2018 5th International Workshop on Artificial Intelligence for Requirements Engineering

Validating Goal Models via Bayesian Networks

Davide Dell'Anna, Fabiano Dalpiaz, Mehdi Dastani
Department of Information and Computing Sciences
Utrecht University
Utrecht, The Netherlands
Email: {d.dellanna, f.dalpiaz, m.m.dastani}@uu.nl

***Abstract*—Goal models are an example of requirement modeling language that has been applied to support the runtime monitoring and diagnosis of software systems and to steer self-adaptive systems. When creating a goal model, requirement engineers make *assumptions* concerning how the goals relate to each other and when they should be considered as satisfied. In dynamic environments, however, the assumptions made in the model may be (or become) invalid. This may result in a system that does not satisfy the stakeholders' needs and, when the model is used in adaptive systems, ineffective reconfigurations. Only few and preliminary works address the automated validation of goal or requirement models. In this paper we propose the use of probabilistic models (Bayesian Networks) to determine the validity of the assumptions underlying a goal model. We employ empirical data and probabilistic inference to automatically determine a quantitative degree of validity of goal model assumptions. We illustrate the approach on a smart traffic scenario.**

***Index Terms*—Goal Models; Requirement Monitoring; Bayesian Networks.**

I. Introduction

Designing modern software systems is a complex task. For example, an urban traffic system includes a multitude of entities, such as pedestrians, drivers, cars, bicycles, traffic lights and signs, speed cameras, and road regulations. These entities operate in dynamic settings [1], are weakly controllable, and their behavior can only be partly predicted at design-time. Anticipating all the possible states and transitions is not an option for modern, open (socio-technical) systems [2].

Requirement engineers often make *assumptions* [3] about requirements and their satisfaction conditions. These assumptions concern the relationships between the system, the requirements and the environment in which the system must operate [3]. In a requirement model, for instance, satisfying certain requirement is assumed to guarantee the satisfaction of higher level requirements, the satisfaction of a requirement is assumed to positively or negatively contribute to the achievement of other requirements, and the priorities of non-functional requirements are only estimated [4].

Several frameworks (e.g., [5], [6]) have been proposed to support runtime requirement monitoring and diagnosis. Many of such approaches represent requirements via goal modeling languages such as KAOS [7] or iStar [8]. These tools enable the collection of system execution data and their analysis in terms of requirements satisfaction. Furthermore, runtime requirement monitoring provides the essential inputs for detecting whether and when design-time assumptions concerning requirement satisfaction become invalid [3].

Early work by Ali *et al.* [3], [9] discusses the necessity of assessing at runtime the assumptions underlying goal models. Such runtime assessments concern the conditions for goal satisfaction (e.g., decompositions, contributions) that the requirement engineers make based on their beliefs and knowledge about the system under design and its environment. Based on such baseline, we set the following research question.

Research Question (RQ): *How to use Bayesian Networks to validate the assumptions in a goal model with empirical data?*

In this paper we propose a novel approach that extends and implements the idea of validating design assumptions at runtime [3]. We use a Bayesian Network to collect statistical information about requirement satisfaction and to learn the correlation between goal and softgoal satisfaction in different operating contexts. This information can be obtained at runtime by using existing monitoring frameworks (e.g., [5], [6]), or from existing datasets already available at design-time (e.g., by examining the business processes logs in an organization). The Bayesian Network, automatically generated from a goal model, provides the engineer a tool that can be used to analyze the behavior of a system and to detect misalignment between the assumptions being made and empirical data.

The paper is structured as follows. Sec. II introduces an illustrative example concerning smart traffic. Sec. III explains the supported types of design-time assumptions. We illustrate and revisit the *assumptions* presented by Ali *et al.* [3]. In Sec. IV we introduce *Requirement Bayesian Networks* (*RBN*s) and show how to map a goal model to an *RBN*. Sec. V presents a technique that uses probabilistic inference on a Requirement Bayesian Network to determine the degree of validity of the assumptions underlying a goal model. In Sec. VI we evaluate the feasibility of our work by applying it to the illustrative example. Finally, we present a discussion of the related work in Sec. VII, and some concluding remarks in Sec. VIII.

II. Illustrative Example: Smart Traffic

Suppose the city council of a smart city aims at improving the urban traffic by offering a Central Navigation Service (CNS) as proposed in the goal model illustrated in Fig. 1 (see [10] for a similar example from self-adaptive system literature).

A major goal is identified: *at least 10% of the cars in the city shall always use the offered CNS* (identified with id *NS* in Fig. 1). To satisfy this goal, two sub-goals are assumed to be necessary: *whenever a car starts a trip toward a destination, the*

978-1-5386-8404-7/18/$31.00 ©2018 IEEE
DOI 10.1109/AIRE.2018.00012

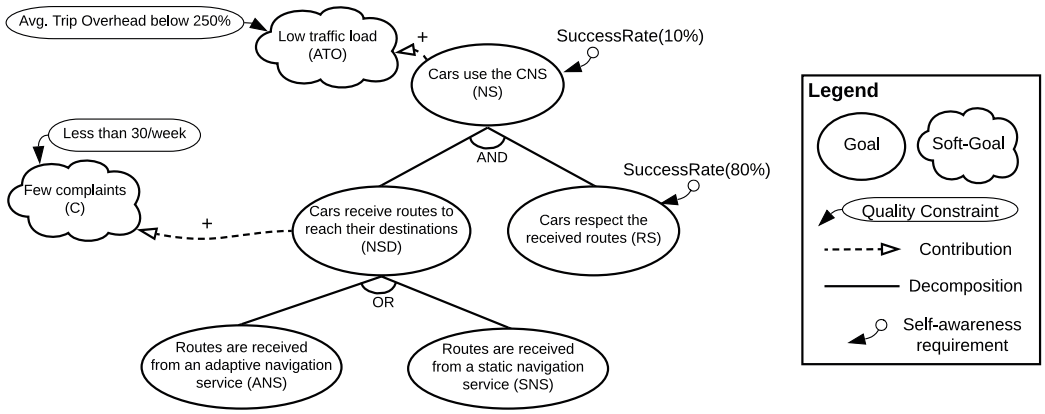

Fig. 1. Illustrative goal model for the smart transportation simulator presented in Sec. II. The text between brackets represents an element's *id*.

car shall receive a route from the Central Navigation Service (*NSD* in Fig. 1) and *at least 80% of all the route suggestions given by the CNS is respected by all the cars equipped with the CNS* (*RS* in Fig. 1). The *NSD* goal can be met by either employing a self-adaptive navigation service (*ANS* in Fig. 1) (e.g., see [10]) or a static navigation service (*SNS* in Fig. 1). In other words, *whenever a car equipped with the CNS starts a trip toward a destination, the car shall receive a route from an adaptive (static) navigation service*. These goals are *assumed* to help achieve two softgoals concerning the global traffic load in the city (evaluated in terms of cars' average trip overhead) and the satisfaction of the users when using the navigation service (evaluated in terms of number of complaints) [10].

In goal models, goals are organized in hierarchies via AND- and OR-*decomposition* links. For example, to achieve the goal *NS*, both goals *NSD* and *RS* shall be satisfied. OR-*decompositions* describe possible *exclusive* ways to achieve a goal. The expected positive or negative impact of goals on softgoals is represented via *contribution* links, shown as dashed directed arrows. Although contributions can connect different node types, in this paper we restrict ourselves to links between goals and softgoals in order to prevent cycles in the model.

We make use of a simplified goal model: we consider only goals and softgoals, leaving out tasks from our discussion. The presented techniques, however, can be generalized and applied at different granularity levels, including tasks. We integrate, instead, our simplified goal model with Souza *et al.*'s [11] *awareness requirements* (*AwReq*s). *AwReq*s qualify the satisfaction of goals. Goals that are not associated to an *AwReq* in Fig. 1 are required to be satisfied by every instance of the referred goal (e.g., an instance of goal *ANS* is created every time a car equipped with the CNS starts a trip, and such instance is satisfied if the car receives a route from an adaptive navigation service). These requirements are called *regular AwReq*s in [11]. Requirements like *SuccessRate*, instead, are called *aggregate AwReq*s: the satisfaction of the associated goal is determined in terms of groups of instances of the goal (e.g. an instance of the goal *NS* is created and evaluated for every car driving in the city, *NS* however is achieved if 10% of such instances is satisfied). Table I describes precisely the conditions for requirement monitors to determine goal satisfaction.

TABLE I
SATISFACTION CONDITIONS OF THE GOALS IN THE RUNNING EXAMPLE.

Goal	Satisfied
NS	>10% cars in the city is using the CNS (eval at every time instant)
NSD	every time a car equipped with the CNS starts a trip it receives a route from the CNS
ANS	every time a car equipped with the CNS starts a trip it receives a route from an ANS
SNS	every time a car equipped with the CNS starts a trip it receives a route from a SNS
RS	at least the 80% of all the suggestions from the CNS has been accepted (eval every week)
ATO	the average trip overhead of all the cars has been below 250% (eval every week)
C	the number of complaints received is below 30 (eval every week)

The system of the running example can operate in four possible operating contexts: *day* or *night* (in the simulator, respectively 600 and 300 cars in the city) under *normal* or *extreme* weather conditions. We call contextual properties the monitorable environmental variables that determine the operating context of the system: *Time* and *Weather*.

III. Design-Time Assumptions

We briefly describe here 6 types of assumptions underlying the structure of a goal model [3], and we illustrate them with respect to our example:

1) *Goal satisfiability assumption*: in a specific operating context, a goal is satisfied.
   Fig. 1 is based on 20 goal satisfiability assumptions: one for each goal in the model for each operating context (e.g., *in context day-extreme, the goal RS is satisfied*).
2) *Softgoal achievement assumption*: in a specific operating context, a softgoal is achieved.
   Fig. 1 is based on 8 softgoal achievement assumptions: one for each softgoal in the model for each operating context (e.g., *in context day-extreme, the softgoal ATO is achieved*).

3) *Contribution assumption*: in a specific operating context, there is a positive (negative) synergy between the satisfaction of a goal and the achievement of a softgoal connected via a *contribution* link.
   Fig. 1 is based on 8 contribution assumptions: one for each contribution link in the model for each operating context (e.g., *in context day-extreme, there is a positive synergy between the satisfaction of goal NS and the achievement of the softgoal ATO*).
4) *Decomposition assumptions*: in a specific operating context, the satisfaction of an AND-decomposed goal depends on the satisfaction of all its sub-goals, and an OR-decomposed goal is satisfied only when one and only one of its sub-goals is satisfied.
   Fig. 1 is based on 4 AND-decomposition assumptions, derived by associating the only AND-decomposition with each of the 4 operating contexts. For example, *in context day-extreme, to satisfy the goal NS both the goals NSD and RS shall be satisfied*. Analogously there are 4 OR-decomposition assumptions.
5) *Adoptability assumption*: in a specific operating context, there is a positive synergy between the satisfaction of a goal and the satisfaction of each one of its sub-goals separately.
   Fig. 1 is based on 16 adoptability assumptions: one for each decomposition link between a sub-goal and a goal, for each operating context (e.g., *in context day-extreme, there is a positive synergy between the satisfaction of the goal SNS and the satisfaction of the goal NSD*).
   Notice that while decomposition assumptions concern *one-to-many* relationships (i.e., between one goal and all of its children), adoptability assumptions concern *one-to-one* relationships (i.e., between a goal and each of its sub-goals separately).
6) *Goal necessity assumption*: in a specific operating context, the activation of a specific goal is necessary condition for achieving all the softgoals.
   Fig. 1 is based on 20 goal necessity assumptions: one for each of the goals in the model for each operating context (e.g., *in context day-extreme, to achieve the two softgoals ATO and C together, the goal ANS must be activated*).
   Note that this assumption concerns the *activation* of a goal, regardless of its satisfaction; i.e., it is the hypothesis that, in order to achieve the softgoals, it is better to keep active a goal rather than disabling it.

The small goal model of the running example, despite its simplicity, contains 80 assumptions that the requirement engineer who constructed it has made! This calls for automated mechanisms that assist requirements engineering in validating such many assumptions.

IV. From Goal Models to Bayesian Networks

In this section we formally define how to automatically generate the structure of a *RBN* from a goal model of the type described in Sec. II. We first provide a formal description of the goal model and of the *RBN* and finally we define the mapping. Notice that in order to generate the structure of a *RBN* we consider only the topological information of the goal model (i.e., goal and softgoal nodes and contribution and decomposition links). Information concerning the satisfaction of the goals (i.e., *AwReq* and quality constraints) must be used, instead, to generate a monitoring system to produce data that will train the *RBN* probability distributions. For this reason, for the sake of simplicity, in the following section we omit *AwReq* and quality constraints from the formalization of the goal model.

A. Goal Model

A goal model, as described in Sec. II, can be defined as a tuple $\mathcal{GM} = \langle (\mathcal{G}, ch, d), \mathcal{SG}, cl \rangle$, where $(\mathcal{G}, ch, d)$ is the AND-OR tree, with $\mathcal{G} = \{G_1, ..., G_n\}$ set of n goals, $ch : \mathcal{G} \rightarrow 2^{\mathcal{G}}$ function determining the children of a goal, $d : \mathcal{G} \rightarrow \{\texttt{AND}, \texttt{OR}\}$ partial function determining the type of decomposition of goals with children, $\mathcal{SG} = \{SG_1, ..., SG_m\}$ set of m softgoals, and $cl : \mathcal{G} \rightarrow 2^{\mathcal{SG}}$ a function that maps goals to the softgoals that they contribute to.

In order to distinguish between different types of goals, in the following we make use of a function $type : \mathcal{G} \rightarrow \{\texttt{agg}, \texttt{reg}\}$ associating each goal to a value describing the type of *AwReq* associated to it. The values respectively represent *aggregate* and *regular AwReq* as described in Sec. II.

B. Requirement Bayesian Network

In this section we provide details about the type of Bayesian Network (called *Requirement Bayesian Network*, or *RBN*) that we generate for assumption validation. Notice that in the context of Bayesian Networks we use the notation reported in Table II.

TABLE II
A summary of the notation used for BNs

Notation	Description
X, Y, ...	Random variables (italic uppercase)
$\mathbf{X}$, $\mathbf{Y}$, ...	Set of random variables (bold uppercase)
v_1, v_2, ...	Value in the domain of a random variable (italic lowercase)
$\mathbf{x}$, $\mathbf{y}$, ...	Assignment of values to a set of nodes (bold lowercase)
X_v	$(X = v)$, assignment of value v to a random variable X
$\mathbf{X}_v$	Assignment of value v to all nodes in$\mathbf{X} \subseteq \mathcal{X}$
X_{act}	$\neg X_{dis} = \neg(X = disabled)$, the fact: X is not *disabled*
$\mathbf{P}$	Probability distribution
P	Single probability

Let $\mathcal{CP} = \{\mathcal{CP}_i, ..., \mathcal{CP}_k\}$ be a set of monitorable contextual properties of the system (e.g., *Time*, *Weather*), each associated to a domain of values (e.g., *Weather* can be either *normal* or *extreme*). A *Requirement Bayesian Network* $\mathcal{RBN} = (\mathcal{X}, \mathcal{A}, \mathcal{P})$ is a Bayesian Network where:

- $\mathcal{X} = \mathbf{G} \cup \mathbf{S} \cup \mathbf{C}$ is a set of nodes, representing random variables in probability theory. The sets $\mathbf{G}$, $\mathbf{S}$ and $\mathbf{C}$ are assumed to be disjoint. The set $\mathbf{G}$ consists of *goal nodes*. Each node $G \in \mathbf{G}$ corresponds to one goal and has a discrete domain of 3 possible values: *obeyed*, *violated* and *disabled*. The set $\mathbf{S}$ consists of *softgoal nodes*. Each node $S \in \mathbf{S}$ corresponds to a boolean softgoal and has

a discrete domain of 2 values: *true* and *false*. Finally, the set $\mathbf{C}$ consists of *context nodes*. Each node $C \in \mathbf{C}$ corresponds to a contextual property $\mathcal{CP}_i \in \mathcal{CP}$ and can have discrete or continuous domain.

- $\mathcal{A} \subseteq (\mathbf{G} \times \mathbf{G}) \cup (\mathbf{C} \times \mathbf{G}) \cup (\mathbf{C} \times \mathbf{S}) \cup (\mathbf{G} \times \mathbf{S})$ is the set of arrows connecting pairs of nodes. If there is an arrow from node X to node Y, X is said to be a parent of Y.
- $\mathcal{P}$ is a set of conditional probability distributions, each one associated to a node in $\mathcal{X}$ and quantifying the effect of the parents on the node.

In the rest of this paper, the pair $(\mathcal{X}, \mathcal{A})$ is called the structure of the Requirement Bayesian Network $(\mathcal{X}, \mathcal{A}, \mathcal{P})$.

An evidence $\mathbf{e}$ is a revealed (observed) assignment of values for some or all of the random variables in the Bayesian Network, i.e., $\mathbf{e} = \{X_v | X \in \mathbf{X}\}$ with $\mathbf{X} \subseteq \mathcal{X}$ and v a possible value in the domain of the variables. An evidence $\mathbf{c}$ for all the context nodes $\mathbf{C}$ is called a *context*[1]. In general, given the set $\mathcal{X}$ of all the nodes in a Bayesian Network $\mathcal{B}$ and an evidence $\mathbf{e}$, reasoning with $\mathcal{B}$ means to determine the distribution $\mathbf{P}(\mathbf{X}|\mathbf{e})$, with $\mathbf{X} \subseteq \mathcal{X}$ a set of nodes of which we want to discover the probability distribution (e.g., $\mathbf{P}(NSD|W_{extreme})$ is the probability distribution of the values of the random variable *NSD*, given that *extreme weather* is observed).

C. From Goal Models to Requirement Bayesian Networks

We formally define how to obtain the *structure* of a $\mathcal{RBN}$ from a goal model $\mathcal{GM}$. This is done by defining a function *GM2BNS* that maps a goal model $\mathcal{GM}$ and a set of contextual properties $\mathcal{CP}$ to an $\mathcal{RBN}$ structure $(\mathcal{X}, \mathcal{A})$. In order to define this function, we denote the set of contributing descendant nodes of a softgoal S in $\mathcal{GM}$ as *cont_desc*(S), where a node G is assumed to contribute to a softgoal S if G is a descendant of S and, moreover, G is either a leaf goal or has the type *aggregate* but does not have any ancestor with type *aggregate*. Formally, we have *cont_desc*(S) = (*desc*$(S) \setminus \{G' | G' \in$ *desc*$(G), G \in$ *desc*$(S),$ *type*$(G) =$ agg$\}) \setminus \{G | ch(G) \neq \emptyset,$ *type*$(G) \neq$ agg$\}$, where $decs(G)$ is the set the descendant of G (similar for S).

Given a goal model $\mathcal{GM} = \langle (\mathcal{G}, ch, d), \mathcal{SG}, cl \rangle$ and a set of contextual properties $\mathcal{CP}$, we have *GM2BNS*$(\mathcal{GM}, \mathcal{CP}) = (\mathcal{X}, \mathcal{A})$, where

- $\mathcal{X} = \mathcal{G} \cup \mathcal{SG} \cup \mathcal{CP}$
- $\mathcal{A} = \{(G, S) \mid G \in$ *cont_desc*$(S)\} \cup$
  $\{(G_1, G_2) \mid G_1 \in ch(G_2)\} \cup$
  $\{(C, G) \mid C \in \mathcal{CP}, G \in \mathcal{G},$ (*type*$(G) =$ agg $\vee\ ch(G) = \emptyset$)$\} \cup$
  $\{(C, S) \mid C \in \mathcal{CP}, S \in \mathcal{SG}\}$

Intuitively $\mathcal{A}$ contains 1) an arrow from a goal node G to a softgoal S if G is a contributing descendant of S (see function *cont_desc* above), 2) an arrow from a sub-goal G_1 to its parent goal G_2, 3) an arrow from a context node C to each goal node G that represent either a leaf goal ($ch(G) = \emptyset$) or a goal with an *aggregate AwReq*, and 4) an arrow from each context node C to each softgoal S.

[1]When we refer to nodes of a specific type we use the corresponding notation convention, e.g., N refers to a node in $\mathbf{N}$, $\mathbf{c}$ refers to a configuration of values of nodes in $\mathbf{C}$, $\mathbf{N}_{viol}$ refers to an assignment of value *violated* to a set of norm nodes $\mathbf{N}$, etc.

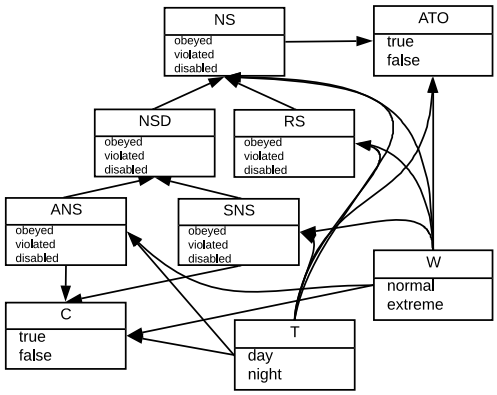

Fig. 2. The structure of the $\mathcal{RBN}$ obtained from the mapping *GM2BNS* presented in Sec. IV-C. The *name* of each node corresponds to their *id*.

Fig. 2 reports the structure of the $\mathcal{RBN}$ automatically generated by the procedure described in this section for the running example.

Notice that besides reflecting the goal model's topology, the network also introduces the context variables. The structure of the network, which consist of three types of nodes (goal, soft-goals and context nodes), allows to analyze the assumptions in different operating contexts. In an $\mathcal{RBN}$, every context node is parent of all the soft-goals nodes. This allows to represent the intuition that the achievement of soft-goals is not only due to the satisfaction (or presence) of goals, but also to events that occurr in the environment. Context nodes are also parents of all the goal nodes whose satisfaction is not exclusively determined by their decomposition (e.g., a goal that is AND-decomposed into two sub-goals is satisfied when both sub-goals are satisfied) but can also be affected by the context in which they are applied.

The choice of a discrete domain of three values for the goal nodes makes the network more versatile: while the *obeyed* and *violated* values allow to evaluate assumptions concerning the satisfaction or violation of goals (e.g., *goal satisfiability assumption*), the *disabled* value allows the network to support OR-decomposed goals. In order to update the conditional probability distribution of a node, it is necessary to provide an evidence for both the node and all its parents. In case of an OR-decomposed goal, it is available an evidence only for one of the parents (sub-goals) at a time. The *disabled* value allows therefore to perform the update also in such case.

Notice finally that *GM2BNS* only produces the structure of the Bayesian Network, without providing any initialization of its parameters. The conditional probability distributions of the nodes must be instead learned from data.

TABLE III
PART OF THE DATASET USED TO TRAIN THE BN OF FIG. 2 AND OBTAINED FROM MONITORING THE EXECUTION OF THE SYSTEM IN SEC. II.

W	T	NS	NSD	ANS	SNS	RS	ATO	C	D	O
norm	night	viol	ob	dis	ob	viol	T	T	T	T
norm	day	ob	ob	ob	dis	ob	F	F	T	F
norm	day	ob	ob	viol	dis	viol	F	F	T	F
extr	night	viol	ob	dis	ob	ob	T	T	T	T
extr	day	ob	dis	dis	dis	dis	T	F	T	T
extr	day	ob	ob	dis	ob	viol	T	F	T	T
...										

D. Populating the RBN: Data collection

Table III reports a dataset example that can be obtained from monitoring the satisfaction of goals and softgoals during executions of the system implementing the running example. Notice that the values that each of the variable assumes belongs to its domain as specified in Sec.IV-B (e.g., *obeyed*, *violated*, *disabled* for goal nodes, *true* or *false* for softgoal nodes). Such type of dataset can be used to train the *RBN* and learn the set of conditional probability distributions $\mathcal{P}$. We omit here a discussion about the learning technique (e.g. classical Bayesian learning) to use (which is out of the scope of the paper) and we remind the reader to the existing literature (e.g., [12], [13]). Furthermore in this paper we do not focus on the mechanisms to monitor the system and retrieve data (e.g., by using EEAT monitoring framework [6] to monitor the goals as expressed in Table I). In the following we assume we dispose of a trained *RBN*, and we define a mechanism to analyze its content and to automatically identify erroneous assumptions.

V. VALIDATING ASSUMPTIONS

We propose a mechanism that uses an *RBN* trained with runtime system execution data in order (i) to determine to what extent the assumptions underlying the system's goal model are valid, and (ii) to show the validity on the goal model.

We introduce the notion of *degree of validity* (δ in the following) for an assumption as a real number in the range $[-1, 1]$. $\delta = 1$ denotes a fully valid assumption, value $\delta = -1$ indicates a fully incorrect assumption, and the intermediate values describe an assumption with partial validity. Below, we define how to calculate the degree of validity for each of the assumption types listed in Sec. III. The validity degree is computed as a difference between two probabilities, representing the collected positive and negative evidence for the validity of that assumption, respectively. Thus, if the collected positive evidence is close to 1 and the negative evidence close to 0, the degree of validity assumes values close to +1. Values around 0 show that the assumption is only partly valid since the positive and negative evidences for the validity have similar likelihood.

A. Degree of Validity of Assumptions

Given a context $\mathbf{c}$, we define the validity degree of the six assumption types presented in this paper as follows:

1) *Goal satisfiability assumption*. Given a goal node G, the degree of validity of the associated goal satisfiability assumption in context $\mathbf{c}$ is

$$\delta_S(G, \mathbf{c}) = P(G_{ob} \mid \mathbf{c}) - P(G_{viol} \mid \mathbf{c})$$

2) *Softgoal achievement assumption*. Given a softgoal node S, the degree of validity of the associated softgoal achievement assumption in context $\mathbf{c}$ is

$$\delta_G(S, \mathbf{c}) = P(S_{true} \mid \mathbf{c}) - P(S_{false} \mid \mathbf{c})$$

3) *Contribution assumption* Given a goal node G and a softgoal node S, the degree of validity of a positive contribution assumption is:

$$\delta_C(S, G, \mathbf{c}) = P(S_{true} \mid G_{ob} \wedge \mathbf{c}) - P(S_{true} \mid G_{viol} \wedge \mathbf{c})$$

Note that the degree of validity of negative contribution assumptions, due to the boolean nature of the softgoal nodes, can be calculated as $-\delta_C$.

4) *Decomposition assumption* Given a goal node G and the set **G'** $\in$ **G** of its goal nodes parents, let **g** be the disjunction of all possible assignments of values to variables in **G'** excluding the assignment **G'**$_{ob}$, let **g1ob** be the disjunction of all possible assignments of values to variables in **G'** such that only one variable takes value *obeyed*, and let **go** be the disjunction of all possible assignments of values to variables in **G'** excluding the assignments in **g1ob**.

$$\delta_{AND}(G, \mathbf{c}) = P(G_{ob} \mid \mathbf{G'}_{ob} \wedge \mathbf{c}) - P(G_{ob} \mid \mathbf{g} \wedge \mathbf{c})$$

$$\delta_{XOR}(G, \mathbf{c}) = P(G_{ob} \mid \mathbf{g1ob} \wedge \mathbf{c}) - P(G_{ob} \mid \mathbf{go} \wedge \mathbf{c})$$

For example, the degree of validity of the AND-decomposition assumption of the goal *NS* is as follows:

$$\delta_{AND}(NS, \mathbf{c}) = P(NS_{ob} \mid NSD_{ob} \wedge RS_{ob} \wedge \mathbf{c}) - P(NS_{ob} \mid \neg(NSD_{ob} \wedge RS_{ob}) \wedge \mathbf{c})$$

5) *Adoptability assumption*. Given two goal nodes G and G' such that G' is parent of G in $\mathcal{RBN}$, the degree of validity of the associated adoptability assumption in context $\mathbf{c}$ is

$$\delta_{AD}(G, G', \mathbf{c}) = P(G_{ob} \mid G'_{ob} \wedge \mathbf{c}) - P(G_{ob} \mid G'_{viol} \wedge \mathbf{c})$$

6) *Goal necessity assumption*. Given a goal node G and a set of softgoal nodes **S**, the degree of validity of the associated goal necessity assumption in context $\mathbf{c}$ is

$$\delta_{AC}(G, \mathbf{c}) = P(\mathbf{S}_{true} \mid G_{act} \wedge \mathbf{c}) - P(\mathbf{S}_{true} \mid G_{dis} \wedge \mathbf{c})$$

VI. FEASIBILITY

We report on a preliminary evaluation of the *feasibility* of our approach for validating goal model assumptions. A full evaluation that encompasses qualities like usability, scalability and generality, is left to future work.

We executed a traffic simulation of the running example by using a modified version of the CrowdNav simulator [10]. We extended CrowdNav in two ways: (i) besides the embedded adaptive navigation service, we have implemented a static

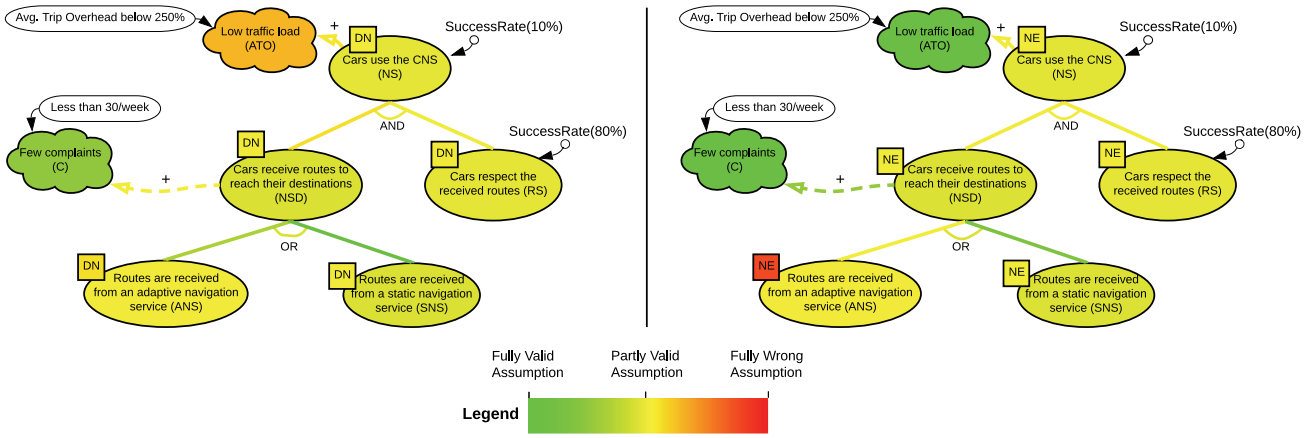

Fig. 3. A coloured visualization of the RM of Fig. 1 in two different operating contexts (DN and NE in the Figure).

navigation service; (ii) we have instrumented the simulator in such a way to monitor the satisfaction of the goals specified in Table I. We ran the simulator in all the operating contexts and we collected from the simulation logs a dataset composed of about 4.6 millions rows, a portion of which is reported in Table III. As a tool for handling the Bayesian Network we used the bnlearn R package [14]. We generated a *RBN* for our scenario using the mapping function *GM2BNS*; this led to the network shown in Fig. 2. Then, we trained such network using the dataset obtained from the traffic simulation. We evaluated then each of the 20 assumptions in the goal model of Fig. 1 in each of the the 4 different operating contexts. Table IV reports the degree of validity of the assumptions for two different operating contexts: *day-normal* (**dn**) and *night-extreme* (**ne**).

Fig. 3 reports a coloured version of the goal model of the running example, which shows the results of the model validation in the two considered operating contexts. Colours represent the degree of validity of the assumptions shown in Table IV: green represents valid assumptions (degree of validity close to 1), red denotes incorrect assumptions (degree of validity close to -1), and intermediate degrees of validity lead to colours in the gradient between green and red.[2]

The filling colour of goals and softgoals in Fig. 3 represents the degree of validity of the *goal satisfiability* and *softgoal achievement assumptions*, respectively. For example, the data that we analyzed show that the softgoal *ATO* is hardly achieved in context *day-normal* (left figure), i.e., the degree of validity of the softgoal achievement assumption $\delta_G(ATO, \textbf{dn})$ is below 0. This can be explained by the high number of cars driving in the city during the day. Note that this information can be useful to trigger a requirements evolution, since it shows that the currently defined requirements cannot guarantee the achievement of the softgoal.

The small squares on the top left of goals provide information

[2]The RGB colour for a degree of validity x is determined as follows: $(\lfloor(1-x)*255\rceil, 255, 0)$, if $x >= 0$, $(255, \lfloor(1+x)*255\rceil, 0)$ otherwise. $\lfloor x\rceil$ denotes the nearest integer to x.

TABLE IV
TWO TABLES REPORTING THE DEGREE OF VALIDITY OF THE ASSUMPTIONS MADE IN FIG.1 IN TWO DIFFERENT OPERATING CONTEXT (**DN** AND **NE**).

Assumption	Validity	Assumption	Validity
$\delta_S(NS, \textbf{dn})$	0.0580	$\delta_S(NS, \textbf{ne})$	0.0675
$\delta_S(NSD, \textbf{dn})$	0.1006	$\delta_S(NSD, \textbf{ne})$	0.0946
$\delta_S(ANS, \textbf{dn})$	1.71E-05	$\delta_S(ANS, \textbf{ne})$	-6.43E-05
$\delta_S(SNS, \textbf{dn})$	0.1002	$\delta_S(SNS, \textbf{ne})$	0.0940
$\delta_S(RS, \textbf{dn})$	0.0596	$\delta_S(RS, \textbf{ne})$	0.0581
$\delta_G(ATO, \textbf{dn})$	-0.2730	$\delta_G(ATO, \textbf{ne})$	0.9999
$\delta_G(C, \textbf{dn})$	0.5079	$\delta_G(C, \textbf{ne})$	0.9998
$\delta_C(ATO, NS, \textbf{dn})$	-0.0743	$\delta_C(ATO, NS, \textbf{ne})$	9.12E-06
$\delta_C(C, NSD, \textbf{dn})$	-0.0136	$\delta_C(C, NSD, \textbf{ne})$	0.4776
$\delta_{AD}(NS, NSD, \textbf{dn})$	-0.1007	$\delta_{AD}(NS, NSD, \textbf{ne})$	-0.0018
$\delta_{AD}(NS, RS, \textbf{dn})$	0.0025	$\delta_{AD}(NS, RS, \textbf{ne})$	0.0025
$\delta_{AD}(NSD, ANS, \textbf{dn})$	0.3541	$\delta_{AD}(NSD, ANS, \textbf{ne})$	-0.0252
$\delta_{AD}(NSD, SNS, \textbf{dn})$	1	$\delta_{AD}(NSD, SNS, \textbf{ne})$	0.6684
$\delta_{AND}(NS, \textbf{dn})$	0.0187	$\delta_{AND}(NS, \textbf{ne})$	-0.0376
$\delta_{XOR}(NSD, \textbf{dn})$	0.1002	$\delta_{XOR}(NSD, \textbf{ne})$	0.0939
$\delta_{AC}(NS, \textbf{dn})$	-0.0067	$\delta_{AC}(NS, \textbf{ne})$	-4.84E-05
$\delta_{AC}(NSD, \textbf{dn})$	-0.0040	$\delta_{AC}(NSD, \textbf{ne})$	0.0028
$\delta_{AC}(ANS, \textbf{dn})$	-0.1002	$\delta_{AC}(ANS, \textbf{ne})$	-0.7336
$\delta_{AC}(SNS, \textbf{dn})$	-0.0032	$\delta_{AC}(SNS, \textbf{ne})$	0.0024
$\delta_{AC}(RS, \textbf{dn})$	-0.0018	$\delta_{AC}(RS, \textbf{ne})$	-0.0014

about *goal necessity assumptions*. For example, the data show that the goal *ANS* is harmful in context *night-extreme* (right figure), i.e., the degree of validity of the goal necessity assumption $\delta_{AC}(ANS, \textbf{ne})$ is close to -1, which means that in context *night-extreme* the probability of achieving the softgoals is very low when the adaptive navigation service is employed, while is very high when it is not employed. This can be explained by the fact that the adaptive navigation service uses some of the cars as "explorers" to find less congested roads (see [10] for more details). However, such type algorithm results being harmful during the night since less cars drive in the city and roads are not that busy.

Finally, the decomposition links in the model are coloured based on the degree of validity of *adoptability assumptions*, the type of decomposition based on the validity of *decomposition assumptions* and contribution links based on the validity of

contribution assumptions.

VII. Related Work

Requirement monitoring is an essential task in order to evaluate the behavior of a system and to diagnose its problems. The availability of a requirements model during execution [15], [16] is crucial to support the evolution of the requirements. The majority of the existing approaches concerning monitoring and diagnosis (e.g., [5], [6]) focus on detecting misalignment between the system behavior and the expected behavior described by the requirements. Research on self-adaptive software leverages this knowledge to trigger an adaptation of the system that automatically restores the compliance with the requirements or that selects a more appropriate requirements variant (e.g., [17], [18]).

Works related to requirement assumptions validation are mainly proposed by Ali *et al.* [3], [9]. They show the advantages of monitoring requirements at runtime to detect when design-time assumptions about requirement satisfaction become invalid. In the same spirit Paucar *et al.* [4] proposed techniques to reassess the assumptions about the priority of softgoals.

The importance of valid assumptions underlying a system is also illustrated by Knobbout *et al.* [19]: under certain assumptions about the compliance of software components with the requirements, it is possible to prove whether some overall system properties can be achieved.

Bayesian Networks have been widely used in many fields (ranging from medicine to forensics) as knowledge representation structures for learning and reasoning about the inter-dependencies between their variables [12]. In RE they have been employed both for the runtime verification of requirements [20] and for decision making (e.g., DDNs can be used to revise the priorities of non-functional requirements [17]). We also present an application of Bayesian Networks to RE, but we focus on a wide range of design-time assumptions that are made by the builder of a goal model.

Wu *et al.* [21] propose a preliminary study of the relationship between an iStar [8] model and Bayesian Networks and a set of heuristics for mapping them. In our work we provide a formal and fully automatic mapping between a goal model and a Bayesian Network, we propose a different and more expressive type of Bayesian Network, integrating contextual information and the possibility of disabling nodes (supporting therefore also the use of the BN at runtime). Furthermore we use the BN to validate a goal model based on data, without relying on expert knowledge to determine the network parameters.

Finally, Reddivari *et al.* [22] show the importance of requirement visualization to improve the analytical capabilities of practitioners. In this work, we make a step toward this direction by visualizing the validity of the design-time assumptions on a requirements model, thereby helping practitioners take decisions on the evolution of a system.

VIII. Discussion and Future Work

We presented a novel approach to validate, by making use of empirical data, several assumptions underlying the structure of a requirement (here, *goal*) model. We illustrated how Bayesian Networks can be successfully employed to automatically determine the validity of design-time assumptions concerning a goal model. In particular we showed that the employment of a *Requirement Bayesian Network* allows to quantitatively determine the degree of validity of the assumptions by means of classical probabilistic inference.

By defining a formal mapping between a goal model and a *Requirement Bayesian Network* and by leveraging classical Bayesian learning and inference techniques, we provided an automated mechanism to validate design-time assumptions by exploiting empirical data (**RQ**).

We have shown how a simple labeling function can be used to visualize the degree of validity of the assumptions onto the original goal model, in order to support requirements engineers in understanding the impact of the data on the system goals.

Our technique can be used both at runtime (by monitoring the execution of the running system) to trigger or suggest an evolution of the requirements, and at design-time (by using already existing empirical data, if available) to help the designers build a requirements model that is more likely to stay valid when the system is put in operation.

Threats to Validity. A full evaluation of the proposal, in particular in terms of generality, scalability and usefulness, is left for future work. The lack of such evaluation affects both conclusion and external validity by threatening the applicability of the approach and its generality. The notion of degree of validity is based on the assumption that the collected positive and negative evidence have the same statistical significance. The choice of such method to evaluate the assumptions affects construct validity. In future work we plan to explore additional techniques that overcome this limitation. The topology of the *Requirement Bayesian Network*, reflecting the structure of a goal model, may influence the conclusions drawn via probabilistic inference. The choice of such topology is a threat to internal validity. Different mappings between a goal model may be tried to overcome this limitation.

Future work. A thorough evaluation of the scalability, usefulness and generality of our proposal is imperative. Moreover, we plan to develop algorithms that can guide software evolution by providing additional information on the most critical and significant assumptions. To do so, we plan to employ other analysis techniques for Bayesian Networks, such as sensitivity analysis [23] or qualitative reasoning [24]. Also, we are currently working on algorithms that, based on the information learned at runtime, automatically revise the requirements applied in different operating contexts. Finally, we plan to embed our techniques in a software tool—in the spirit of visual requirement analytics—that can be used by practitioners for monitoring their systems and for guiding their evolution.

References

[1] I. Sommerville, D. Cliff, R. Calinescu, J. Keen, T. Kelly, M. Z. Kwiatkowska, J. A. McDermid, and R. F. Paige, "Large-scale complex IT systems," *Communications of the ACM*, vol. 55, no. 7, pp. 71–77, 2012.

[2] J. Whittle, P. Sawyer, N. Bencomo, B. H. C. Cheng, and J. Bruel, "RELAX: a language to address uncertainty in self-adaptive systems requirement," *Requirements Engineering*, vol. 15, no. 2, pp. 177–196, 2010.

[3] R. Ali, F. Dalpiaz, P. Giorgini, and V. E. S. Souza, "Requirements evolution: From assumptions to reality," in *Proc. of EMMSAD*, 2011, pp. 372–382.

[4] L. H. G. Paucar, N. Bencomo, and K. K. F. Yuen, "Juggling preferences in a world of uncertainty," in *Proc. of RE*, 2017, pp. 430–435.

[5] Y. Wang, S. A. McIlraith, Y. Yu, and J. Mylopoulos, "Monitoring and diagnosing software requirements," *Automated Software Engineering*, vol. 16, no. 1, pp. 3–35, 2009.

[6] W. N. Robinson, "A requirements monitoring framework for enterprise systems," *Requirements Engineering*, vol. 11, no. 1, pp. 17–41, 2006.

[7] A. Van Lamsweerde, *Requirements engineering: From system goals to UML models to software*. Chichester, UK: John Wiley & Sons, 2009, vol. 10.

[8] F. Dalpiaz, X. Franch, and J. Horkoff, "iStar 2.0 language guide," arXiv:1605.07767 [cs.SE], 2016.

[9] R. Ali, F. Dalpiaz, and P. Giorgini, "Reasoning with contextual requirements: Detecting inconsistency and conflicts," *Information & Software Technology*, vol. 55, no. 1, pp. 35–57, 2013.

[10] S. Schmid, I. Gerostathopoulos, C. Prehofer, and T. Bures, "Self-adaptation based on big data analytics: A model problem and tool," in *Proc. of SEAMS*, 2017, pp. 102–108.

[11] V. E. S. Souza, A. Lapouchnian, W. N. Robinson, and J. Mylopoulos, "Awareness requirements for adaptive systems," in *Proc. of SEAMS*, 2011, pp. 60–69.

[12] S. J. Russell and P. Norvig, *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.

[13] D. J. Spiegelhalter, A. P. Dawid, S. L. Lauritzen, and R. G. Cowell, "Bayesian analysis in expert systems," *Statistical science*, pp. 219–247, 1993.

[14] M. Scutari, "Learning bayesian networks with the bnlearn r package," *arXiv preprint arXiv:0908.3817*, 2009.

[15] G. Blair, N. Bencomo, and R. B. France, "Models@ run. time," *Computer*, vol. 42, no. 10, 2009.

[16] N. Bencomo, J. Whittle, P. Sawyer, A. Finkelstein, and E. Letier, "Requirements reflection: requirements as runtime entities," in *Proc. of ICSE*, vol. 2, 2010, pp. 199–202.

[17] N. Bencomo, A. Belaggoun, and V. Issarny, "Dynamic decision networks for decision-making in self-adaptive systems: a case study," in *Proc. of SEAMS*, 2013, pp. 113–122.

[18] F. Dalpiaz, P. Giorgini, and J. Mylopoulos, "Adaptive socio-technical systems: a requirements-based approach," *Requirements Engineering*, vol. 18, no. 1, pp. 1–24, 2013.

[19] M. Knobbout and M. Dastani, "Reasoning under compliance assumptions in normative multiagent systems," in *Proc. of AAMAS*, 2012, pp. 331–340.

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

[21] H. Wu, L. Liu, and W. Ma, "Optimizing requirements elicitation with an i* and bayesian network integrated modelling approach," in *Proc. of COMPSAC*, 2010, pp. 182–188.

[22] S. Reddivari, S. Rad, T. Bhowmik, N. Cain, and N. Niu, "Visual requirements analytics: a framework and case study," *Requirements Engineering*, vol. 19, no. 3, pp. 257–279, 2014.

[23] L. van der Gaag, S. Renooij, and V. Coupé, "Sensitivity analysis of probabilistic networks," *Advances in probabilistic graphical models*, pp. 103–124, 2007.

[24] M. P. Wellman, "Fundamental concepts of qualitative probabilistic networks," *Artificial Intelligence*, vol. 44, no. 3, pp. 257–303, 1990.