich the decision-making supported by goal models by using Dynamic Decision Networks (DDNs). Goal realization strategies and their impact on softgoals have a correspondence with decision alternatives and conditional probabilities and expected utilities in the DDNs respectively. Our novel approach allows the specification of preferences over the softgoals and supports reasoning about partial satisfaction of softgoals using probabilities. We report results of the application of the approach on two different cases. Our early results suggest the decision-making process of SASs can be improved by using DDNs.

Keywords: requirements, specification-methodologies, goal models, dynamic decision networks, bayesian decision theory.

1 Introduction

Goal models have been used to model requirements and decision-making of self-adaptive systems [8, 18, 12, 23]. Goal models support the reasoning about partial and total fulfillment of functional (or goals) and non-functional requirements (or softgoals). Measurement of softgoals fulfillment is difficult due to the vague or fuzzy nature of softgoals satisfaction. Softgoals may not be absolutely fulfilled, yet they can be labelled as sufficiently satisficed [4]. An area of limited study has been the use of probability on goal models [14]. Probability theory can

J. Doerr and A.L. Opdahl (Eds.): REFSQ 2013, LNCS 7830, pp. 221–236, 2013.

also be used to describe the lack of crispness about the satisfiability nature of softgoals. Given a chosen goal realization strategy a probability of satisfaction of a softgoal can be associated with it. The higher this probability the better the satisfaction level associated with the softgoal. Information can be incorporated as new knowledge is acquired.

In this paper we present a mathematical model supported by Dynamic Decision Networks (DDNs) [21] that enriches the decision-making support provided by the goal-based approach and allows reasoning about partial satisfaction of softgoals (expressed with probabilities) and expected utilities.With DDNs, preferences among softgoals are specified using expected utilities with reward functions but expected utilities are also associated with penalty functions. In this paper we explore the usefulness of DDNs to support decision-making for self-adaptation and we also describe a translation method from goal models to DDNs. The resulting DDNs can then be used to trigger adaptation and automatically make the best decision in SASs.

The remainder of this paper is organized as follows: Section 2 presents background on DDNs and previous work using goal models. Section 3 presents how the requirements specification of requirements can be performed using DDNs. Section 4 reports results of experiments. Section 5 described related work. Finally, Section 6 concludes the paper and overviews future research directions.

2 Background

This section briefly overviews DDNs and goal models explaining their relevance for decision-making in SASs.

2.1 Dynamic Decision Networks

Dynamic decision networks (DDNs) extend decision networks, which in turn extend Bayesian networks. Bayesian networks [16] are composed of chance nodes with their associated conditional probabilities and influence arcs that collectively form a directed acyclic graph. Decision networks [10] extend Bayesian networks to provide a mechanism for making rational decisions by combining probability and utility theory. In decision networks, in addition to chance nodes, utility and decision nodes are also included. The decision nodes represent the choices of the decision-maker while utility nodes model the decision-maker's preferences.

DDNs [21] provide a principled approach to make rational decisions in the face of uncertainty within changing environments. To cope with time varying nodes, DDNs maintain a series of time slices to represent nodes at successive moments in time. An arc connecting a node in a previous time slice to a node in a later time slice encodes an influence on the node's value from the previous node value. DDNs provide a useful framework for modeling beliefs about the world, associating preferences with states of the world, and making decisions. Fig. 1 shows a DDN with its components and several time slices.

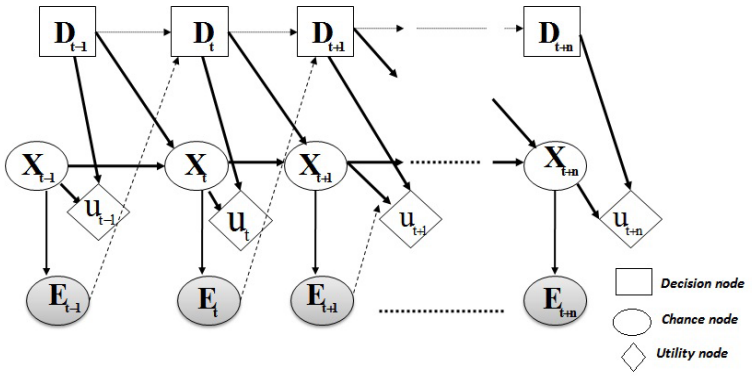

Fig. 1. The General structure of DDN

Why DDNS to Support Decision-making in Self-adaptive Systems. Dynamic decision networks address the problem of decision-making with the following characteristics:

- The environment for making decisions changes over time.
- Information is available to the DDN (as a decision maker) based on data provided by monitorables (i.e. entities in the environment and the system itself that can be monitored) and human-made reports.
- The DDN can be prompted to make a decision at specific times (known or unknown before the DDN is built).
- These decisions are best characterized as choices associated with meeting a goal.
- New decision alternatives can arise at unexpected times. Decision alternatives can also disappear (product of earlier decisions or by known/unknown causes) [5].

Crucially, the above are characteristics exposed by SASs. If we assume that the DDNs can provide support for decision-making in a SAS, the decision process of a DDN-based approach must do the following:

- Define the uncertainty associated with the current situation.
- Balance different conflicting softgoals according to given preferences.
- Maintain the definition of uncertainty over time as new information arrives in a consistent way with the past.
- Incorporate risk preferences (i.e., rewards and penalties) that properly address the current situation modeled.

The above are the basis of the approach presented in this paper and represent the assumptions we have used. The rest of the paper shows our ideas on the use of DDNs for the case of SASs. The incorporation of new decision alternatives and preferences at runtime are not the focus of this paper, but is discussed as future work.

2.2 Goal-Based Models to Support Decision-Making

In [8] and [23] goal-based approaches to reason under uncertainty have been presented. A goal can be satisfied by different goal realization strategies also called tasks. The set of alternative realization strategies that describe different ways how a goal g can be realized is called a variation point associated with goal g (VPg). Different realization strategies have different effects on softgoals. Authors of [23] show how the automatic selection of the best strategy is based on a utility function that sums the possible realization strategies′ impacts on the softgoals and priorities of goals (see equation (1)).

The determination of the best realization (task) is as follows: Let the function *satisfices* represent the contribution value for a task, softgoal:

$$satisfices : T \times SG \rightarrow C$$

where T is a set of tasks, SG a set of softgoals and C is the set of possible contribution values *break, hurt, neutral, help, make*. These are interpreted as corresponding to the range of integer values -2,-1,0,1,2. Moreover i is an index in the set of tasks that represent alternative realizations of goals g, and t_{ig} is thus one of these tasks. The task selected as the realization strategy for goal g is the one with the greatest value of contribution link values for all of the softgoals it influences as presented in the following objective function:

$$max_i \sum_{sg \in SG} w_{sg} satisfices(t_{ig}, sg) \tag{1}$$

Claims [4, 23] has been used to explicitly represent design assumptions made about the contexts that a system may encounter at runtime, and their affect on the realization of system goals. At runtime, such design assumptions can proof to be wrong or not valid anymore, i.e., *Claims* can be seen as markers of uncertainty that can be solved at runtime when more information in obtained. The authors in [23] have shown how *Claims* are useful during execution to maximize the satisficement of a system's softgoals by dynamically choosing between alternative goal realizations after the assumptions have proven to be not valid anymore. The verification of the validity of a *Claim* is done based on monitorables. At runtime and when the monitoring infrastructure notifies that a *Claim* does not hold anymore, system adaptations to an alternative goal realization can be triggered. In terms of the variation points (VPs), it means that a VP will be solved during runtime by the selection of new alternative configurations that will correspond to the realization strategies.

Example: The Vacuum Cleaner. As an example to show the mapping consider the fragment of a simple i* Strategic Rationale(SR) model of a robot vacuum cleaner for a domestic apartment in Fig.2. The vacuum cleaner has a goal to clean apartment (clean apartment) and two softgoals; to avoid causing danger to people within the house (avoid tripping hazard) and to be economical to run (minimize energy costs). The goal clean apartment can be satisfied by two different realization strategies; Clean at night or Clean when empty. These are represented by two alternatives tasks connected to clean apartment by means-end links. The expected effects of the two tasks on the two softgoals are represented by the contribution links between the tasks and the softgoals clean at night task

and the avoid tripping hazard softgoal. Cleaning at night partially denies tripping hazard avoidance but completely satisfies energy cost minimization, while cleaning when empty partially denies energy cost minimization but completely satisfies tripping hazard avoidance. Therefore, the decision of what is the best goal operationalization is not clear as the sum of both tasks' effects on the softgoals is the same, *hurt* + *make*.

A Claim with the value *break* is attached to the contribution link with the value *hurt* that connects the clean at night task and the avoid tripping hazard softgoal. According to the semantics of Claim propagation [23], this has the effect of changing the contribution link value to *neutral*. This in turn has the effect of favouring the task cleaning at night over the task cleaning when empty because the former has a more positive net contribution to satisfaction of the two softgoals; *neutral* + *make* > *hurt* + *make*. During runtime the goal models are kept in memory to support reasoning. Let us suppose that during the execution and when the vacuum cleaner is cleaning the apartment, the monitoring infrastructure may sense a person is at home. In this case, the Claim No tripping hazard is falsified and the run-time reasoning engine (supported by the runtime goal models) is able to evaluate the consequences and order an adaptation from cleaning at night strategy to cleaning when empty. The focus of this paper is to evaluate decisions supported by DDNs instead of the goal-based reasoning capabilities shown above, during both development time and runtime. DDNs are briefly described in the next section.

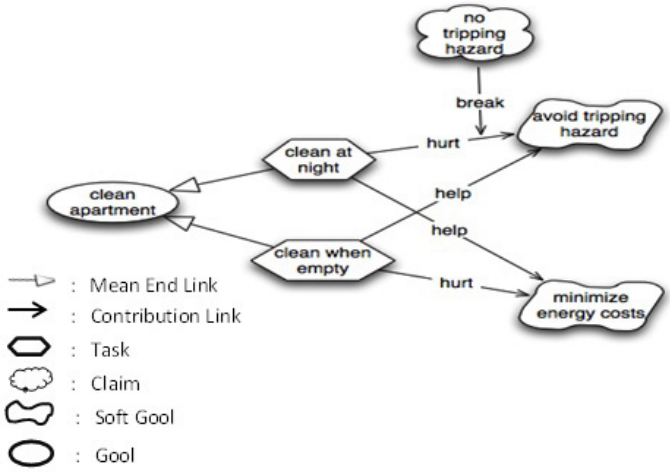

Fig. 2. A robot vacuum cleaner

3 Requirements Specifications of SASs Using Dynamic Decision Networks

In this section we describe and justify the process to map goal models, as presented in Section 2.2, into DDN-based specifications of decision-making for

self-adaptive systems. A set of mapping rules are described and discussed in the context of the vacuum cleaner example.

3.1 Mapping from Goal Models into DDNs

To construct a DDN, we need to specify 5 kinds of information:

- Chance nodes (C_k) also called random variables. Each chance node is associated with a conditional distribution that is indexed by the state of the parent nodes (i.e., the decision node) [20].
- A set of decisions $D_1, ..., D_m$ related to the decision node D_t.
- Utility node and its utility function U.
- The evidence node (E_t)(also called observation node).
- The dependencies between the different nodes described above.

In the rest of this section we explain the mapping process from the goal model ($\mathcal{GM}$) to DDNs.

a) DDNs Correspond to Variation Points and their Subgraphs in Goal Models. We adopt a separate DDN for each goal and its required decision-making. Specifically, a DDN corresponds to the variation point of a given goal g and its subgraph (i.e., realization strategies, softgoals, and claims). In the running example of the vacuum cleaner the *VP* associated with the goal Clean apartment and its subgraph is mapped into a *DDN* (see Fig 3).

b) Decision Nodes and Goal Realization Strategies. Goal-realization strategies in the goal model represent the set of the possible design alternatives. In the context of DDNs, these strategies correspond to the set of possible decisions in the DDN. The following is the corresponding mapping rule:

Mapping Rule 1. Each goal-realization strategy $T_k \in \{T_1, ..., T_l\}$ in $\mathcal{GM}$ corresponds to a $D_k \in \{D_1, ..., D_m\}$, where D_k represents a value of the decision in the DDN.

c) Chance Nodes and Softgoals. The softgoals represent the non-functional requirements to be satisficed [4]. Different design decisions may have positive or negative effects, and in different proportions, towards meeting a softgoal. Different from goals, softgoals can hardly ever be labelled 100% satisfied or %100 unsatisfied in an unambiguous sense. Satisficement of a softgoal needs a decision-making strategy that attempts to meet an acceptability threshold rather than an absolute value [4]. In the case of goal models, like the one in Fig. 2, whether the softgoals are considered satisficed or not depends on the realization strategies and their effects on each softgoal (represented by the contribution links).

In the context of DDNs, each softgoal SG_j in the goal model is viewed as a chance node C_k whose values are dictated by some probability distribution.

Definition 1. *The probability distribution represents the probability of being satisficed given a decision (i.e., realization strategy).*

Therefore, each contribution link that departs from a realization strategy to a SG_j in the goal model is translated into a conditional probability distribution (CPD) associated with each softgoal SG_j. Given that the realization strategies are mutually exclusive, the Bayes theorem can be applied to calculate the probability of satisficement for each SG_j. Table 1 and Table 2 show examples of the conditional probabilities tables for the example of the vacuum cleaner. Given the above the following is the corresponding mapping rule:

Mapping Rule 2. Each softgoal $SG_j \in \{SG_1, ..., SG_n\}$ in $\mathcal{GM}$ corresponds to a chance node $C_k \in \{C_1, ..., C_n\}$ in the DDN.

Each contribution link $l_k(T_i, SG_j)$ that describes the effect of a T_i on a SG_j corresponds to a conditional probability $P(SG_j|T_i)$. A simple way to propose the values of these conditional probabilities is to make a direct map from the five point range of values {*break, hurt, neutral, help, make*} to the probability values {*0.0, 0.25, 0.5, 0.75, 1.0*}. However, if more information is available a more sophisticated mapping can be performed.

In the case of the example of the vacuum cleaner, two realization strategies T_1, T_2 exist that affect the softgoal SG_1 (i.e., Avoid tripping hazard)and SG_2 (i.e., Minimize energy costs). The conditional probability tables associated to SG_1 and SG_2 are shown in Tables 1 and 2.

Table 1. CPT of the node Avoid Tripping Hazard

Avoid tripping hazard node (SG_1)		
T_i	P(SG_1=F)	P(SG_1=T)
Clean when empty	0.45	0.55
Clean at night	0.11	0.89

Table 2. CPT of the node Minimize energy Costs

Minimize energy Costs node (SG_2)		
T_i	P(SG_2 = F)	P(SG_2 = T)
Clean when empty	0.25	0.75
Clean at night	0.1	0.9

d) Preferences in the Utility Node and Softgoal Priorities. In decision theory, a utility function is a scalar that assigns a cardinal scale to each outcome and decision indicating its desirability [9].

Softgoals can have an associated priority, that indicates how important it is to satisfice that particular softgoal. The specification of the weights in the utility function (utility node in the DDN) can be based on the softgoals priorities.

Table 3 defines the utility table with all the possible combinations of effects on the softgoals (using the values *true* T and *false* F) given a cleaning strategy.

The weights are ranged from 0 until 200 in this case. The following is the corresponding mapping rule:

Table 3. Utility table (preferences)

	Utility node		
Cleaning Strategy	Avoid tripping hazard	Minimize energy costs	**Weight**
1 Clean When empty	F	F	0
2 Clean When empty	F	T	15
3 Clean at night	F	F	0
4 Clean at night	F	T	30
5 Clean When empty	T	F	200
6 Clean When empty	T	T	90
7 Clean at night	T	F	150
8 Clean at night	T	T	90

Mapping Rule 3. For each goal realization T_i (i.e., decisions in DDNs) and each softgoal SG_j (i.e., the chance nodes in DDNs) we assign a weight w_{ji} that expresses the preferences which is set as a function $U(SG_j|\,T_i)$.

$$w_{ji}\colon \mathbf{T} \times \mathbf{SG} \rightarrow U(SG_j|\,T_i)$$

where SG is the set of softgoals, T is the set of goal realizations and w_{ji} represents the set of the priorities over the goal realizations. The domain expert sets the weights of the utility table. These weights are known as rewards or penalties. Table 3 shows an example of a possible set of weights that describes the domain expert preferences. The weight 0 in the 1st and 3rd rows means that the domain expert penalizes those combinations as they have negative effects on both softgoals (note the value *false* F related to both softgoals). Similarly, the 2nd and 4th row also have low weights (respectively 15 and 30) what means a low level of preference. The 5th and 7th rows, on the contrary, show high weights, 150 and 200 respectively. These highest weights mean that these combinations are considered by the expert as the most suitable. The domain expert has a preference on the strategy "Clean when empty" over the strategy "Clean at night". Furthermore, both combinations represent positive effects on the softgoal "Avoid tripping hazard" and negative effects on "Minimize energy costs" what means that for the expert it is more important to favor 'Avoid tripping hazard" than "Minimize energy costs" (see that 6th and 8th rows have lower weights, specifically 90).

e) Evidence Node and Claim Monitoring *Claim* monitoring offers the appropriate mechanism to support the observation model and provide the observations required by the DDNs.

An example of an Observation or Evidence can be the fact that a Claim has been falsified, e.g. the *Claim* No tripping hazard goes from True to False). This falsification could trigger the need to make a decision about what adaptation to realize, if any.

The observation model should include the possibility of failure, i.e., the possibility that the observation may not be 100% accurate due to problems and failures associated to monitorables. In terms of the observation of the falsification of a *Claim*, this refers to the fact that such a falsification may not be true.

In the running example, if we have consider an ideal world where failures do not exist (i.e., the monitorables are 100% reliable) when Evidence is observed the probability is believed to be P(E) = 1. Otherwise, if the monitorables are not 100% reliable, P(E) is less than 1. P(Obs | (no shock detected AND light level constant)) < 1. A graphic showing the mapping is depicted in Fig 3.

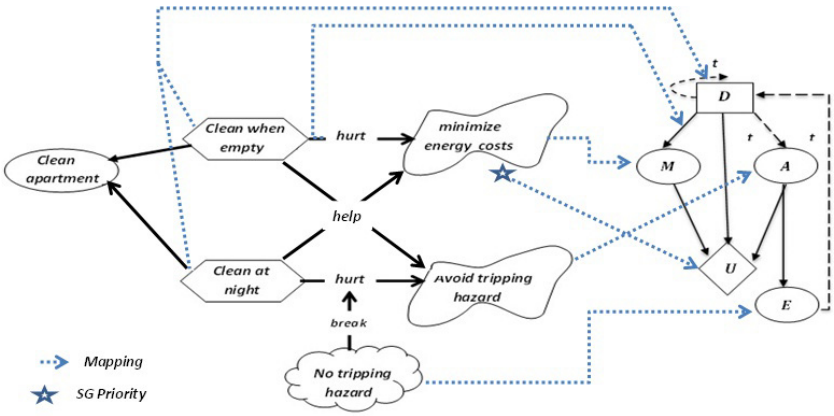

Fig. 3. The robot vacuum cleaner system's i* goal model mapped into a condensed form of DDN

3.2 Evaluating the DDN

A DDN is evaluated in order to make a decision based on the realization strategies with the highest utilities. The DDN is evaluated using the formula (2) for every realization realization strategy T_i to compute the probability-weighted average utility for that realization strategy, also known as the expected utility. The set of preferences over every softgoal is represented by $U(SG_j|T_i)$ and the conditional probability of each softgoal given the available evidence E is represented by $P(SG_j|E, T_i)$. The realization strategy with the highest expected utility is chosen.

$$EU(T_i|E) = \sum_j P(SG_j|E, T_i) \times U(SG_j|T_i) \tag{2}$$

Next, we present an application of DDNs to decision-making in SASs.

4 Experiments

This section describes experimental results for demonstrating the value of our approach using DDNs to support decision-making for self-adaptation. Section 4.1 describes the Remote Data Mirroring (RDM) example. Section 4.2 shows the application of our approach based on DDNs for the case of the RDM application.

4.1 Remote Data Mirroring

RDM [19] is a classic technique for tolerating failures by keeping copies of important data at physically isolated locations to protect data against inaccessibility, to reliability and provide resistance to data loss. An RDM system can be configured in terms of the topology of the network (e.g. using a minimum spanning tree algorithm) and also in terms of how data is distributed among data servers. There are two modes to configure data distribution: synchronous and asynchronous. The synchronous mode is the only mode in which non-catastrophic multiple failures will ever provoke the lose of data. In contrast, in the asynchronous remote mirroring mode, data that hasn't propagated to other sites can be lost at certain risk. Each configuration provides different levels of data protection, performance and costs. For example, the synchronous mode provides better data protection than the asynchronous mode, but it also incurs a network performance penalty as every change must be distributed across the network. The asynchronous mode provides better network performance, however it also provides weaker data protection.

Fig. 4 shows the i* SR goal models for the RDM application. The RDM application must achieve functional goals such as constructing a connected network and distributing data. These functional goals can be achieved through alternative goal realization strategies that includes constructing different network topologies, such as a Minimum Spanning Tree or Redundant Topology and Changing Propagation Parameters. The application has three softgoals Minimize Operational Expense, Maximize Data Reliability and Maximize Network Performance. A Claim Redundancy Prevent Network Partition has been attached to the goal model [19] to make explicit the uncertainty about the usefulness of the choice of Redundant Topology at any point at runtime. The Claim can become disproven at runtime for several reasons, for example it can be provoked due to simultaneous failures of two or more links. Monitorables and sensor allow the the system to check the validity of the Claim at any point during runtime.

4.2 Experiments

Netica [1], a tool for Decision and Bayesian networks, has been used for the experiments. A DDN to support the decision-making of the best topology to use has been associated with the the variation point Select Topology (see Fig. 4). The DDN constructed (with three unrolled time slices) is shown in Fig. 5 . D_{jt} represents the decision node with two possible decisions $D_1 and D_2$ which correspond with the realization strategies T_1:Use MST Topology and T_2:Use Redundant Topology respectively. The DDN also presents a set of three chance nodes MR_t, MP and MO used to model the softgoals Maximize Reliability, Maximize Performance and Minimize Operational Costs respectively. Notice that MR_t comes from a softgoal with a contribution link with a Claim attached. MR_t is therefore an observable chance node. MP and MO are modeled as static chance nodes. The evidence node E_t represents the event that the Claim Redundancy Prevents Network Partitions is changes is value True or False at time t.

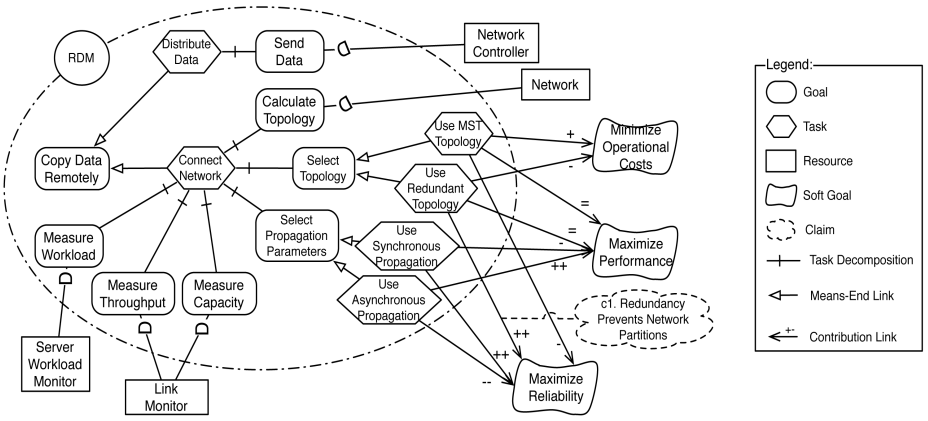

Fig. 4. i* SR goal model for the remote data mirroring application (RDM) (from [19])

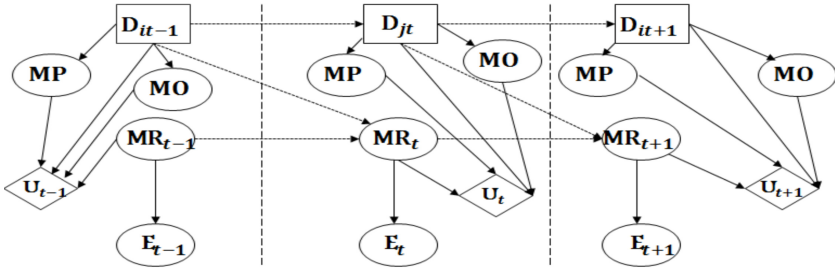

Fig. 5. DDN for the case of the RDM application (with three time-slices)

In order to evaluate the DDN we have considered the following probabilities: $P(MR_t = true|\text{Mst Topology}) = 0.3$, $P(MR_t = true|\text{Redundant Topology}) = 0.9$, $P(\text{MP} = true\ |\text{Mst Topology}) = 0.5$, $P(\text{MP} = true|\text{Redundant topology}) = 0.5$, $P(\text{MO} = true\ |\text{Mst Topology}) = 0.75$, $P(\text{MO} = true|\text{Redundant topology}) = 0.3$.

The weights associated with the possible combination of nodes are given in Table 4. Take note that these weights express the preferences that represent the relative importance of each combination of effects of the topology used on the softgoals. For example, the 3rd row in Table. 4 has the highest weight value (200) in the utility table what means that the 3rd row is the most favored combination as it encodes that the cost of using Redundant Topology has a positive effect on the three softgoals Maximize Reliability, Minimize Operational Costs and Maximize Performance(see the values T for the three softgoals). Note also the 2nd row has a weight value that is much lower (i.e., 10) what means that even if we have the same Redundant topology this time, it is not suitable as it has negative effects on Maximize Reliability and Minimize Operational costs.

Fig. 6 shows the computation of the expected utility (EU) of the possible topologies from time $t = 0$ to $t = 7$, based on the probability of Maximize Reliability node, Maximize Performance node, Minimize Operational Costs and

Table 4. Utility table

Utility node				
Topology	MR	MP	MO	**Weight**
1 Use MST Topology	F	F	F	0
2 Use Redundant Topology	F	T	F	10
3 Use Redundant Topology	T	T	T	200
...	..	..	..	...
15 Use Redundant Topology	T	F	F	10
16 Use MST Topology	T	T	F	80

utility node predicted by each decision taken at the next time slice (as shown in Fig. 5 topology decisions have influence on the Maximize Reliability node on the next time slice).

At development time (i.e. at time slice 0), the designer selects Redundant Network Topology as the best decision to use with synchronous propagation as the initial configuration. This configuration is based on the validity of the assumption Redundancy Prevents Network Partitions that states that a redundant network topology prevents network link failures from partitioning the network (the reason why the *Claim c1* has been made to explicitly record that assumption). In terms of a DDN this mean that with no evidence about the fact that Redundancy Prevent Networks Partitions entered yet in the DDN the most likely decision is to use Redundant Topology as the expected utility EU(Redundant Topology)>the expected utility EU(MST Topology).

At some points during runtime (i.e., from time slice 1 to 7 in Fig. 6), however, new information is collected that concludes the *Claim c1* Redundancy Prevent Networks Partitions is false meaning that according to current environmental conditions, the Use of redundant Topology decision does not necessarily prevent network partitions anymore. EU(MST Topology) >EU(Redundant Topology) and therefore, the decision to use MST Topology is considered by the DDN as the best one. The DDN triggers an adaptation accordingly. Fig. 7 shows the case of the monitored falsification of the *Claim c1* at slice time 3 and the monitored value True of *Claim c1* at slice time 6; the DDN has correctly suggested the adaptations from the original design decision Redundant Topology to MST Topology after slice time 3 to go back to Redundant Topology after slice time 6.

The results appear to be consistent with the evidence (observations) that provoked the adaptations and furthermore, they agree with those presented in [19]. Our approach using DDNs has also been successfully applied on the case study of the sensor network GridStix [11] with results also compatible with those shown in [23]. The results of the evaluation of the DDN to the robot vacuum cleaner, RDM system, and GridStix, reported in [2], while somewhat preliminary, are positive as the DDNs allowed both (1) the analyst to make design decisions during development time and (2) the applications to make decisions to adapt to new situations at runtime.

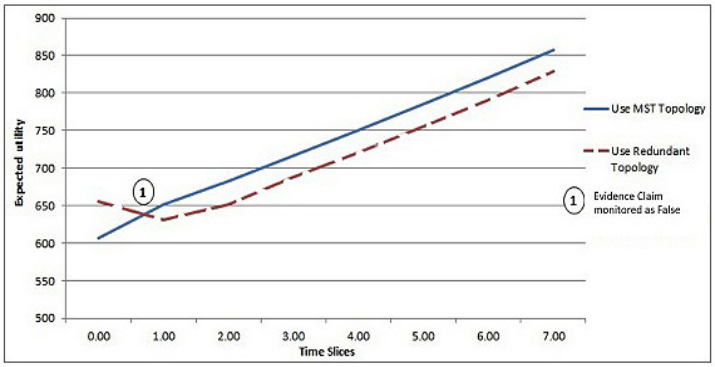

Fig. 6. Expected utilities during seven time slices

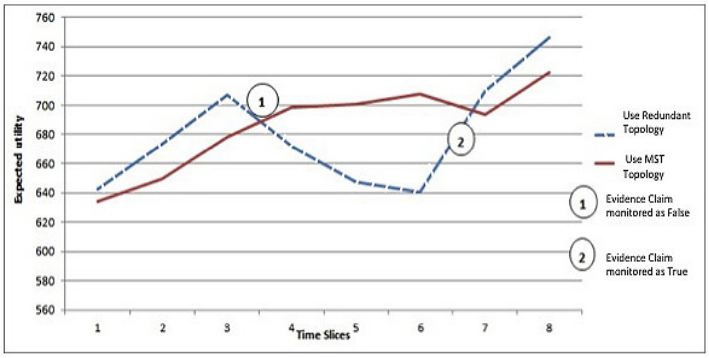

Fig. 7. Expected Utilities during eight time slices

5 Related Work

The related work described in this section is divided in two categories, work on uncertainty tackled using goal-based models for the case of SASs and decision-making using Bayesian theory.

Researchers have tackled uncertainty in SASs in different ways. As discussed by [3, 13], there is a dearth of applicable techniques for handling uncertainty in this setting. Welsh et al [23] introduced REAssuRE to use goal models and *Claims* for driving self-adaptation. In contrast to REAssuRE, in our case and when using DDNs, preferences among softgoals are specified using both expected utilities with reward functions and also penalty functions. Uncertainty in adaptive systems has also been tackled by RELAX [22], a requirements language that explicitly addresses uncertainty inherent in adaptive systems. While RELAX uses fuzzy logic to specify more flexible requirements within a goal model to handle the uncertainty, we use probabilities. Emmanuel et al. [14] specify partial degrees of goal satisfaction and quantify the impact of different system alternatives on high level goals that can be used to guide requirements elaboration and design decision-making. The degree of satisfaction of such goals is

modeled by objective functions on quality variables. The non-functional goals are specified formally using a probabilistic model and interpreted in terms of application specific measures. Their approach is different from ours. They tackle decision about alternative system designs during requirements and design engineering. In our case we are concerned about decision-making between alternative decisions to meet a functional goal due to environmental changes what crucially includes also decision-making at runtime. The work of Giorgini et al. [7] deals with formal reasoning about goal models. They use a probabilistic model and label propagation to calculate the evidence for satisfiability and deniability of goals. Their approach deals with conflicts between softgoals, however, different from our work, they do not resolve conflicts but just tackle their identification. Instead of probability they use evidence which can be seen as less precise than calculating the exact probability. In this paper, we have not taken into account uncertainties about quality of observations, i.e. we have assumed no errors or noise introduced by the monitoring infrastructure and therefore we trust 100% the monitoring infrastructure. Hence, we have assumed the following values for the evidence node P(Observation|Redundant Prevents Network Partitions = *true*)=0.0 and P(Observation|Redundant Prevents Network Partitions =*false*)= 1.0. However, different from our earlier research [23], the DDN approach can take into account those uncertainties what we leave for future work. As in the case of DDNs, the approach presented in [19] can tackle the uncertainty related to lack of confidence of sensor's reports.

Liaskos et al. [15] present a framework for specifying both mandatory and optional requirements, along with quantitative preferences over the optional requirements, within the context of a goal model. The goal tree and the specified preferences are translated into the Hierarchical Task Network (HTN) and Planning Domain Description Language (PDDL) planning formalisms, respectively; the HTNPlan-P planning tool is then used to obtain the most preferred design. Similar to our work, [15] focus on on modeling and reasoning about properties and alternative solutions and working on preferences-based exploration of alternatives requirements. However, they do not use probability theory.

Bayesian networks have been used to enable reasoning over probabilistic causal model and to make predictions about partially satisfied affirmation [6]. However the Bayesian paradigm does not provide any direct means for modelling dynamic systems [21]. In contrast to our model in which we combine Bayesian networks and decision networks to achieve a sophisticated architecture that could be used as a powerful decision-making tool for solving complex or real-time decision problems and to model a system that is dynamically changing or evolving over time such as SASs. A number of interesting and related research approaches using DDNs can be found in the area of AI, Portinale and Raiteri [17] have proposed a formal model for FDIR(Fault Detection, Identification and Recovery) analysis in autonomous systems based on a formal Fault Tree modeling language able to express stochastic dependencies and multi-state components which is called Extended Dynamic Fault Tree (EDFT). In their approach, a compilation process producing from EDFT an equivalent DDN on which to exploit standard DDN algorithms to perform the required FDIR analysis. Their approach is very

relevant in our case because we are using similar model to trigger the adaptations needed by the systems however we have different focus.

6 Conclusion and Future Work

In this paper we have argued how decision analysis of a SAS can be defined as a formal quantitative technique based on Bayesian and decision theory to guide an informed decision making process under uncertainty. Using our approach the best choices to meet a goal are identified from a range of alternatives decisions (i.e., goal realization strategies). Satisficement of softgoals is modeled using conditional probabilities (probability of satisficement of SGj given that a goal realization strategy was chosen). Preferences are modeled using weights associated to pairs of alternative solutions and softgoals. A typical problem arising during the construction of the DDN model is the choice of this quantitative parameters (i.e., weights). More experience in this direction is expected in the near future. Further work is also required towards systematic techniques for studying the value of the probabilities, and even utility weights, that change over time (due to the machine learning process) an their impact on the evaluation of the alternative decisions.

We also want to take advantage of the dynamic structure of the DDNs. We are studying the suitability of DDNs for domains where requirements, goals, and their respective expected values change over time (i.e., during execution). Interesting issues for future research concern the possibility of the utilization of the model with imprecise evidences (e.g. low level of confidence of sensors) to study how the quality of the infrastructure monitoring affect the decisions made by a DDNs. Also the development of new tools to help the requirement engineer to design a DDNs would be certainly very helpful as the current tool support imposes limitations; there is not enough software that supports DDNs.

Acknowledgments. We thank Pete Sawyer and Valerie Issarny for their useful feedback. Also thanks to Andres Ramirez for the support on the use of the RDM case study. This research is partially supported by Marie Curie Fellowship "Requirements@run-time".

References

[1] Norsys software corporation. netica - user guide (1997)
[2] Belaggoun, A.: Exploring the Use of Dynamic Decision Networks for Self-Adaptive Systems. Master's thesis, Univ. de Versailles Saint-Quentin-En-Yvelines (2012)
[3] Cheng, B.H., de Lemos, R., Giese, H., Inverardi, P., Magee, J.: Software engineering for self-adaptive systems: A research roadmap. In: Cheng, B.H.C., de Lemos, R., Giese, H., Inverardi, P., Magee, J. (eds.) Software Engineering for Self-Adaptive Systems. LNCS, vol. 5525, pp. 1–26. Springer, Heidelberg (2009)
[4] Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering, vol. 5. Springer (1999)

[5] da Costa, P.C.G.: The Fighter Aircrafts Autodefense Management Problem: A Dynamic Decision Network Approach. Master's thesis, School of Information Technology and Engineering, George Mason University (1999)
[6] Fenton, N.E., Neil, M.: Making decisions: using bayesian nets and mcda. Knowl.-Based Syst. 14(7), 307–325 (2001)
[7] Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R.: Formal reasoning techniques for goal models. In: Spaccapietra, S., March, S., Aberer, K. (eds.) Journal on Data Semantics. LNCS, vol. 2800, pp. 1–20. Springer, Heidelberg (2003)
[8] Goldsby, H.J., Sawyer, P., Bencomo, N., Hughes, D., Cheng, B.H.: Goal-based modeling of dynamically adaptive system requirements. In: IEEE Int. Conference on the Engineering of Computer Based Systems, ECBS (2008)
[9] Horvitz, E.J., Breese, J.S., Henrion, M.: Decision theory in expert systems and artificial intelligence. Int. Journal of Approximate Reasoning 2, 247–302 (1988)
[10] Howard, R., Matheson., J.: Influence diagrams. In: Readings on the Principles and Readings on the Principles and Applications of Decision Analysis II. Strategic Decisions Group, Menlo Park (1984)
[11] Hughes, D., Greenwood, P., Coulson, G., Blair, G.: Gridstix: Supporting flood prediction using embedded hardware and next generation grid middleware. In: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks, pp. 621–626. IEEE Computer Society, USA (2006)
[12] Lapouchnian, A.: Exploiting Requirements Variability for Software Customization and Adaptation. Ph.D. thesis, University of Toronto (2011)
[13] de Lemos, R., Giese, H., Müller, H., Shaw, M.: Software Engineering for Self-Adpaptive Systems: A second Research Roadmap. In: Software Engineering for Self-Adaptive Systems. No. 10431 in Dagstuhl Seminar Proceedings, Schloss Dagstuhl, Germany (2011)
[14] Letier, E., van Lamsweerde, A.: Reasoning about partial goal satisfaction for requirements and design engineering. SIGSOFT Softw. Eng. Notes 26 (2004)
[15] Liaskos, S., McIlraith, S.A., Sohrabi, S., Mylopoulos, J.: Representing and reasoning about preferences in requirements engineering. Requir. Eng. 16(3), 227–249 (2011)
[16] Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann Publishers Inc., San Francisco (1988)
[17] Portinale, L., Raiteri, D.C.: Using dynamic decision networks and extended fault trees for autonomous fdir. In: ICTAI, pp. 480–484 (2011)
[18] Qureshi, N.A., Peini, A.: Engineering adaptive requirements. In: Workshop on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2009 (2009)
[19] Ramirez, A.J., Cheng, B.H.C., Bencomo, N., Sawyer, P.: Relaxing claims: Coping with uncertainty while evaluating assumptions at run time. In: France, R.B., Kazmeier, J., Breu, R., Atkinson, C. (eds.) MODELS 2012. LNCS, vol. 7590, pp. 53–69. Springer, Heidelberg (2012)
[20] Russell, S.J., Norvig, P.: Artificial intelligence - a modern approach: the intelligent agent book. Prentice Hall series in artificial intelligence. Prentice Hall (1995)
[21] Russell, S.J., Norvig, P.: Artificial intelligence: A modern approach, 2nd edn. Prentice Hall series in artificial intelligence. Prentice Hall (2003)
[22] Sawyer, P., Bencomo, N., Letier, E., Finkelstein, A.: Requirements-aware systems: A research agenda for re self-adaptive systems. In: Proc. of the 18th IEEE International Requirements Engineering Conference, pp. 95–103 (2010)
[23] Welsh, K., Sawyer, P., Bencomo, N.: Towards requirements aware systems: Runtime resolution of design-time assumptions. In: ASE, pp. 560–563 (2011)