ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/235984560

Dynamic Decision Networks for Decision-Making in Self-Adaptive Systems: A Case Study

Conference Paper · May 2013

DOI: 10.1109/SEAMS.2013.6595498

CITATIONS
15

READS
306

3 authors:

Nelly Bencomo
Aston University
91 PUBLICATIONS **1,641** CITATIONS
SEE PROFILE

Amel BELAGGOUN
Atomic Energy and Alternative Energies Com…
6 PUBLICATIONS **41** CITATIONS
SEE PROFILE

Valérie Issarny
National Institute for Research in Computer …
301 PUBLICATIONS **4,242** CITATIONS
SEE PROFILE

All in-text references underlined in blue are linked to publications on ResearchGate, letting you access and read them immediately.

Available from: Amel BELAGGOUN
Retrieved on: 07 September 2016

Dynamic Decision Networks for Decision-Making in Self-Adaptive Systems: A Case Study

Nelly Bencomo
Inria Paris - Rocquencourt,
78153 Le Chesnay, France
nelly@acm.org

Amel Belaggoun
Inria Paris - Rocquencourt,
78153 Le Chesnay, France
amel.belaggoun@inria.fr

Valerie Issarny
Inria Paris - Rocquencourt,
78153 Le Chesnay, France
valerie.issarny@inria.fr

Abstract—**Bayesian decision theory is increasingly applied to support decision-making processes under environmental variability and uncertainty. Researchers from application areas like psychology and biomedicine have applied these techniques successfully. However, in the area of software engineering and specifically in the area of self-adaptive systems (SASs), little progress has been made in the application of Bayesian decision theory. We believe that techniques based on Bayesian Networks (BNs) are useful for systems that dynamically adapt themselves at runtime to a changing environment, which is usually uncertain. In this paper, we discuss the case for the use of BNs, specifically Dynamic Decision Networks (DDNs), to support the decision-making of self-adaptive systems. We present how such a probabilistic model can be used to support the decision-making in SASs and justify its applicability. We have applied our DDN-based approach to the case of an adaptive remote data mirroring system. We discuss results, implications and potential benefits of the DDN to enhance the development and operation of self-adaptive systems, by providing mechanisms to cope with uncertainty and automatically make the best decision.**

Index Terms—**self-adaptive systems, dynamic decision networks, bayesian networks, uncertainty modeling.**

I. Introduction

Due to the dynamism required by modern software systems, there has been strong interest in dealing with uncertainty as a first-class concept in Software Engineering [1], [2] and more recently in the design and operations of self-adaptive systems (SASs) [3], [4], [5], [6].

As with any software system, a SAS must satisfy its functional requirements – i.e., what the system does – and exhibit quality attributes or non-functional requirements (NFRs) such as performance or reliability [7]. Several studies [8], [9] have shown how these NFRs play a key role in driving run-time adaptation between alternative configurations. The functionality is satisfied on a per-context basis [3], and the configuration of the running system is dynamically chosen according to what is the optimal trade-off among the NFRs in each context. The decision is made by using utility functions to evaluate trade-offs between the NFRs [8]. As a simple example, consider a sensor network with the functional requirement “collect data about a volcano", and the NFRs “conserve battery power” and “collect data frequently”. The NFR “conserve battery power” might be prioritized during a quiescent context, but the NFR “collect data frequently” might be the priority in the context associated with an eruption that appears to be imminent.

Measurement of satisfaction of NFRs is difficult due to their vague or fuzzy nature. NFRs may not be absolutely fulfilled, yet they can be labelled as sufficiently satisficed [7]. Satisficement[1] is used frequently instead of satisfaction in the literature. Furthermore, NFRs usually interfere between them which, makes it difficult to reason about their fulfillment. Probabilistic approaches to model and solve uncertainty in SASs have been seen as a promising new research direction [3], [5], [6]. However, the use of probability in this area has been of limited study. Probability theory can be used to describe the lack of crispness about the satisfiability nature of NFRs. Given a decision that requires a certain configuration, the satisficement of a NFR can be modeled using probability distributions. Some researchers have started to study probability-based approaches to tackle the impact of the changes in the environment on the compositions of services and therefore the quality properties or QoS of the the service-based applications [10]. Different from [10] whose authors use Discrete Time Markov Chains (DTMC), we use another probability-based technique called dynamic decision networks (DDNs) what allows us to support the decision-making process of a SAS.

In this paper, we identify compelling reasons for DDNs [11], a form of Bayesian networks, in supporting decision-making of self-adaptive systems. DDNs provide mathematically sound techniques to explicitly model uncertainties that are intrinsic in SASs. The graph-based structure of a DDN matches that of a SAS. It is possible to cast the decision-making of a SAS as a DDN by associating (1) decisions in the DDN with design alternatives and (2) conditional probabilities tables with the effects of those decisions over levels of satisficement of NFRs. The required utility functions to support decision-making using (1) and (2) are based on the expected utility function of a DDN, which also allows analysts to fit their

[1]Satisficement and satisficing are portmanteau words of satisfy and suffice, and refer to a decision-making strategy that attempts to meet an acceptability threshold, like in the case of NFRs, which contrasts with optimal decision-making.

preferences and priorities with respect to the quality attributes. Furthermore, we also argue that DDNs can be used to reflect dynamic changes in the running system due to changes in the environment by means of probabilistic inference based on observed evidence variables [12].

The paper is organized as follows. In the next section, we give a short background on DDNs and their role in tackling uncertainty in SASs. Section III provides a preliminary evaluation of the approach applied on the self-adaptive Remote Data Mirroring application taken as our case study. Section IV provides an overview of the related work in the area. Finally, the paper concludes and offer a discussion of the future perspectives.

II. Background

In this section, we first offer some background about dynamic decision networks and continue with a discussion about needed assumptions for modelling and reasoning about self-adaptive systems under uncertainty and using DDNs.

A. Dynamic Decision Networks

A Bayesian network (BN) [13] is a probabilistic model that represents a set of random variables or chance nodes and their conditional dependencies (i.e., probabilities of the status of one node given the status of others). An important property of BNs is *Bayesian inference*, which refers to the fact that probabilistic beliefs about random variables can be updated automatically as additional evidence *e* is learned [11].

Formally, the structure of a Bayesian network is represented as a triplet (N, E, P), where N is a set of nodes, E $\subseteq N \times N$ is a set of arcs, and P is a set of probabilities [1]. Each node in N is labeled by a random variable X_i with $i = 1...|N|$. Each random variable X_i takes values from a discrete domain[2] and is assigned a vector of probabilities (also called Beliefs). Each probability $P_X(x_i)$ represents a "grade of belief" that X will take the value x_i. $D = (N, E)$ is a Directed Acyclic Graph (DAG) such that a directed arc $e =< a_i, b_i >\in E$ represents causal influence from the source node a_i to the target node b_i. For each node b_i, the strength of causal influence from its parent nodes a_i are quantified by a conditional probability distribution $P(b_i|a_i)$ specified in an $m \times n$ edge matrix, where m and n are the number of discrete values possible for a_i and b_i respectively.

Decision networks (DNs) [14] extend Bayesian networks to provide a mechanism for making rational decisions by combining probability and utility theory. In addition to chance nodes of the BN, a DN also includes utility and decision nodes. Decision nodes represent the set of choices of the decision maker while utility nodes are used to express preferences among possible states of the world represented by the chance nodes, and the decision nodes. A utility function $U(X, d_j)$ defines the usefulness or desirability of the result of making the decision d_j for each value x_i of the random variable X [14]. To decide among alternative decisions, the probability-weighted expected utility EU_j of each decision d_j given evidence e is calculated using the following equation [14]:

$$\mathbf{EU(d_j|e)} = \sum_{\mathbf{x_i \in X}} \mathbf{U(x_i, d_j)} \times \mathbf{P(x_i \mid e, d_j)} \tag{1}$$

Where $P(x_i \mid e, d_j)$ is the conditional probability of $X = x_i$ given the evidence collected e and the decision d_j. The decision with the highest EU associated is chosen.

The probabilistic models BNs and DNs do not offer mechanisms for representing temporal relations between and within the random variables. Instead, DDNs can be used to represent variables that change over time.

DDNs allow us to model decision-making for situations in which decisions, variables that describe the world and preferences can change over time. DDNs provide a principled approach to make rational decisions in the face of uncertainty within changing environments over time. 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 from a previous time slice to a node in the next time slice encodes an influence on the node's value from the previous node and is called the transition model (i.e., the probability $P(X_{t+1}|X_t, d_t)$). The arc connecting E_t and X_t refers to the conditional distribution $P(E_t|X_t)$ which is called the observation model (sometimes known as the sensor model). E_t denotes the set of observable evidence variables. The observation at time t is denoted by E_t = e_t for some set of values e_t [11]. The observation model describes how sensors (i.e. the evidence variables) are affected by the actual state of the world [11].

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 (decision, chance and utility nodes) and several time slices.

B. Modeling Uncertainty in SASs using DDNs

Different sources of uncertainty in self-adaptive systems have been identified and studied [6], [15], [16]. In this paper, we are concerned with the uncertainty associated with the satisficement of NFRs given a set of design decisions reflected in a system configuration. The approach can be applied during runtime when new evidence about certain events are collected which may require a reconfiguration of the system.

Different design alternatives have different impacts (positive and negative) on the satisficement of NFRs. Thus, the configuration of a system can be determined according to an optimal trade-off of NFRs using utility functions in a given context of operation. Furthermore, using the utility function, different priorities or weights can be associated with the NFRs [9].Given the above, NFRs have been identified as key drivers for runtime adaptation between pre-established design alternatives (i.e., configurations). For example, authors of [9] have shown how Claims (i.e., assumptions made at design time) can support decision-making. The rationale behind is that

[2]In this paper we focus on discrete random variables.

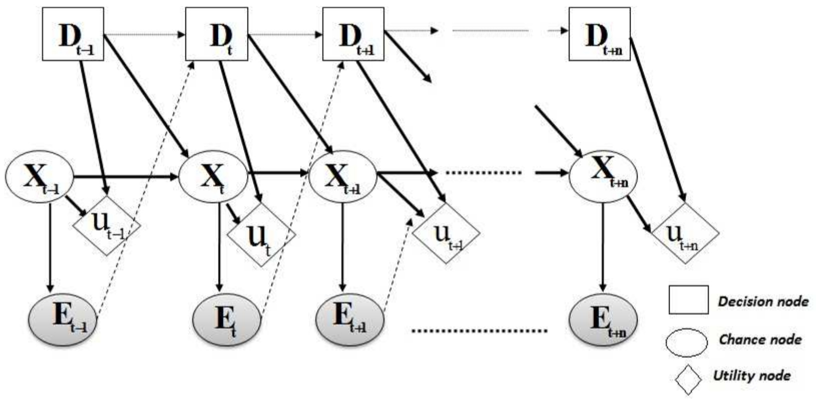

Fig. 1. The General structure of DDN.

the validity of a Claim can be monitored at runtime. The result of the change of the validity of a Claim can provoke or trigger an adaptation to reach more suitable system configurations for the new operating context of the system.

In this paper, we propose a new alternative approach to underpin the decision-making for self-adaptation under uncertainty and driven by NFRs, which is based on the mathematical model provided by DDNs. In our approach, decisions in the DDN correspond with alternative design decisions d_j that define the different configurations a SAS can undertake. The random variables associated with the chance nodes in the DDN represent the levels of satisficement of NFRs given the different design decisions (i.e., configurations). Thus, the conditional probabilities in the DDN correspond with the effects of the different design alternatives over the NFRs which are expressed as $\mathrm{P}(\mathrm{NFR}_i|d_j)$. For each NFR_i, a utility function $\mathrm{w}(\mathrm{NFR}_i,d_j)$ over every design alternative d_j considered is identified.

The global expected utility of a DDN provides the utility function that takes the values of the conditional probabilities and the preferences and priorities over the NFRs (according to contexts) as the weights. Given $\mathrm{P}(\mathrm{NFR}_i|d_j)$ and $\mathrm{w}(\mathrm{NFR}_i,d_j)$, we can compute the probability-weighted average utility for each design alternative d_j - otherwise known as the expected utility of d_j.

The initial conditional probabilities are either estimated by experts or compiled from previously gathered statistics [1]. These conditional probabilities will be updated by the DDN using probabilistic inference (i.e., Bayesian updating). Probabilistic inference occurs whenever new evidence arrives. As a result, DDNs can provide a quantitative technique to make informed decisions due to the arrival of new evidence during either runtime or during a process to explore the operating environment to elicit requirements.

Changes on the conditional probability functions and its values (i.e., beliefs) due to learned information can cause the need to reevaluate the DDN. Therefore, an important step when modeling a DDN to support decision-making for self-adaptation is the definition of the observation model. Environmental properties (i.e., properties of the operational context) that can cause changes on the probability distributions and therefore on the conditional probabilities of the chance nodes need to be identified accordingly. We call those environmental properties, uncertainty factors. These environmental properties are monitored by monitorables to produce the evidences needed to trigger the decision-making process given the changes in the systems beliefs. Monitorables are supported by sensors in the monitoring infrastructure that can observe and provide information to determine the values of the environmental properties that correspond to the uncertainty factors. These uncertainty factors are associated with evidence nodes linked to NFRs (i.e., they constitute part of the observation models of the DDN). Note that not every NFR will have an evidence node. In earlier work [17], we explained the initial ideas the gave place of the DDN approach. The work presented in [17] explain the mapping from a goal model that supports decision-making to a dynamic decision networks.

The scope of uncertainty we deal with in this paper is twofold. The first source of uncertainty has to do with not knowing the exact impact of design alternatives on NFRs, i.e., lack of ability to precisely specify the impact as a crisp value. The second source of uncertainty that can be tackled with our approach is that associated with the accuracy offered by the monitoring infrastructure (i.e., monitorables).

III. Experiments

This section describes a partial evaluation for demonstrating the value of our DDN-based approach to support decision-making for self-adaptation. First, we describe the case study used. Then, we show and discuss the application of DDNs on the case study. Finally, we discuss the experimental results.

A. *Remote Data Mirroring*

Remote Data Mirroring (RDM) [18], [19] is a classic technique to protect data against inaccessibility, and to provide resistance to data loss. Using RDM techniques, copies of important data are kept at physically isolated locations. An RDM application can be configured according to (i) alternative network topologies such as "Minimum Spanning Tree" (MST) and "Redundant Topology"; and (ii) the way how data distribution is achieved, e.g., "synchronous" and "asynchronous". Each configuration provides different levels of data availability, performance and costs. For example, on the one hand, the "synchronous data distribution" provides better data availability than the "asynchronous mode", but it also incurs a potentially high network performance cost as every change must be distributed across the network. On the other hand, the "asynchronous mode" provides higher levels of network performance; however, it provides weaker data availability than the "synchronous mode". Similarly, different network topologies pose different costs and benefits and therefore different trade-offs need to be done when choosing one. A "redundant topology" offers a higher level of reliability than a "MST topology". However, the costs of maintaining a non-stop "redundant topology" may be prohibitive. Given the above, the NFRs identified for the RDM application are respectively: "Minimize Operational Expense", "Maximize Data Reliability" and "Maximize Network Performance" [19].

The adaptation capabilities of an RDM application may be implemented using static rules that correspond to decisions made at design time. However, that means the application would present the behavior of a conventional reconfigurable system. Furthermore, an effective pre-defined solution would be dependent on the requirements analyst anticipating and enumerating all possible environmental states and the corresponding behaviour required of the RDM application, which may not be feasible due, in part, to uncertainty in the process. Instead, in this paper, we consider that the RDM application itself is able to make the adaptation decisions during its execution and according to changes in monitored properties of the environment that may pose specific counteracting behaviour. For example, the system may be required to self-adapt dynamically in response to adverse environmental conditions that can be known only during execution, such as network link failures rate, repeatedly dropped network messages, or periods of unreliable monitoring data.

B. DDNs for RDM System

We now propose the decision-making model for the RDM system explained above using DDNs. Fig. 2 shows the DDN's structure for the remote data mirroring system (unrolled with three time-slices). This structure is explained in detail in the following.

At each time slice t of the DDN, there are six nodes:

(a) the Decision node D_t, the chance nodes related to the NFRs modelled: (b) "Maximize Reliability" MR_t, (c) "Maximize Performance" MP and (d) "Minimize Operational Costs" MO, (e) the Utility node u_t, and (f) the Evidence node E_t. The random variables associated with the chance nodes MP, MO and MR_t are discrete and they take values either *true* or *false*.

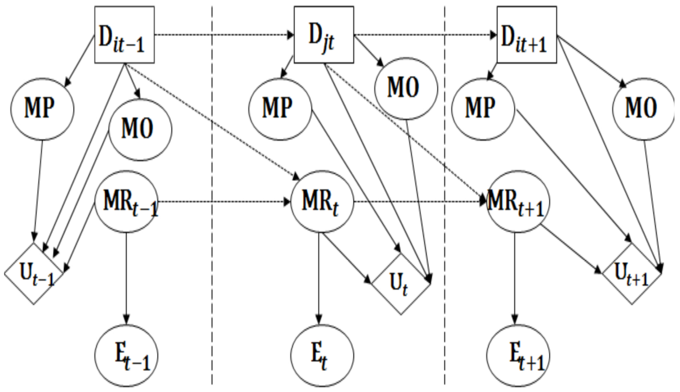

Fig. 2. DDN for the case of the RDM application (unrolled with three time-slices).

The observation model for the case study of the RDM, i.e., the way how evidences are obtained, is partially supported by the mechanism presented in [9], [19]. In [9], Claims are used to record the rationale for a design decision made with incomplete information. The hypothesis is that part of the incomplete information can be learned at runtime and thus other design decisions may be found to be more suitable for the new situation discovered. When a Claim is not valid anymore the system may consider re-evaluating the current situation and check the impacts of several design decisions on the NFRs to decide if an adaptation should be performed. Specifically, the process of learning new evidences to evaluate the DDN is supported by monitorable Claims. Let us focus on the design assumption C1= "Redundancy prevents networks partitions" whose validity can be monitored at runtime. This assumption C1 is falsified if two or more network links fail simultaneously [19].

The assumption C1 affects positively the NFR "Maximize Reliability" MR_t, which is the reason why the chance node MR_t is influenced by evidences related to C1, and therefore parameterized with the variable t. This is represented by the Evidence node E_t connected to MR_t (see Fig. 2). Specifically, changes of the validity state of the assumption (from *true* to *false* or vice versa) provoke changes in the Bayesian beliefs of the DDN. As such, it provides evidence of needed re-evaluation of the DDN during the execution. After the re-evaluation of the DDN, the Bayesian beliefs of the DDN (i.e., the probabilities associated with the nodes) in the network (i.e, DDN) are updated. In contrast to MR_t, the other two chance nodes MO, MP were not considered affected by any evidence in the model described.

Given the above, each type of node in the DDN is described in detail as follows:

(a) The decision node D_t represents the decision taken when specifying the topology to be used in time slice t. The possible decisions in this case study are : Use MST Topology and Use Redundant Topology.

(b) The node MR_t represents the NFR "Maximize Data Reliability". In contrast to (c) and (d), we have modeled this node as a dynamic node, which means that its probability distribution can be affected by the temporal dimension due to the uncertainty factors identified above. The transition model $P(MR_{t+1}|MR_t, D_t)$ needs to be represented by the conditional probability values, usually presented in a conditional probability table (CPT). Domain experts are required to fill in the initial values of that table. The DDN will update the CPT in the subsequent slice times.

(c) The chance node MP represents the NFR "Maximize Network Performance" of the RDM application. We have assumed here that the node MP is a static node, which means that its probability distribution does not change over time.

(d) The chance node MO represents the NFR "Minimize Operational Costs". We have also modeled this node as a static node.

(e) The utility node u_t in a DDN represents the utility function to be used to compute the expected utilities. The utility node has as parents all nodes describing the outcome that directly affect utility. Each utility node has an associated utility table with one entry for each possible instantiation of its parents

(f) The evidence node E_t represents the information observed by monitorables. In our case study, this monitorible verifies the validity of the design assumption C1. The observation model for the case study is represented by $P(E_t|MR_t)$.

Evaluating the DDN to make decisions. Having set up the structure of the DDN model, we formulate the inference computation that must be solved in order to make a decision with the DDN. Let us denote the set of decisions and the observations from time 1 to time t as $D_{1:t}$ and $E_{1:t}$, respectively. The DDN is evaluated using formula (2) to support the decision-making process (see Fig. 3).

C. Evaluation

We have performed three kinds of experiments to test the feasibility of the DDN-based approach for the case of the RDM application.

Specifically, we have simulated the decision-making process of the RDM application based on the experience presented in [19]. The decisions made by our tests have not been executed on a real RDM application. Instead, the experiments have been carried out using the Netica development environment (http://www.norsys.com) [20]. Netica is a software to model and run Decision and Bayesian Networks. Next, the results of the experiments are presented and discussed.

The generic scenario that has been used two perform the experiments is as follows: Let us assume that there are two possible architectural options, "Use MST Topology" or "Use Redundant topology". The state of the design assumption "C1 = Redundancy prevents the networks partitions" is monitored. The value of C1 can be either *true* or *false*. At design time, C1 has been considered valid (*true*). However, during runtime a change on this value is monitored, specifically at time slice $t = 3$, the value *false* is observed, which means that the design assumption has been falsified. Later, specifically at time slice $t = 7$, according to the monitoring infrastructure the design assumption C1 is *true* again.

Given the above scenario we have performed three experiments to see how the DDN makes decisions when new evidences that require the re-evaluation of the DDN are observed and therefore how the application switches from one configuration to another. New evidences that require the re-evaluation of the current decision of the DDN correspond with changes in the environment such as network link failures and unreliable monitoring data.

1) Experiment 1- Decision-Making: In the first experiment, we have examined the role of DDNs to trigger adaptations needed by a SAS. In order to evaluate the DDN shown in Fig. 2, we have considered the following initial conditional probabilities P(NFR$_i|d_j$), which were explained in section II.B and which are based on the experience from study shown in [19]:

P(MP =*true* |MST Topology)= 0.5,
P(MP =*true*|Redundant topology)= 0.5,
P(MO =*true* |MST Topology)= 0.75,
P(MO =*true*|Redundant topology)= 0.25.
$P(MR_t$=*true*|MST Topology)= 0.25,
$P(MR_t$=*true*|Redundant Topology)= 0.95.

The utility values to be used in the computation of the expected utilities EUs are shown in Table I. Specifically, each row in the Table represents the utility value associated with a decision (topology chosen) and its effects on the chance nodes (i.e., NFRs). Given a row and its decision, a value T (*true*) for a chance node states that the decision has a positive effect on the NFR represented by the chance node. Otherwise a value F (*false*) states that the decision has a negative effect on the NFR. As an example, see the second row of Table I, which states that the decision "Use MST Topology" has a negative effect on "Maximize Reliability" and "Maximize Performance" and a positive effect on "Minimize Operational Costs" expressed as the triple (F, F, T).

The last two columns, Utility 1 and Utility 2, represent different weights associated with the sets of effects offered by the decisions that we use in the following experiments. Using those utility values, analysts express their preferences and priorities over the satisficement levels required for the NFRs. The range for the utility values in Table I is [0 . . . 100]. To show how analysts state their preferences, let us consider the preferred combinations of effects characterized with the highest values 90 and 100 for the rows 14 and 16 in the column of Utility 1. As a contrast, the less favorite combination of effects correspond with rows 1, 3, 9, 10 and 11 with the utility values 0, 5, 0, 5, and 1 respectively.

Note the favorite combinations of rows 14 with values (T, F, T) and 16 with values (T,T,T) describing the effects on MR_t, MP, MO, respectively. See that with those values, the analyst is favoring the NFRs MR_t and MO but s/he is rather neutral about the performance property MP.

$$EU(D_t|E_{1:t}) \quad = \sum_{mrt+1} u(MR_{t+1}).P(MR_{t+1}|E_{1:t}, D_{1:t}) + \sum_{mp} u(MP).P(MP|E_{1:t}, D_{1:t}) + \sum_{mo} u(MO).P(MO|E_{1:t}, D_{1:t}) \quad (2)$$

where $EU(D_t|E_{1:t})$ is the expected utility of the decision D_t given the evidence $E_{1:t}$, $P(MR_{t+1}|E_{1:t}, D_{1:t})$ can be computed using equation (3), $P(MP|E_{1:t}, D_{1:t})$ corresponds to the conditional probability of the node "Maximize Performance" (MP) and $P(MO|E_{1:t}, D_{1:t})$ corresponds to the conditional probability of the node "Minimize Operational Costs" (MO), $u(MR_{t+1})$,$u(MP)$ and $u(MO)$ correspond to the utility functions on the nodes "Maximize Reliability" at time *t+1* (MR_t), "Maximize Performance" (MP), and "Minimize Operational Costs" (MO) respectively.

It is shown in [11] that $P(MR_{t+1}|E_{1:t}, D_{1:t})$ can be computed as follows using Markov property:

$$P(MR_{t+1}|E_{1:t}, D_{1:t}) = \sum_{mr_t} P(MR_{t+1}|MR_t, D_t).P(MR_t|E_{1:t}, D_{1:t-1}) \quad (3)$$

Where $P(MR_{t+1}|MR_t, D_t)$ is the transition model and the value of $P(MR_t|E_{1:t}, D_{1:t-1})$ in (3) can be computed as follows:

$$P(MR_t|E_{1:t}, D_{1:t-1}) = \alpha P(E_t|MR_t) \sum_{mr_{t-1}} P(MR_t|MR_{t-1}, D_{t-1}).P(MR_{t-1}|E_{1:t-1}, D_{1:t-2}) \quad (4)$$

In the equation(4), $P(E_t|MR_t)$ is the observation model, $P(MR_t|MR_{t-1}, D_{t-1})$ is the transition model, and α is a normalization constant that ensures the probabilities sum up to one.

The optimal decision suggested by the DDN at time slice *t* is the decision that maximizes the expected utility [11] and is expressed as follows:

$$\arg max_j [EU_{d_j}(D_t|E_{1:t})] \quad (5)$$

Fig. 3. Evaluation of the DDN.

TABLE I
UTILITY TABLE (PREFERENCES).

	Utility node				
Decision	MR	MP	MO	**Utility1**	**Utility2**
1 Use MST	F	F	F	0	0
2 Use MST	F	F	T	20	20
3 Use MST	F	T	F	5	5
4 Use MST	F	T	T	35	55
5 Use MST	T	F	F	8	8
6 Use MST	T	F	T	70	90
7 Use MST	T	T	F	10	10
8 Use MST	T	T	T	80	100
9 Use Redundant	F	F	F	0	0
10 Use Redundant	F	F	T	5	5
11 Use Redundant	F	T	F	1	1
12 Use Redundant	F	T	T	10	10
13 Use Redundant	T	F	F	15	15
14 Use Redundant	T	F	T	90	90
15 Use Redundant	T	T	F	20	20
16 Use Redundant	T	T	T	100	100

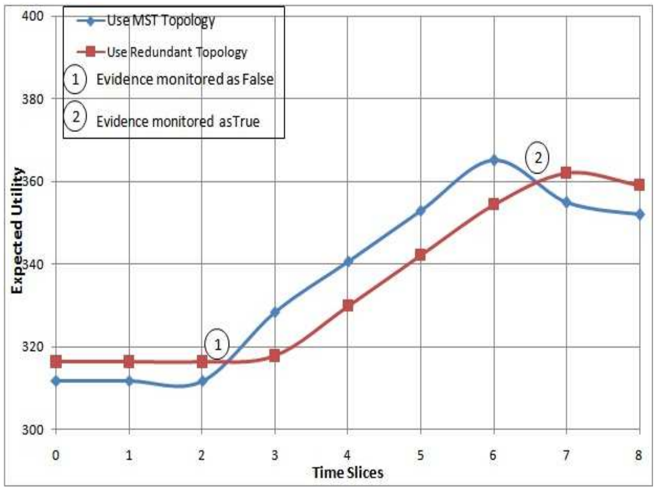

Fig. 4. Expected utilities during eight time slices.

Fig. 4 shows the results of the computation of the expected utility (EU) for each configurations chosen during time slices t=0 to t=8.

This experiment evaluates how evidences about the validity (or falsification) of the recorded assumption C1 = "Redundancy prevents network partitions" can trigger the need of runtime adaptations for the RDM system. Given the initial conditional probabilities and the utilities provided by experts before runtime (i.e., at time slice 0), as expected, the DDN suggests that the best decision is to "Use Redundant Topology" with synchronous propagation as the initial configuration to be chosen. This configuration is based on the validity of the assumption C1 = "Redundancy Prevents Network Partitions"" that states that a redundant network topology prevents network link failures from partitioning the network [16].

In terms of a DDN, this means that with no evidence to contradict the assumption C1 =“Redundancy Prevent Networks Partitions”, the apparent suitable decision is to use “Redundant Topology” as the expected utility EU(Redundant Topology)>EU(MST Topology).

During runtime, specifically at time slice 3, according to the scenario describe above, new information is collected that concludes the assumption C1 is not longer true and therefore EU(MST Topology)>EU(Redundant Topology). Certainly, the decision “Use MST Topology” is considered by the DDN as the best decision this time. It is the most suitable as the use of “Redundant Topology” decision does not necessarily prevent network partitions anymore.

Following the scenario, later on, specifically at time slice 7, the monitoring infrastructure finds the design assumption C1 is valid again (i.e., its value is *true*), and the DDN correctly suggests to adapt to the original design decision “Use Redundant Topology”.

2) Experiment 2- Effects of Weights on Decision-Making: In the second experiment, we examine the effects of weights described in the Utility Table while deriving the best topology(sensitivity analysis). Specifically, we have evaluated the DDN’s sensitivity to these weights on the RDM application. We have used the same scenario presented earlier to compute the best decision during time slice 1 until time slice 8. However, different from the previous experiment where we kept the values of the utility weights constant (using just the column Utility 1), in this second experiment we have assumed that the weights assigned to NFRs can be changed on-the-fly during runtime (using values from both columns Utility 1 and 2). The different set of weights were used at time slice 7 using the values dictated by the column Utility 2 in Table I. In both experiments 1 and 2, the DDN started running with the same initial configuration. The results of the experiment 2 are shown in Fig. 5. The DDN adapts accordingly when new information become available at time $t = 3$, and again and as expected, the DDN selects the “MST Topology” (as this configuration has greater value of EU than the “Redundant Topology”). However, the value *true* of the design assumption is monitored at time slice 7, it can be observed that the two configurations “MST” and “Redundant Topology” have very similar EU values and therefore the DDN decides that no adaptation is required. This effect is due to the newly higher weight associated with the configuration “Use MST Topology” at time slice 7.

As observed, the values of the weights to calculate the expected utilities of decisions can have an important impact on the evaluation of alternative decisions. In this experiment, we have discussed the sensitivity analysis as an important step to check how sentient the decison-maker is to changes on the values of utilities by systematically varying those values when running the DDN. Sensitivity analysis is one of the main criticisms of DDN-based approaches is that of the effort needed during the assessment of the numerical weights required [13]. However, often those values can be successfully specified by expert knowledge elicitation.

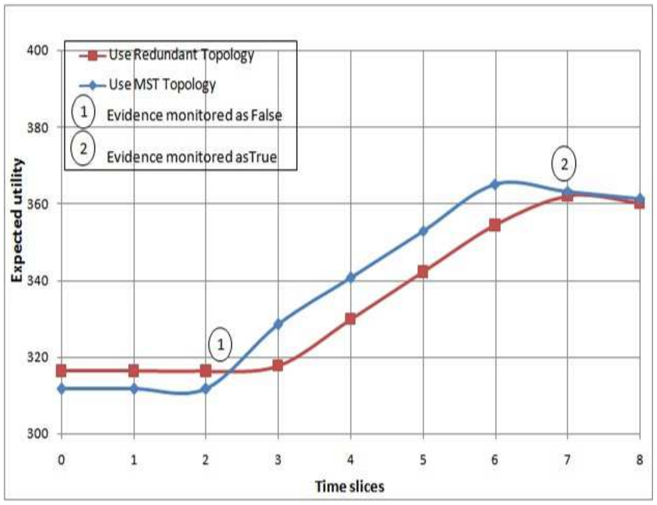

Fig. 5. Impact of the utilities on the selected configuration.

3) Experiment 3- Levels of Confidence on the Monitoring Infrastructure: In our third experiment, we examine how noise from the monitoring infrastructure affects the decisions of a DDN.

A decision made due to an evidence learned using a monitor that is not reliable may not be appropriate. Crucially, the level of confidence over the monitoring infrastructure can be taken into account when using DDNs. In this experiment, we have computed the expected utility of design alternatives (configurations) selected by the DDN during the 8 time slices of the experiment using the same values of the probabilities and the utility table presented in experiment 1. However, different from experiment 1 and 2, here we consider uncertainties about the quality of the observations associated to the evidence node. In other words, we have considered in this experiment errors or noise introduced by the monitoring infrastructure.

To study the effects of such an uncertainty in the DDN, we have considered three levels of confidences associated to each evidence node, and specifically the following three values were considered C = 0.90, 0.80 and 0.4. In the context of the DDN, such values represent the following probabilities:

$$P(\text{e}|\text{C1}=\textit{true}) = 0.9$$
$$P(\text{e}|\text{C1}=\textit{true}) = 0.8$$
$$P(\text{e}|\text{C1}=\textit{true}) = 0.4$$

Fig. 6 (a, b, c) shows the computation of the decisions made by a DDN for the RDM example until time slice $t = 8$ when there is uncertainty about the recorded assumption “Redundancy prevent network partitions” .

Fig. 6a and 6b illustrate the expected utility results for $C = 0.90$ and $C = 0.80$ respectively, it can be seen that the DDN follows the same pattern as in the experiment 1 even if there is uncertainty about the falsification or the validity of the design assumption. Again, at time slice t =3, the DDN suggests “Use MST topology” rather than “Use Redundant Topology” when the assumption is *false* (even with the slightly lower

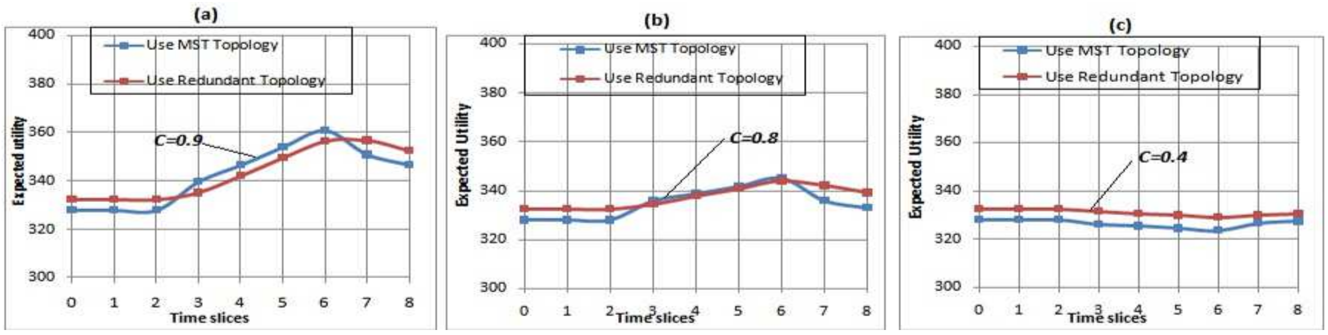

Fig. 6. Impact of uncertainty in the monitoring infrastructure on DDN's decision

levels of confidence 90% and 80%). The DDN suggests to go back to use "Redundant Topology" after time slice 7 when the assumption was recorded as *true*.

Fig. 6c illustrates the expected utility results for $C = 0.40$. In this case, the level of trust is rather low (just 40%). The DDN decides to use the "Redundant Topology" even if the assumption recorded at runtime was falsified or validated.

As we see, Fig. 6a and Fig. 6b show a similar behavior of adaptation of the DDN that was observed in experiment 1. Fig. 6c shows that for low level of trust toward the monitoring infrastructure, the decisions suggested by the DDN may not be meaningful.

We see that, as expected, the introduction of noise in the evidence node does degrade the decision made by a DDN. However, the amount of this degradation is rather small when we consider a high degree of confidence level about quality of observations.

The results of the evaluation of the DDN to RDM system and other case studies like GridStix are reported in [21]. While the result are somehow preliminary, they are also positive as the DDNs allowed applications to adapt to a new situations. Furthermore, the new situations appear to be consistent with the observations that provoked the adaptations to new situations at runtime.

IV. Related Work

There is an increasing need for software systems that are able to adapt dynamically to changes in their environment. However, there is still a dearth of applicable techniques for handling uncertainty [3], [22] in this setting. Several research initiatives have recently aimed to tackle uncertainty in different ways. The related work described here is divided in two categories : (i) tackled uncertainty in the research area of SASs, and (ii) decision-making under uncertainty using Bayesian theory.

A. Uncertainty for Self-Adaptation

Esfahani et al. [23] have proposed GuideArch, a quantitative framework that allows engineers to make decisions using imperfect information. As in our case, GuideArch helps the requirements engineers to make decisions during the RE process. Both approaches deal with the same scope of uncertainty and has to do with not being able to precisely specify the impact of architectural alternatives as a crisp value. However, different from the work presented in [23], our DDN-based approach can also be used to make adaptation decisions at runtime. Furthermore, they represent the anticipated impact of an architectural alternative on the system's properties as a range of values. Specifically, given the partly unknown impact of architectural alternatives on properties, they quantify the overall value of a given architecture using fuzzy logic methods. In our case, we use probabilistic methods to quantify the anticipated impact of an architectural alternative on the system's properties (i.e. NFRs). Besides, the overall value of a given architecture is calculated using the expected utility function provided by the DDN.

Uncertainty in adaptive systems has also been tackled by RELAX [24], a formal requirements language that explicitly addresses uncertainty inherent in adaptive systems. While RELAX uses fuzzy logic to specify more flexible requirements to handle uncertainty, we use probability theory to quantify uncertainty. Another approach is POISED [25] by Esfahani et al., which is based on possibility theory [26] and fuzzy mathematics to assess the consequences of uncertainty. Furthermore, similar to our approach, POISED tackle uncertainty associated with adaptation decisions aimed at satisfying the system's NFRs. As in the case of RELAX, POISED is based on fuzzy mathematics. While RELAX targets the specification of requirements, our approach and POISED aim at supporting decision-making at runtime as well. Our model also differs from these two approaches in the use of Bayesian theory to perform reasoning and decision making under uncertainty at runtime.

Welsh et al. [9] introduce REAssuRE to use goal models and *Claims* (i.e., design assumptions) to support decision-making and drive self-adaptation. In the same context, Ramirez et al. [19] adopt the use of *Claims* and introduced an approach for RELAXing *Claims* that focuses on how uncertainty can affect the validity of assumptions at runtime to avoid unnecessary reconfiguration due to transient conditions. In [27], we explain how our approach can also offer an implementation of RELAX for RELAXing *Claims*.

Letier et al. [28], as in our case, 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 models and interpreted in terms of application specific measures. Their approach is different from ours in some relevant aspects. 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.

Liaskos et al. [29] proposed an extension of goal modeling techniques to support the representation of preferences. Both approaches, the one described in [29] and ours, focus on modeling and reasoning about priorities and alternative solutions and working on preference-based exploration of alternatives requirements. However, they do not use probability theory.

Dynamic configuration of service based systems (SBSs) was studied by Filieri et al. [10]. They conceived KAMI, a framework for runtime modeling of SBSs. Similar to our case, their approach focus on non-functional properties that can be specified quantitatively in a probabilistic way and target the challenge of making adaptation decisions under uncertainty. However, while they use Markov models, we use DDNs. Their focus is on verification and dependability.

Unlike our case, and with the exception of the work presented in [10], none of the approaches described above employ machine learning techniques. In [10], Discrete-Time Markov Chains can be automatically updated by observations of run-time data collected from the environment. Our approach learns and updates the probabilities (i.e., beliefs) over time when new evidence becomes available at runtime.

B. *Decision-Making under Bayesian Theory*

Bayesian networks have been used to enable reasoning over probabilistic causal model and to make predictions about partial satisfaction of non-functional requirements [30]. However, the Bayesian paradigm does not provide any direct means for modelling dynamic systems [11]. In contrast to our model in which we combine Bayesian networks and decision networks to provide a tool to support decision-making 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 related research approach using DDNs can be found in the area of AI, Portinale and Raiteri [31] have proposed a formal model that also uses DDNs 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 that produces an equivalent DDN from the EDFT on which to exploit standard DDN algorithms to perform the required FDIR analysis. Their approach is relevant to our case because we are using a similar model to trigger adaptations. However, we have a different focus; we are tackling the challenge of making adaptation decisions under uncertainty in SASs.

V. Conclusion and Future Work

In this paper we have presented a novel approach that uses the mathematical model of DDNs to support decision-making under uncertainty for self-adaptation. Unlike other related work, DDN-based approaches adopt probabilistic methods (i.e., Bayesian methods) and decision theory to assess the consequences of uncertainty. Our approach provides a holistic view to tackle self-adaptation under uncertainty that covers design time and runtime and also different sources of uncertainty. Using our approach, suitable choices to satisfice functional requirements of the system are identified from a range of alternative decisions and their expected utilities. Satisficement of NFRs is modeled using conditional probabilities given the design decisions. Preferences over decisions are modeled using weights associated with pairs of design alternatives and NFRs and used when computing the expected utilities of the architectural design alternatives. The decision taken by the DDN is that with the highest expected utility. The approach offers the benefits of machine learning. Furthermore, the approach also tackles uncertainty due to unreliable monitoring information. The results achieved so far are rather promising. Different research avenues have been identified and are part of our research agenda.

First, we envisage using our approach to derive quantitative requirements. Quantitative requirements [28] are estimations on quality variables of goals assigned to the software-to-be like cost or response time. These quantitative requirements may be needed to achieve given targets (for example, a maximum cost or a minimum response time). We also envisage an extension of our approach to manage uncertainties while estimating parameters based on confidence intervals for given targets.

Further work is required towards systematic techniques for studying the value of the probabilities that change over time (due to the machine learning process) an their impact on the evaluation of the alternative decisions. We are also interested in the complementary use of the goal models with our approach. The Bayesian learning provided by the DDN-based approach can complement the benefits of goal-based models like self-explanation support [32] and visual analysis for example.

Currently, we are working on a formal Bayesian definition of surprise as the basis for quantitative analysis to measure degrees of uncertainty and deviation of self-adaptive systems from normal behaviour and partial results have been presented in [27]. Specifically, a Bayesian surprise quantifies how new evidence affects assumptions of the world (properties in the models). A “surprising" event may provoke a large divergence between the beliefs distributions prior and posterior to that event. As such and depending on how big or small this divergence is, the running system may decide to either (i) dynamically adapt accordingly or (ii) temporarily avoid any action of adaptation and flag up the fact that a potential abnormal situation has been found. Early partial but promising results are shown in [27].

Further study on how the quality of the infrastructure monitoring, using the level of confidence of sensors, affects the decisions made by the DDN.

We are studying how probability values can also be considered as the basis to implement RELAX-based requirements specifications. We are working on how to relax Claims and compare results with those obtained in the [19]. Finally, development of tools to help the requirement engineer to design a DDN would be certainly very helpful as the current tool support imposes limitations; there are not many tools that support DDNs.

Acknowledgment

We thank Pete Sawyer for his useful feedback. Also thanks to Andres Ramirez for the support on the use of the RDM case study. This research is partially supported by the EU Marie Curie Fellowship Requirements@run-time and the EU FP7 project CHOReOS (257178).

References

[1] H. Ziv, D. J. Richardson, and R. KlŽsch, "The uncertainty principle in software engineering," 19th International Conference on Software Engineering,ICSE'97, boston, Massachusetts, USA.

[2] N. Fenton, W. Marsh, M. Neil, P. Cates, S. Forey, and M. Tailor, "Making resource decisions for software projects," in *Proceedings of the 26th International Conference on Software Engineering*, ser. ICSE '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 397–406. [Online]. Available: http://dl.acm.org/citation.cfm?id=998675.999444

[3] B. H. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, "Software engineering for self-adaptive systems: A research roadmap," in *Software Engineering for Self-Adaptive Systems*, B. H. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, Eds. Springer-Verlag, 2009, vol. 5525, pp. 1–26.

[4] J. Whittle, P. Sawyer, N. Bencomo, B. Cheng, and J.-M. Bruel, "Relax: a language to address uncertainty in self-adaptive systems requirement," *Requirements Engineering*, vol. 15, pp. 177–196, 2010.

[5] D. Garlan, "Software engineering in an uncertain world," in *Proceedings of the FSE/SDP workshop on Future of software engineering research*, ser. FoSER '10. New York, NY, USA: ACM, 2010, pp. 125–128. [Online]. Available: http://doi.acm.org/10.1145/1882362.1882389

[6] N. Esfahani and S. Malek, "Uncertainty in self-adaptive software systems," in *Software Engineering for Self-Adaptive Systems 2 (SEfSAS 2)*. Springer-Verlag, 2012.

[7] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*. Springer, 1999, vol. 5.

[8] H. J. Goldsby, P. Sawyer, N. Bencomo, D. Hughes, and B. H. Cheng, "Goal-based modeling of dynamically adaptive system requirements," in *IEEE Int. Conference on the Engineering of Computer Based Systems (ECBS)*, 2008.

[9] K. Welsh, P. Sawyer, and N. Bencomo, "Towards requirements aware systems: Run-time resolution of design-time assumptions," in *ASE*, 2011, pp. 560–563.

[10] A. Filieri, C. Ghezzi, and G. Tamburrelli, "A formal approach to adaptive software: continuous assurance of non-functional requirements," *Formal Asp. Comput.*, vol. 24, no. 2, pp. 163–186, 2012.

[11] S. J. Russell and P. Norvig, *Artificial intelligence: A modern approach*, 2nd ed., ser. Prentice Hall series in artificial intelligence. Prentice Hall, 2003.

[12] J. Bilmes and J. Bilmes, "On virtual evidence and soft evidence in bayesian networks," 2004.

[13] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[14] R. Howard and J. Matheson., "Influence diagrams," in *Readings on the Principles and Readings on the Principles and Applications of Decision Analysis II*. Menlo Park CA:: Strategic Decisions Group, 1984.

[15] K. Welsh and P. Sawyer, "Understanding the scope of uncertainty in dynamically adaptive systems," in *REFSQ*, 2010.

[16] A. Ramirez, A. Jensen, and B. Cheng, "A taxonomy of uncertainty for dynamically adaptive systems," in *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012 ICSE Workshop on*, june 2012, pp. 99 –108.

[17] N. Bencomo and A. Belaggoun, "Supporting decision-making for self-adaptive systems: From goal models to dynamic decision networks," in *9th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)*, 2013.

[18] M. Ji, A. Veitch, and J. Wilkes, "Seneca: Remote mirroring done write," *USENIX 2003 Annual Technical Conference.*, pp. 253–268, 2003.

[19] A. Ramirez, B. Cheng, N. Bencomo, and P. Sawyer, "Relaxing claims: Coping with uncertainty while evaluating assumptions at run time," *ACM/IEEE Int. Conference on Model Driven Engineering Languages & Systems MODELS*, 2012.

[20] *Norsys Software Corporation. Netica - User guide*, 1997.

[21] A. Belaggoun, "Exploring the use of dynamic decision networks for self-adaptive systems," Master's thesis, Univ. de Versailles Saint-Quentin-En-Yvelines, 2012.

[22] R. de Lemos, H. Giese, H. Müller, and M. Shaw, "Software Engineering for Self-Adpaptive Systems: A second Research Roadmap," in *Software Engineering for Self-Adaptive Systems*, ser. Dagstuhl Seminar Proceedings, no. 10431. Germany: Schloss Dagstuhl, 2011.

[23] N. Esfahani, K. Razavi, and S. Malek, "Dealing with uncertainty in early software architecture," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, ser. FSE '12. New York, NY, USA: ACM, 2012, pp. 21:1–21:4. [Online]. Available: http://doi.acm.org/10.1145/2393596.2393621

[24] P. Sawyer, N. Bencomo, E. Letier, and A. Finkelstein, "Requirements-aware systems: A research agenda for re self-adaptive systems," in *Proc. of the 18th IEEE International Requirements Engineering Conference*, 2010, pp. 95–103.

[25] N. Esfahani, E. Kouroshfar, and S. Malek, "Taming uncertainty in self-adaptive software," in *Proc.of the 19th ACM SIGSOFT Symp FSE*, no. 11, 2011.

[26] L. Zadeh, "Fuzzy sets as a basis for theory of possibility," *Fuzzy Sets and Systems*, vol. 1, pp. 3–28, 1978.

[27] N. Bencomo, A. Belaggoun, and V. Issarny, "Bayesian artificial intelligence for tackling uncertainty in self-adaptive systems: the case of dynamic decision networks," in *2nd International NSF sponsored Workshop on Realizing Artificial Intelligence Synergies in Software Engineering RAISE*, 2013.

[28] E. Letier and A. van Lamsweerde, "Reasoning about partial goal satisfaction for requirements and design engineering," *SIGSOFT Softw. Eng. Notes*, vol. 26, 2004.

[29] S. Liaskos, S. A. McIlraith, S. Sohrabi, and J. Mylopoulos, "Representing and reasoning about preferences in requirements engineering," *Requir. Eng.*, vol. 16, no. 3, pp. 227–249, 2011.

[30] N. E. Fenton and M. Neil, "Making decisions: using bayesian nets and mcda," *Knowl.-Based Syst.*, vol. 14, no. 7, pp. 307–325, 2001.

[31] L. Portinale and D. C. Raiteri, "Using dynamic decision networks and extended fault trees for autonomous fdir," in *ICTAI*, 2011, pp. 480–484.

[32] N. Bencomo, K. Welsh, P. Sawyer, and J. Whittle, "Self-explanation in adaptive systems," in *ICECCS*, 2012, pp. 157–166.