Information and Software Technology 75 (2016) 105–121

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Run-time requirements verification for reconfigurable systems

George Chatzikonstantinou*, Kostas Kontogiannis

National Technical University of Athens, Department of Electrical and Computer Engineering, 9 Iroon Polytechniou, Athens 157 80, Greece

ARTICLE INFO

Article history:
Received 24 May 2015
Revised 6 March 2016
Accepted 10 April 2016
Available online 12 April 2016

Keywords:
Software engineering
System analysis
Run-time requirements verification
Goal models reasoning
Fuzzy reasoning

ABSTRACT

Context: Modern software systems often are distributed, run on virtualized platforms, implement complex tasks and operate on dynamically changing and unpredictable environments. Such systems need to be dynamically reconfigured or evolve in order to continue to meet their functional and non-functional requirements, as load and computation need to change. Such reconfiguration and/or evolution actions may cause other requirements to fail.

Objective: Given models that describe with a degree of confidence the requirements that should hold in a running software system, along with their inter-dependencies, our objective is to propose a framework that can process these models and estimate the degree requirements hold as the system is dynamically altered or adapted.

Method: We present an approach where requirements and their inter-dependencies are modeled using conditional goal models with weighted contributions. These models can be translated into fuzzy rules, and fuzzy reasoners can determine whether and to what degree, a requirement may be affected by a system change, or by actions related of other requirements.

Results: The proposed framework is evaluated for its performance and stability on goal models of varying size and complexity. The experimental results indicate that the approach is tractable even for large models and allows for dealing with models where contribution links are of varying importance or weight.

Conclusion: The use of conditional weighted goal models combined with fuzzy reasoners allowed for the tractable run-time evaluation of the degree by which system requirements are believed to hold, when such systems are dynamically altered or adapted. The approach aims to shed light towards the development of run-time requirements verification and validation techniques for adaptive systems or systems that undergo continuous, or frequent evolution.

1. Introduction

Over the past few years we experienced a significant growth on the deployment of highly interconnected systems operating in a number of domains such as banking, commerce, and government to name a few. These systems encompass complex requirements yielding thus large and complex models [1]. In addition to the complex requirements these systems entail, advances in virtualization technology make also possible the continuous evolution, dynamic provision, and dynamic adaptation of system resources, in a quest for these systems to constantly meet their numerous, and possibly diverse, functional and non-functional requirements. The problem has been recognized in the related research literature as an important one [2,3], both from the requirements verification perspective and, from the policy compliance perspective. Hence, it is important to develop techniques, and tools for assessing at run time whether changes in the system structure, and, operating environment violate requirements set forth by its stakeholders.

To date, the software engineering community has responded by investigating models, and frameworks allowing for the specification, analysis, and evaluation of system requirements. In this respect, research efforts have focused so far on the off-line static analysis and reasoning in such models for the purpose of evaluating the completeness, and validity of system requirements against stakeholder goals [4,5]. However, as business processes become more complex requiring highly inter-connected and inter-dependent services (e.g. Systems-of-Systems), issues related to the run-time verification of global system properties have emerged as very timely, important and challenging [6–8]. Work in the area

* Corresponding author. Tel.: +302107722511; Fax: +30 210 772 2511.
E-mail addresses: gchatzik@cslab.ece.ntua.gr (G. Chatzikonstantinou), kkontog@softlab.ntua.gr (K. Kontogiannis).

http://dx.doi.org/10.1016/j.infsof.2016.04.005
0950-5849/

of adaptive systems is attempting to address some of these issues, and research efforts aiming to model and analyze the run-time behavior of such systems have started emerging in the related literature [9,10]. However, there is still limited work on dealing with large requirements models, especially when the dependencies and impact of one design decision or system requirement cannot be fully and deterministically modeled due to the high complexity of structural and behavioral inter-dependencies present in such complex systems. It is therefore important to investigate analysis techniques that are tractable so that can be applied at run-time, and allow for reasoning on large requirements models, especially in the presence of incomplete knowledge related to the impact changes in one system or component may have on the behavior and requirements of other interconnected systems or components.

For this paper, we adopt a goal-driven view of this run-time verification problem aiming to assess whether a system that operates in a specific context is compliant to a set of requirements. This issue entails a number of challenges that have to be considered and addressed. First, an appropriate modeling notation for denoting goals that should hold in the running system under various contexts must be used. Such a modeling notation should be expressive enough to capture dependencies that exist between goals, allow for the definition of multiple variations of the model under different contexts, and allow for modeling the way design decisions and system properties affect these goals. Such a notation must also have well defined semantics so as to enable the use of tools and algorithms that can automate the processing of the models. Second, as dependencies in a complex system may not be fully known, or properly recorded, a fuzzy approach should be used for specifying such dependencies within an acceptable degree of ambiguity. Third, a reasoning framework is required to evaluate at run-time how and to what degree system changes affect the requirements taking into account events collected from the running system.

In a nutshell, the proposed framework allows for a) system goals and their variations under multiple contexts, hereinafter referred to as *views*, to be modeled as conditional goal trees [11], whose nodes hold with a degree of truth; b) dependencies between goals to be modeled as weighted contributions [12,13], where weights are interpreted as the degree by which a stakeholder's belief for the target node satisfaction/dissatisfaction is *increased* given the satisfaction status of the source node; c) transformations to be utilized in order to map conditional goal trees to weighted fuzzy rules [14] and; d) fuzzy reasoning to be applied in order to tractably verify whether, and to what degree, specific system goals hold at run-time.

The paper is organized as follows. Section 2 formally defines the requirements-based view of the verification problem. Section 3 summarizes key concepts in the areas of Goal Models and weighted fuzzy rules. Section 4 briefly describes the details of our approach, and Section 5 presents a running example used throughout the paper. Section 6 introduces the semantics of the notation utilized to model system goals and views, and Section 7 describes the transformation of goal models to rules that can be used by the proposed inference engine. Subsequently, a lab experiment is presented in Section 8, and the performance of the proposed framework is evaluated against randomly generated models of varying size and complexity in Section 9. Finally, Section 10 presents related work and Section 11 concludes the paper.

2. Formal definition of the problem

Run-time verification is the process of evaluating, while the system operates, whether it meets certain expected behavior and goals [15,16]. In this paper we adopt a requirements-based view of the run-time verification problem that we refer to as the *ReqRV* problem. To formally define it, we use as a starting point the seminal study by Zave and Jackson on software requirements [17].

According to this study, given a set of requirements R, and a set of domain assumptions D, the requirements satisfaction problem aims to determine the specifications S that can ensure the fulfilment of requirements R. The above can be summarized by the following equation as originally presented in [17]:

$$D \cup S \vdash R \tag{1}$$

In a similar manner as in [18], Eq. (1) can be adapted to the ReqRV problem, where now the problem can be formulated as follows. Given a set of domain assumptions D, and the description of the system in terms of a set S of observable characteristics of the system (e.g. logged events), deduce the values of the requirements in R (i.e. get the values that are logically implied by D and S for R), hereinafter referred to as *system goals*. However, in a software system, goals that hold at run-time and the dependencies that exist between these goals, may vary depending on the context C the system operates in, hence the problem can be formally defined as:

$$D(C) \cup S \models R(C) \tag{2}$$

where $D(C)$ are the rules that describe the knowledge related to those goals in context C, S is the values of the observable characteristics that reflect the state of the running system, and $R(C)$ the satisfaction values (i.e. true if goal is satisfied, false otherwise) for all system goals in context C. Note that in Eq. (2) S does not depend on context C, as we consider that the observable characteristics collected via the monitoring infrastructure of the system do not depend on context changes, and the same data are collected no matter what the context is. Subsequently, having deduced the values for all system goals from Eq. (2), we can check whether they are satisfied, i.e. their value in $R(C)$ is true.

However, in our analysis, we consider a *fuzzy* approach towards system goals' satisfaction, i.e. a system goal is not either satisfied (true) or denied (false), but rather it may be satisfied to a certain degree in the interval [0,1] expressed as a percentage, e.g. a goal may be 80% satisfied. Additionally, rules in $D(C)$ are annotated with a weight in the interval (0,1] which denote the subjective degree of belief a domain expert has in a rule. In this context, by applying Eq. (2), we get for each system goal in $R(C)$, a truth value in the interval [0,1] denoting its satisfaction degree in the running system. We can now check whether the system complies with the predefined set of requirements, by checking whether either the inferred degrees are greater that an specific threshold or if the satisfaction degrees are within an interval of acceptable values.

3. Background

3.1. AND/OR goal trees

AND/OR goal trees are a modeling formalism extensively utilized in requirements engineering, where its basic concept is the top-down AND/OR decomposition of goals into sub-goals. An AND-decomposed goal can be satisfied if all of its sub-goals are true, while an OR-decomposed goal is fulfilled if at least one of its sub-goals holds.

Additionally, two goals may be connected by a contribution link. More specifically, a goal may potentially contribute to other goals in four ways, namely, S^P, S^N, D^P and D^N. In this paper we adopt the semantics stated in [19] for those four contribution types, and annotate each contribution from a source node g_s to a target node g_t with a number called weight (w) like in [12,13],

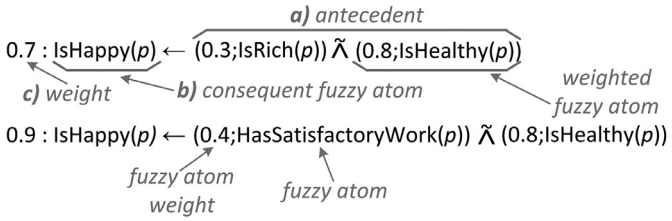


Fig. 1. Examples of wf-rules.

interpreted as follows:

$$\begin{aligned} S^P/D^P &: \text{ a stakeholder's belief for } g_t \text{ satisfaction/denial is } \textit{increased} \text{ by } w \text{ if } g_s \text{ is satisfied/denied} \\ S^N/D^N &: \text{ a stakeholder's belief for } g_t \text{ denial/satisfaction is } \textit{increased} \text{ by } w \text{ if } g_s \text{ is satisfied/denied} \end{aligned} \tag{3}$$

Finally, goals can be classified into *hard goals* and *soft goals* [20]. Hard goals denote goals for which there are clear cut criteria to define their truth value, while soft goals are goals that are satisfied when there is sufficient positive and little negative evidence for their satisfaction. We adopt here the terminology introduced in [21], where hard goals referred to as *crisp goals* and soft goals as *fuzzy goals*. In the rest of this paper, when we refer to goals we mean both crisp and fuzzy ones, unless otherwise stated.

3.2. Weighted fuzzy rules

Fuzzy logic is a many-valued logic in which variables can hold with a degree of truth in the range of [0, 1], as opposed to boolean logic where a variable can be either 0 (false) or 1 (true). Building on fuzzy logic theory, Chortaras et al. [14] proposed a rule base language that can be used to carry out inferences on imprecise or fuzzy knowledge bases.

In this context, a fuzzy knowledge base is defined as a set of *weighted fuzzy rules (wf-rules)*. Two examples of wf-rules that could be used to reason for the degree of happiness of a person p are listed in Fig. 1. Each wf-rule is composed of *fuzzy atoms*, i.e. predicates like IsHappy that reflect properties for individuals, and consists of:

a) the *antecedent* given as a fuzzy conjunction of *weighted fuzzy atoms*, where the weight of each fuzzy atom models its relative significance among other fuzzy atoms in the antecedent;
b) the *consequent fuzzy atom*, e.g. IsHappy in both rules; and
c) a *weight* in the range of [0, 1] that denotes the relative importance of the rule among other rules with the same consequent.

Additionally, as described in [14, p. 97] we can explicitly assign a degree of truth to fuzzy atoms as:

$$0.4 : \mathit{IsRich}(\mathit{John}) \leftarrow (1.0; t),$$

where 0.4 is the truth value of the *IsRich* fuzzy atom for *John* and t is a pre-defined fuzzy atom used to denote the absolute truth. We refer to this type of rules as *fuzzy facts*.

Given a set of wf-rules and a set of fuzzy facts, the reasoner introduced in [14] deducts the truth values for all consequent fuzzy atoms. This is performed by using an appropriate s-norm (i.e. fuzzy OR operator) to combine rules with the same consequent, and a weighted t-norm (i.e. weighted fuzzy AND operator) to combine the weighted fuzzy atoms appear in the antecedent of a wf-rule. In this paper, we apply reasoning utilizing the probabilistic sum s-norm defined as:

$$\perp_{sum}(a, b) = a + b - a \cdot b \tag{4}$$

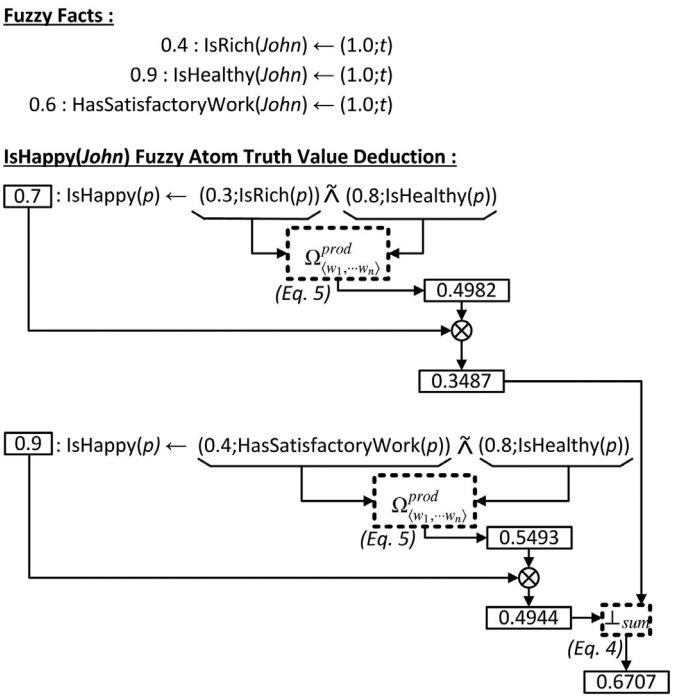


Fig. 2. Reasoning example.

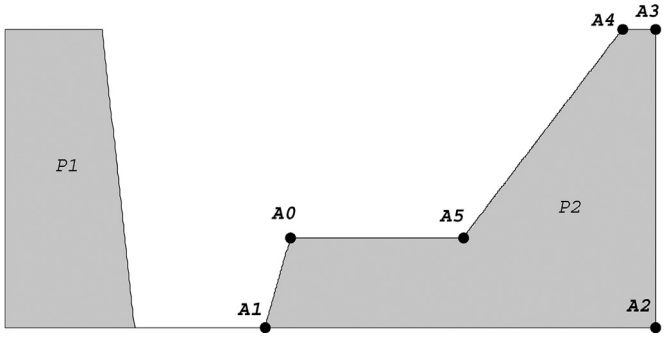


Fig. 3. Two non-self-intersecting closed polygons.

and the generalized weighted fuzzy conjunction operator defined in [14] :

$$\Omega^{prod}_{\langle w_1,\cdots w_n\rangle} = \max\left\{0, \bar{w} - 1 + \prod_{i=1}^{n} a_i^{w_i}\right\} \tag{5}$$

where a_i are the truth values of the fuzzy atoms in the antecedent, w_i are the corresponding weights, and $\bar{w} = max_{i=1,\cdots n}(w_i)$. The way the above operators are used to extract the truth value of *IsHappy(John)* for a given set of fuzzy facts is depicted in Fig. 2. Finally, when all the weights are equal to 1, the operator of Eq. (5) is the classical product t-norm defined as:

$$\top_{prod}(a, b) = a \cdot b \tag{6}$$

3.3. Polygon area centroid calculation

We outline here the calculation of the "center of gravity" (or centroid) x-coordinate [22] for areas that can be divided into a finite number of non-self-intersecting closed polygons, like the area depicted in Fig. 3. Centroid x-coordinate calculation is part of the defuzzification process that is described in Section 7.4, however we summarize the required calculation steps for clarity and background purposes.

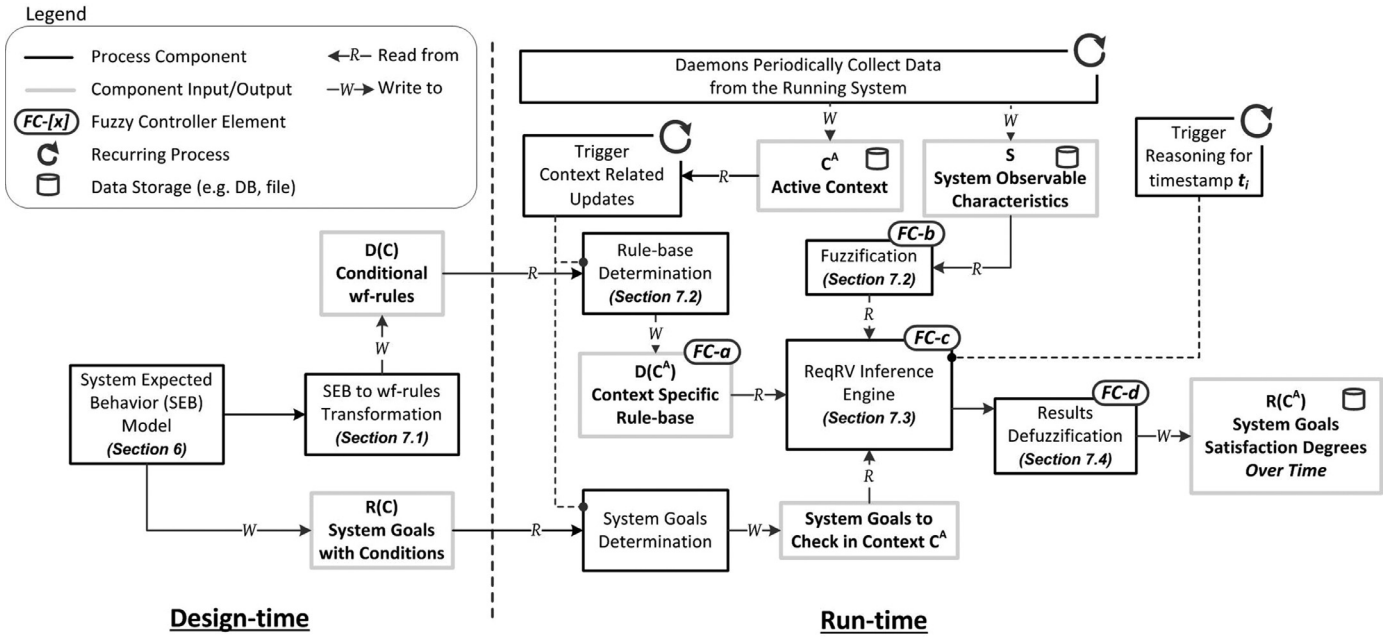


Fig. 4. Outline of the proposed process for the ReqRV problem.

Given a polygon P with m vertices, we number the vertices in counterclockwise order of their occurrence along the polygon's perimeter, e.g. vertices $A0$ to $A5$ of $P2$ polygon in Fig. 3. Subsequently, we use the sequence $(x_0, y_0), \ldots, (x_{m-1}, y_{m-1})$ of the m vertices to calculate the area of P as follows:

$$E(P) = \frac{1}{2}\sum_{k=0}^{m-1}(x_k y_{k+1} - x_{k+1} y_k) \tag{7}$$

with $(x_m, y_m) = (x_0, y_0)$, as P is a closed polygon (i.e. last and first vertices coincide).

The centroid x-coordinate for polygon P can then be calculated using the formula:

$$C_x(P) = \frac{1}{6E(P)}\sum_{k=0}^{m-1}(x_k + x_{k+1})(x_k y_{k+1} - x_{k+1} y_k) \tag{8}$$

Finally, the combined centroid x-coordinate of an area consisting of n non-self-intersecting closed polygons $P_1, \ldots, P_n$, is given by the formula:

$$C_x = \frac{\sum_{i=1}^{n} C_x(P_i)\cdot E(P_i)}{\sum_{i=1}^{n} E(P_i)} \tag{9}$$

where $E(P_i)$ and $C_x(P_i)$ are calculated using Eqs. (7) and (8), respectively.

4. Process outline

We address the ReqRV problem by introducing a framework which is based on the principles of fuzzy controllers. A fuzzy controller is composed of four elements [23]: *FC-a) a rule-base*, i.e. rules describing the experts' knowledge of the domain; *FC-b) a fuzzification process* for the transformation of the input in a processable form; *FC-c) an inference mechanism* which combines the rule-base with the input to deduce membership degrees for a set of output variables; and *FC-d) a defuzzification process* which converts the inference outcome (i.e. the membership degrees) for each output variable into a quantifiable result by combining the membership degrees.

The main components of the framework are classified to Design-time and Run-time components according to the phase being utilized, and are depicted in Fig. 4 along with the mappings that exist between the sets of Eq. (2), and the aforementioned four elements of fuzzy controllers.

Design-time components : initially, stakeholders define the set of required goals for the system, along with the dependencies that exist between them. This is achieved by using an appropriate visual notation, whose semantics are described in Section 6.

The resulting final model ("System Expected Behavior Model") is then transformed in a set of *conditional* wf-rules through an appropriate transformation process, i.e. "SEB to wf-rules Transformation" component in Fig. 4, described in Section 7.1. The generated conditional wf-rules correspond to set $D(C)$ of Eq. (2), and are called conditional as each rule is accompanied by a CNF formula which determines whether the rule should be used during the reasoning process. Also, we extract the full set of system goals for which the framework should calculate the satisfaction degrees under various contexts. This set of system goals corresponds to set $R(C)$ of Eq. (2). Each goal in the set is accompanied by a condition which defines whether the goal should be checked in the given context or not.

Run-time components: at run-time, the system is monitored and periodically data that reflect the state of the running system ("System Observable Characteristics"), and the context in which the system operates ("Active Context") are collected. The extraction and collection of these values can be performed by utilizing a log analysis technique like the ones presented in [24,25] and is beyond the scope of this paper.

Additionally, either periodically or when a certain percentage of the data that reflect the active context changes, two processes are triggered in order to perform context related updates ("Trigger Context Related Updates"). The first process ("Rule Base Determination") evaluates the CNF formulas, and defines the subset of wf-rules that should be utilized during the reasoning process ("Context Specific Rule-base"). This process requires linear time in the total number of CNF formulas, and the produced subset of wf-rules constitutes the rule-base for the ReqRV reasoner. The second process ("System Goals Determination") defines the set of system

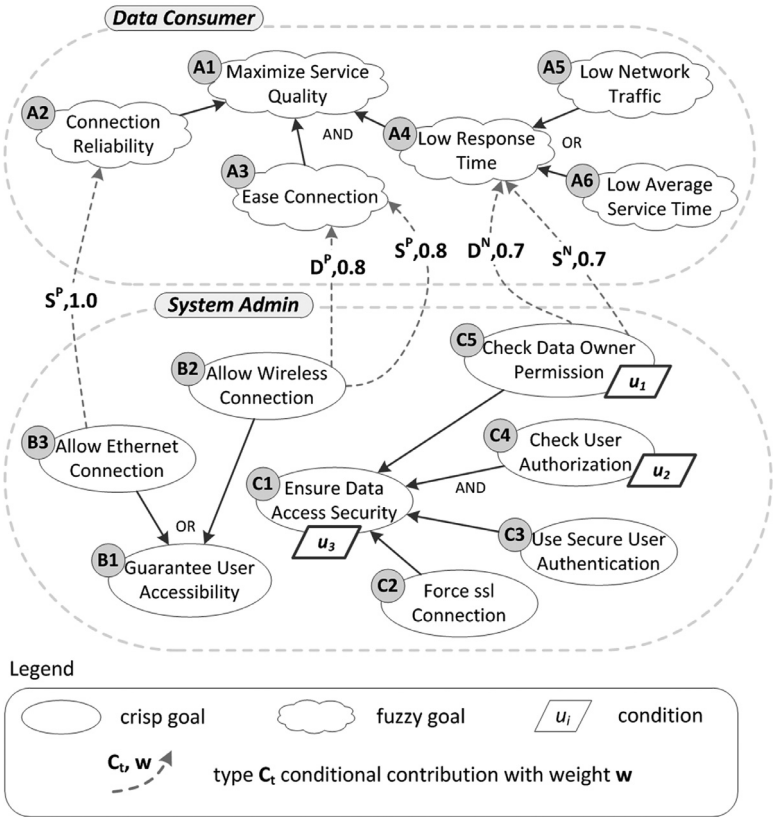


Fig. 5. The running example.

goals that should be checked in the given active context, and hence the goals for which the reasoner should calculate their values. This process also requires linear time in the total number of conditional goals in the initial model.

Finally, at regular time intervals the reasoning process is triggered ("Trigger Reasoning for timestamp t_i"), allowing thus the end user to continuously assess system's compliance against the expected behavior. Initially, the "System Observable Characteristics" i.e. set S of Eq. (2), are transformed to fuzzy facts through an appropriate "Fuzzification" process. The fuzzy facts and the rule-base constitute the knowledge base that subsequently is used by the "ReqRV Inference Engine" to deduce membership degrees for all system goals. A proper defuzzification process ("Results Defuzzification") can then be utilized to calculate for each timestamp t_i, and for each system goal in $R(C^A)$ its satisfaction degree.

5. Running example

To better illustrate the proposed process for the ReqRV problem, we employ a running example which is partially based on examples used in [26–28]. The example is illustrated in Fig. 5, and describes a fraction of system goals of a simplified, yet realistic, data records management software.

For this example, we consider the goals stemming from two stakeholders, namely "Data Consumer" and "System Admin". The former may correspond to an assistant trying to retrieve patients' records and personal data from a patient management system, who requests high quality of service. This is captured by the fuzzy goal "Maximize Service Quality", which is further AND-decomposed to the fuzzy sub-goals "Connection Reliability", "Ease Connection" and "Low Response Time".

Similarly, the "System Admin" stakeholder aims to "Ensure Data Access Security" and to "Guarantee User Accessibility". Both of these goals are crisp and can be decomposed into simpler sub-goals.

Additionally, the goal model in Fig. 5 contains the three conditional goals "Check Data Owner Permission", "Check User Authorization" and "Ensure Data Access Security" that are associated with conditions u_1, u_2 and u_3 respectively. u_1 corresponds to a security policy that enforces a schema where data owners should give their permission for other users to access the data. This policy should apply for example to a medical records system, where patients should be able to restrict access to specific details of their medical record, and could formally defined using an OCL2.0 derivation rule [29] of the form:

```
context ContextHelper::u1Holds()
derive:security.dataPermRequired() = true
```

Furthermore, condition u_2 specifies whether the system permits certain actions to specific users (u_2 is true) or all system users have the same rights in the system (u_2 is false). Conditions attached to goal model elements can be used to differentiate the analysis. For example, "Ensure Data Access Security" crisp goal exists in the model depending on the truth value of u_3, and is decomposed to a different set of sub-goals depending on the truth values of u_1 and u_2 (e.g. set {$C2$, $C3$, $C4$, $C5$} if both conditions are true and set {$C2$, $C3$} if both conditions are false).

Moreover, contribution links are used to depict dependencies that exist between goals. For example, the fact that users can connect to the system through a wireless connection, i.e. "Allow Wireless Connection" is true, contributes positively to fuzzy goal "Ease Connection" with a degree equal to 0.8. In the following sections we are going to refer to the nodes of the model using the codes assigned to each one of them. e.g. $A2$ corresponds to "Connection Reliability" node.

Table 1
Constraints for gm-rules.

Rule type	Constraints
AND, OR	***(1a)*** $w = 1$ ***(1b)*** **if** g_t is fuzzy/crisp **then** all nodes in G_s must be fuzzy/crisp
S^P, D^P, S^N, D^N	***(2a)*** $\lvert G_s \rvert = 1$ ***(2b)*** **if** g_t is crisp **then** the source node must be crisp and $w = 1$

Finally, note that while in the example of Fig. 5 the two stakeholders have disjoint sets of goals, there is no restriction to the goals stakeholders can define. Contrariwise, different stakeholders can define similar goals with different decompositions, applying thus an analysis that better fits their needs.

6. Modeling stakeholder goals and views

To model the expected behavior of a system (i.e. system goals) under various contexts we utilize conditional goal models (an example is depicted in Fig. 5). What follows is a formal definition of these models in order to ease the formulation of their transformation to wf-rules. More specifically:

Definition 1. A Goal Model Rule (gm-rule) is a tuple of the form $\langle R_t, g_t, G_s, w\rangle$, where $R_t \in \{\text{AND, OR, } S^P, D^P, S^N, D^N\}$ is the type of the rule, g_t is the target goal of the rule, G_s is a non-empty set of source goals (crisp or fuzzy ones) that does not contain g_t, and $w \in (0, 1]$ is the weight of the rule.

For example the AND-decomposition of fuzzy goal A_1 to nodes A_2, A_3, and A_4 in Fig. 5 can be formally written as:

$$r_{A1} = \langle \text{AND}, A1, \{A2, A3, A4\}, 1.0\rangle$$

Depending on the type of a gm-rule, the constraints summarized in Table 1 apply, and ensure that:

(1a) decomposition rules always have weight equal to 1,
(1b) goals can only be decomposed to goals of the same type,
(2a) contribution rules have only one source node denoted as g_s, and
(2b) only crisp nodes can contribute to crisp nodes and for those contribution rules the weight is always equal to 1.

Regarding constraint (1b), while allowing fuzzy goals to be decomposed to crisp ones will not differentiate the analysis as this is described hereinafter, we consider that it is more "natural" to model dependencies between fuzzy and crisp goals using decomposition rules.

Goals and gm-rules may be conditional. To support conditional elements, we define the set $\mathcal{U}$ of all possible conditions that can be linked to rules or goals of the model, where each condition is a variable that can be either true or false and can add elements to or remove elements from the model. For example, for the model in Fig. 5 set $\mathcal{U}$ is defined as $\mathcal{U} = \{u_1, u_2, u_3\}$, with condition's u_1 truth value determining the existence or absence of node $C5$ to the model. In contrast, elements that are always part of the model (e.g. node $A1$) are called *unconditional*, and are linked to the pre-defined condition $\top$ which is always true.

Taking the above into consideration we formulate the following definition for a conditional goal model that describes the expected behavior of the running system under various contexts:

Definition 2. A System Expected Behavior (SEB) model is a tuple of the form $\langle G, D, \mathcal{U}, Cond\rangle$, where G is a non-empty set of goals (crisp or fuzzy ones), D is the set of gm-rules that describe the relations that exist between goals in G, $\mathcal{U}$ is a set of conditions that can be linked to either goals or rm-rules, and $\text{Cond} : G \cup D \mapsto Pow(\mathcal{U} \cup \{\top\})$, is a function that returns the set of conditions that are linked to each goal or gm-rule of the SEB.

More specifically, we define function Cond(a) for a goal or gm-rule a, to return the conditions $U^a \subseteq \mathcal{U} \cup \{\top\}$ for which a exists in the model, i.e $U^a = \text{Cond}(a)$. Hence, a exists in the model only if at least one of the conditions in U^a is true. We also extend the *Cond* function for sets of model elements as follows; given a set $A = \{a_1, \cdots a_n\}$ of goals or gm-rules $Cond(A) = \bigcup_{i=1}^{n} Cond(a_i)$.

For example, the SEB depicted in 5, can be formally defined as $\langle G, D, \mathcal{U}, \text{Cond}\rangle$, where:

$$G = \{A1, \cdots, A6, B1, B2, B3, C1, \cdots, C5\}, \mathcal{U} = \{u_1, u_2, u_3\}$$

$$D = \begin{cases} r_{A1} = \langle \text{AND}, A1, \{A2, A3, A4\}, 1\rangle, \\ r_{A4} = \langle \text{OR}, A4, \{A5, A6\}, 1\rangle, \\ r_{B1} = \langle \text{OR}, B1, \{B2, B3\}, 1\rangle, \\ r_{C1} = \langle \text{AND}, C1, \{C2, C3, C4, C5\}, 1\rangle, \\ r_1 = \langle S^N, A4, \{C5\}, 0.7\rangle, r_2 = \langle D^N, A4, \{C5\}, 0.7\rangle, \\ r_3 = \langle S^P, A3, \{B2\}, 0.8\rangle, r_4 = \langle D^P, A3, \{B2\}, 0.8\rangle, \\ r_5 = \langle S^P, A2, \{B3\}, 1.0\rangle \end{cases}$$

$$Cond_{SA}(a) = \begin{cases} \{u_1\}, & \text{if } a = C5 \\ \{u_2\}, & \text{if } a = C4 \\ \{u_3\}, & \text{if } a = C1 \\ \{\top\}, & \text{otherwise} \end{cases}$$

By assigning truth values to the condition variables in $\mathcal{U}$, we produce multiple *views* for the same SEB, where each view contains a different set of goals and gm-rules. Hence, by defining a set $C^A \subseteq \mathcal{U} \cup \{\top\}$ which contains only the conditions that are true, hereinafter referred to as *active context*, we can produce a specific view of an SEB. By definition, the active context contains at least the variable $\top$, as it represents a condition that is always true. Given an active context C^A and an SEB $m = \langle G, D, \mathcal{U}, Cond\rangle$, we say that we produce the view of m restricted to C^A, denoted as $m \restriction_{C^A}$ which is a goal model that contains:

1. All nodes $g \in G$ for which $Cond(g) \cap C^A \neq \emptyset$, and
2. All rules $r = \langle R_t, g_t, G_s, w\rangle \in D$ for which:
   (a) $Cond(r) \cap C^A \neq \emptyset$, and
   (b) $Cond(g_t) \cap C^A \neq \emptyset$, and
   (c) $Cond(G_s) \cap C^A \neq \emptyset$

So, $m \restriction_{C^A}$ is a model that contains all goals related to the conditions that hold (i.e. the active context), and additionally the gm-rules related to conditions that hold (2a in the above list) and for which the target node (2b) and at least one source node (2c) exist in the model.

For example, for the model depicted in Fig. 5, the view for the active context $\{u_2, \top\}$ will be a model that will not contain the conditional goal $C5$, as:

$$Cond(C5) \cap \{u_2, \top\} = \{u_1\} \cap \{u_2, \top\} = \emptyset$$

Furthermore, it will not contain the S^N and D^N contribution links stemming from $C5$, as their source node, i.e. node $C5$, will not appear in the model.

7. Reasoning on system goals

SEB models presented above provide the theoretical foundation for the ReqRV problem, and enable system goals to be expressed under various contexts. However, for reasoning to take place, the following actions must be performed (also see Section 4) :

1. *conditional wf-rules generation*,

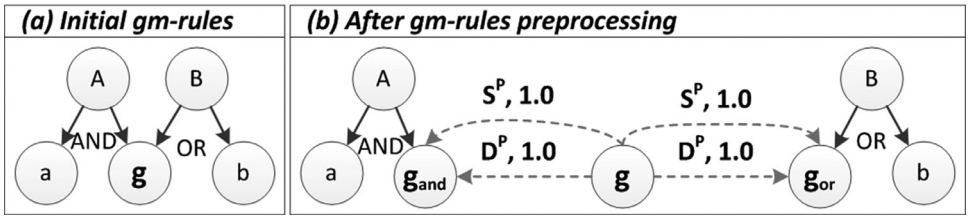


Fig. 6. Addition of pseudonodes and contribution links.

2. *rule-base determination* and *fuzzification* of system observable characteristics' values extracted via monitoring techniques,
3. *inference*, i.e reasoning for all system goals using an appropriate inference mechanism,
4. *defuzzification* of the calculated values to extract the satisfaction degrees for all system goals.

The first of these steps can be completed at design time, while the rest of them are steps executed at regular intervals as the system operates as this is described in Section 4. In the rest of this section we describe in detail the aforementioned four actions.

7.1. Conditional weighted fuzzy rules generation

To generate the required wf-rules, we must first define the set of fuzzy atoms that will be used in the rules. More specifically, we utilize two fuzzy atoms, namely *LowSat*(g) and *HighSat*(g), which are fuzzy predicates that correspond to whether goal g is "poorly" or "highly satisfied" expressed by a truth value in the interval [0, 1]. Details about these two predicates are given in Section 7.2.

Given a SEB model in the form $\langle G, D, \mathcal{U}, Cond\rangle$, we use sets G and D to generate a list of conditional wf-rules, a process that completes in two phases. During the first phase ("gm-rules preprocessing" phase) a number of pseudo-nodes are added to the model where necessary. Subsequently, during the second phase ("conditional wf-rules generation" phase), the model of the previous phase is processed in order to generate the final set of conditional wf-rules.

Phase 1 [gm-rules preprocessing]. The first phase aims at substituting goal nodes that participate as child nodes to *both* AND and OR decomposition rules (like g in Fig. 6) with an OR-psudonode (g_{or}) and an AND-pseudonode (g_{and}). The necessity of these substitutions will be justified in the second phase of the algorithm.

Having replaced g with g_{or} and g_{and} we have to ensure that the same rules that were used to deduct the truth value of g, will now be used to deduct the truth value of the newly added pseudonodes. To ensure that, we add 4 contribution links which practically assign the truth value of g, as this has been calculated from the existing rules, to its pseudonodes. More specifically, the two S^P contributions ensure g_{or} and g_{and} satisfaction when g is satisfied, while the two D^P contributions, ensure g_{or} and g_{and} denial when g is denied (see Eq. (3)). To sum up, through this phase an initial SEB model m with a set D of gm-rules, and a set G of nodes, is transformed to a SEB model m' with a set D' that contains all rules in R plus the additional contributions, and a set G' that contains all nodes in G plus the pseudonodes. Note that the produced model m' is only intended to be used for the generation of the wf-rules in the second phase, and not to replace the initial model.

Phase 2 [conditional wf-rules generation]. The model m' produced in the previous phase can now be used to generate the required set of wf-rules as this is described in Algorithm 1. As some of the goals and gm-rules are conditional, we should extract a boolean formula that determines whether this gm-rule should be present in the model, and hence, whether the wf-rules generated from this gm-rule should be used in the analysis given an active context C^A.

Algorithm 1 Conditional wf-rules generation

Input : G': goals, D': gm-rules
Output : CondRB : cond. wf-rules

1: **for all** gm_ r $= \langle R_t, g_t, G_s, w\rangle \in D'$ **do**
2: $f \leftarrow$ CNF(Cond(gm_ r), Cond(g_t), Cond(G_s))
3: WF $\leftarrow$ produceWFRules(gm_ r)
4: **for all** $wf_r \in$ WF **do**
5: add $\langle f, wf_r\rangle$ in CondRB
6: **end for**
7: **end for**
8: **for all** $g \in G'$ **do**
9: **if** $Cond(g) - \{\top\} \neq \emptyset$ **then**
10: **if** isAndChild(g) **then**
11: $f \leftarrow \neg$CNF($Cond(g)$)
12: add $\langle f, 1.0 : \text{HighSat}(g) \leftarrow (1.0; t)\rangle$ in CondRB
13: add $\langle f, 0.0 : \text{LowSat}(g) \leftarrow (1.0; t)\rangle$ in CondRB
14: **end if**
15: **if** isOrChild(g) **then**
16: $f \leftarrow \neg$CNF($Cond(g)$)
17: add $\langle f, 0.0 : \text{HighSat}(g) \leftarrow (1.0; t)\rangle$ in CondRB
18: add $\langle f, 1.0 : \text{LowSat}(g) \leftarrow (1.0; t)\rangle$ in CondRB
19: **end if**
20: **end if**
21: **end for**
22: **return** CondRB

For every gm-rule $gm\_\ r = \langle R_t, g_t, G_s, w\rangle$ in the set returned from the first phase (line 1) we create a CNF expression (line 2) defined as:

$$\textbf{CNF}(\text{Cond}(gm\_r),\ \text{Cond}(g_t),\ \text{Cond}(G_s)) \tag{10}$$

which corresponds to a *conjunction* of 3 clauses, one for each set of conditions, where each clause is a *disjunction* of the condition variables in the corresponding set (i.e. condition variables separated by ORs). This expression describes the fact that for a gm-rule to exist at least one source node and the target node should be present in the model, and the rule itself should be related to a condition that belongs to the active context. For example for the AND-decomposed goal $C1$ in Fig. 5, Eq. (10) takes the form:

$$\textbf{CNF}(\{\top\}, \{u_3\}, \{u_1, u_2, \top\}) = \top \wedge u_3 \wedge (u_1 \vee u_2 \vee \top) = u_3$$

which means that when u_3 is false, the AND-decomposition rule will not be part of the model.

Subsequently, a set of wf-rules is generated for every gm-rule in the model (line 3), using the transformation rules summarized in Table 2 (see Section 7.2 for details on LowSat and HighSat). For example, for the AND-decomposition $r_{C1} = \langle \text{AND}, C1, \{C2, C3, C4, C5\}, 1\rangle$, of the example in Fig. 5, the

Table 2
gm-rules to wf-rules transformation.

gm-rule	Interpretation	wf-rules
$\langle$AND, g, $\{g_1, \cdots g_n\}$, 1$\rangle$	g is satisfied if all $g_1 \cdots g_n$ are satisfied	$1.0 : \text{HighSat}(g) \leftarrow (1.0; \text{HighSat}(g_1)) \tilde{\wedge} \cdots \tilde{\wedge} (1.0; \text{HighSat}(g_n))$ $1.0: \text{LowSat}(g) \leftarrow (1.0; \text{LowSat}(g_1))$ $\vdots$ $1.0: \text{LowSat}(g) \leftarrow (1.0; \text{LowSat}(g_n))$
$\langle$OR, g, $\{g_1, \cdots g_n\}$, 1$\rangle$	g is satisfied if at least on of $g_1 \cdots g_n$ is satisfied	$1.0 : \text{LowSat}(g) \leftarrow (1.0; \text{LowSat}(g_1)) \tilde{\wedge} \cdots \tilde{\wedge} (1.0; \text{LowSat}(g_n))$ $1.0: \text{HighSat}(g) \leftarrow (1.0; \text{HighSat}(g_1))$ $\vdots$ $1.0: \text{HighSat}(g) \leftarrow (1.0; \text{HighSat}(g_n))$
$\langle S^P, g_t, \{g_s\}, w\rangle$	g_s satisfaction implies g_t satisfaction with degree w	$1.0: \text{HighSat}(g_t) \leftarrow (w; \text{HighSat}(g_s))$
$\langle S^N, g_t, \{g_s\}, w\rangle$	g_s satisfaction implies g_t denial with degree w	$1.0: \text{LowSat}(g_t) \leftarrow (w; \text{HighSat}(g_s))$
$\langle D^N, g_t, \{g_s\}, w\rangle$	g_s denial implies g_t satisfaction with degree w	$1.0: \text{HighSat}(g_t) \leftarrow (w; \text{LowSat}(g_s))$
$\langle D^P, g_t, \{g_s\}, w\rangle$	g_s denial implies g_t denial with degree w	$1.0: \text{LowSat}(g_t) \leftarrow (w; \text{LowSat}(g_s))$

following wf-rules will be generated:

$$1.0 : \text{HighSat}(C1) \leftarrow (1.0; \text{HighSat}(C2)) \tilde{\wedge} (1.0; \text{HighSat}(C3)) \tilde{\wedge} (1.0; \text{HighSat}(C4)) \tilde{\wedge} (1.0; \text{HighSat}(C5))$$
$$1.0 : \text{LowSat}(C1) \leftarrow (1.0; \text{LowSat}(C2))$$
$$1.0 : \text{LowSat}(C1) \leftarrow (1.0; \text{LowSat}(C3))$$
$$1.0 : \text{LowSat}(C1) \leftarrow (1.0; \text{LowSat}(C4))$$
$$1.0 : \text{LowSat}(C1) \leftarrow (1.0; \text{LowSat}(C5))$$

The generated wf-rules are then combined with the previously generated CNF expression to produce the required conditional wf-rules (lines 4–6).

However, some child nodes in decomposition rules may be conditional and hence may not be present in the run-time view of the SEB model, yet they appear in the generated rules, e.g. nodes $C4$ and $C5$ in the previous wf-rules.

Let us assume that the active context does not contain condition u_1 which implies that goal $C5$ does not exist in the run-time view of the SEB. Also every wf-rule that can be used to deduct the truth value of $C5$ will not be active as the corresponding CNF expression will contain the clause u_1 which will be false. Hence, as the truth value of HighSat($C5$) and LowSat($C5$) will be used for the deduction of HighSat($C1$) and LowSat($C1$), we must ensure that HighSat($C5$) and LowSat($C5$) are set to values that will not affect the value calculated for $C1$ when applying the reasoning.

To overcome this problem, we iterate over each goal g in the model and if the goal is a conditional one (line 9) we add two conditional fuzzy facts to the set of conditional wf-rules (set CondRB), only if the goal node is a child node of an AND/OR-decomposition rule (lines 10 and 15). In case g is a child of an AND-decomposition we add the fuzzy facts (lines 12 and 13):

$$\mathbf{1.0} : \text{HighSat}(g) \leftarrow (1.0; t) \text{ and } \mathbf{0.0} : \text{LowSat}(g) \leftarrow (1.0; t)$$

while if g is a child of an OR-decomposition we add the fuzzy facts (lines 17-18):

$$\mathbf{0.0} : \text{HighSat}(g) \leftarrow (1.0; t) \text{ and } \mathbf{1.0} : \text{LowSat}(g) \leftarrow (1.0; t)$$

These fuzzy facts will be part of the rule base only if formula $\neg\mathbf{CNF}(Cond(g))$ is true (lines 11 and 16), where :

$$\mathbf{CNF}(\{a_1, a_2, \ldots a_n\}) = a_1 \vee a_2 \vee \ldots \vee a_n$$

that is equal to u_1 for node $C3$. This implies that the fuzzy facts participate in the rule base only if node g does not exist in the model, and hence all wf-rules that can be used to deduct its value are not present in the rule base. This is the reason for explicitly assigning values to HighSat(g) and LowSat(g).

Finally, as the type of decomposition rule differentiates the conditional fuzzy facts added to the conditional rule base, by adding the pseudonodes in phase 1 we ensure that in phase 2 there will be no node that appears as a child node to both AND and OR decomposition gm-rules.

7.2. Rule-base determination and fuzzification

Rule-base determination step includes the evaluation of the CNF formulas related to each conditional wf-rule generated in the previous step, taking into account the active context. On the other hand, fuzzification process which is a standard process in fuzzy reasoners, involves the transformation of the observable characteristics to fuzzy facts with the help of one membership function for each fuzzy atom, i.e. LowSat and HighSat. In this paper, LowSat(g)/HighSat(g) membership value corresponds to the truth value of the statement "Goal g is poorly/highly satisfied", and the membership functions for the two atoms depend on whether g is a crisp or a fuzzy goal.

More specifically, given a crisp goal g_c with a satisfaction degree s equals to 100% (g_c is true) or 0% (g_c is false), we define the membership values w_L and w_H for LowSat(g_c) and HighSat(g_c) respectively by utilizing the following membership functions:

$$w_L(s) = \begin{cases} 1, & s = 0 \\ 0, & s \neq 0 \end{cases} \quad w_H(s) = \begin{cases} 0, & s \neq 100 \\ 1, & s = 100 \end{cases} \tag{11}$$

In contrast, given a fuzzy goal g_f with a satisfaction degree equal to s, we calculate the membership values w_L and w_H for LowSat(g_f) and HighSat(g_f) respectively by utilizing the following membership functions illustrated in Fig. 7:

$$w_L(s; a_L, b_L) = \begin{cases} 1, & s \leq a_L \\ \frac{b_L - s}{b_L - a_L}, & a_L \leq s \leq b_L \\ 0, & b_L \leq s \end{cases} \tag{12}$$

$$w_H(s; a_H, b_H) = \begin{cases} 0, & s \leq b_H \\ \frac{s - b_H}{a_H - b_H}, & b_H \leq s \leq a_H \\ 1, & a_H \leq s \end{cases} \tag{13}$$

where a_H, a_L, b_H and b_L are parameters of the framework which must be set before the analysis, and allow for the definition of the two fuzzy predicates according to the needs of the application.

To sum up, for each characteristic with a satisfaction degree s two fuzzy facts will be generated:

$$\mathbf{w_H} : \text{HighSat}(p) \leftarrow (1.0; t) \text{ and } \mathbf{w_L} : \text{LowSat}(p) \leftarrow (1.0; t)$$

with w_H and w_L given by Eq. (11) for crisp goals, and Eqs. (12) and (13) for fuzzy ones. By using the two fuzzy predicates (LowSat/HighSat) we enable the use of fuzzy reasoning, utilizing at the same time the semantics that normally apply to goal models given in the form of wf-rules in Table 2. Nevertheless, this is

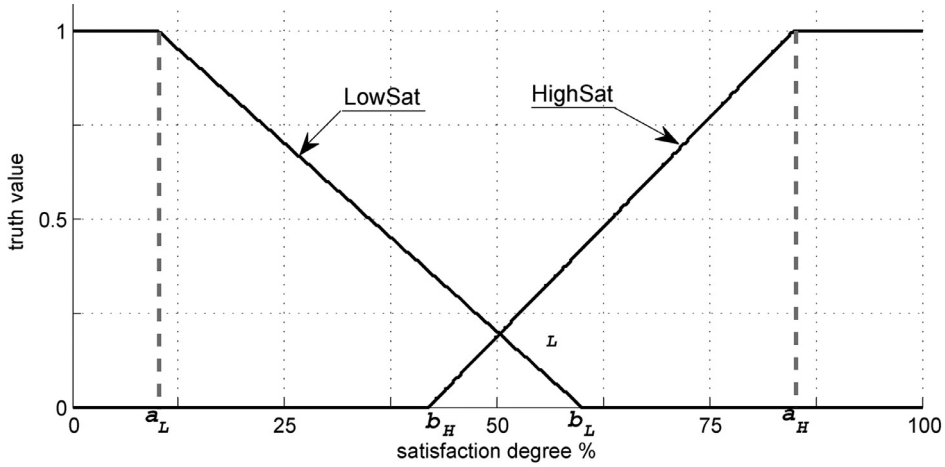


Fig. 7. LowSat/HighSat membership functions for *fuzzy* goals.

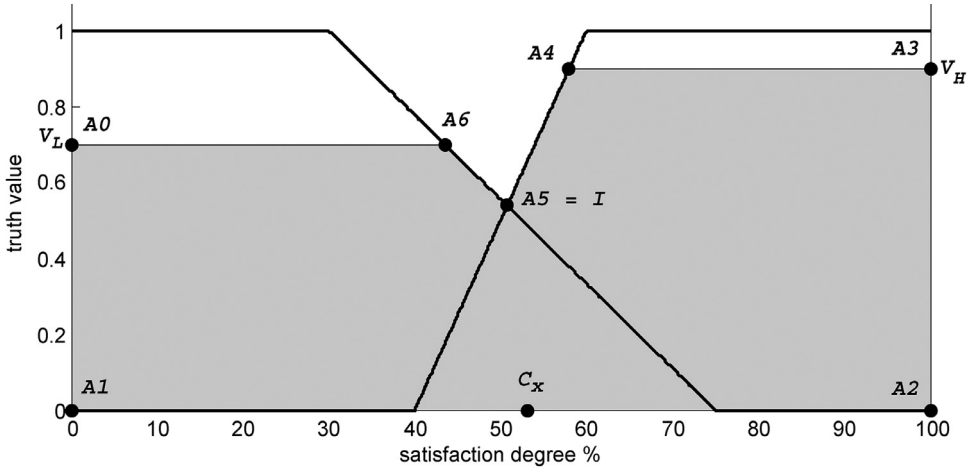


Fig. 8. Defuzzification example ($a_L = 30$, $b_L = 75$, $b_H = 40$, $a_H = 60$, $V_L = 0.7$, $V_H = 0.9$).

not the only combination of fuzzy predicates that can be used, we can rather define membership functions for additional predicates (e.g. MediumSat), which however will require more complicated wf-rules, resulting thus in a more complex analysis.

7.3. Inference

The reasoner introduced in [14] is used as the inference mechanism of the proposed framework. The rule-base, and the fuzzy facts generated through the fuzzification process (i.e w_H and w_L membership values for each leaf node) are utilized by the reasoner in order to deduct truth values for HighSat and LowSat fuzzy predicated for all system goals. The HighSat and LowSat values calculated for each fuzzy or crisp root node can then be combined into a quantifiable result, i.e. satisfaction degree, by applying a proper defuzzification process.

7.4. Defuzzification

The defuzzification process used for each system goal depends on the type of the goal. For crisp system goals we apply a quite straightforward defuzzification by classifying system goals with LowSat = 1.0 and HighSat = 0.0 as false (0% satisfaction degree), and system goals with LowSat = 0.0 and HighSat = 1.0 as true (100% satisfaction degree). In case LowSat and HighSat values are both 1.0 and 0.0 for a system goal, then no conclusion can be drawn for this goal (i.e. its value is unknown). As for values in the interval (0,1), it is not actually possible for crisp goals to receive such partial values due to the constrains of Table 1, which implies that all rules with a crisp consequent have crisp antecedent and weight equal to 1.

In contrast, for fuzzy system goals a more complex defuzzification process is utilized, namely centroid defuzzification, which is one of the most commonly used defuzzification techniques [30]. The Centroid defuzzification method calculates the center of "gravity" for the area under the combined membership function. The x coordinate of the centroid is the defuzzified value.

As an example, consider the membership functions depicted in Fig. 8 for which $a_L = 30$, $b_L = 75$, $b_H = 40$, $a_H = 60$. Given that LowSat and HighSat values calculated for a root goal are $V_L = 0.7$ and $V_H = 0.9$ respectively, the defuzzied value is the centroid x-coordinate (C_x) for the shaded area of the figure. The area is a non-self-intersecting closed polygon defined by the sequence A0, ..., A6 of vertices, hence we use Eq. (8) to compute the defuzzified value of the fuzzy goal which in this case is $C_x = 53.17$ %.

Depending on the parameters of the membership functions and on the LowSat and HighSat values calculated for a fuzzy system goal, the area under the combined membership function may have various forms. In all cases however, this area consists of non-self-intersecting closed polygons, allowing for the use of Eqs. (8) and (9).

Finally, defuzzification is also used to calculate the initial value that should be assigned to fuzzy leaf nodes that either are not

Table 3
Reasoning results for the SEB illustrated in Fig. 4.

Input values										Output values		
Seq.	Active context	A5 (%)	A6 (%)	B2	B3	C2	C3	C4	C5	A1 (%)	B1	C1
1	$\{u_3, \top\}$	50	30	F	T	T	T	–	–	19.3	T	T
2	$\{u_1, u_3, \top\}$	50	30	F	T	T	T	–	T	18.7	T	T
3	$\{u_1, u_2, u_3, \top\}$	50	30	F	T	T	T	F	T	18.7	T	F
4	$\{u_1, u_2, u_3, \top\}$	50	30	T	T	T	T	F	F	73.9	T	F
5	$\{u_1, u_2, u_3, \top\}$	50	70	T	T	T	T	F	F	79.8	T	F

measurable or its satisfaction degree cannot be calculated from the characteristics collected from the running system. More specifically, we calculate the defuzzified value for the extreme case in which LowSat(g)=1 and HighSat(g)=1. We denote this value as V_u, and assign it to all fuzzy leaf nodes with an unknown initial value. More details on V_u will be presented in Section 9.3.

8. Lab experiment

As a proof of concept we developed a prototype of the ReqRV inference engine, and a simulator for a data management application[1] which uses the SEB model of Fig. 5. We designed the SEB model using Eclipse *ecore* tools, and we applied model to model and model to text transformation techniques to generate the conditional wf-rules. The parameters of the membership functions were set to a_L=5, b_L=55, a_H=95 and b_H=45, resulting in symmetrical membership functions for HighSat and LowSat, in order not to enhance any fuzzy predicate over the other. For the selected parameters the value of V_u used as the initial value for unknown fuzzy leaf nodes is equal to 50%. More details on parameters selection will be presented in Section 9.3.

The simulator generated a sequence of truth values (i.e. satisfaction degrees) for leaf nodes *B*2, *B*3, *C*2, *C*3, *C*4, *C*5, and *A*6 of the SEB depicted in Fig. 5. The remaining leaf node, i.e. node *A*5, is considered to be unknown (i.e. 50% satisfaction degree). Also, for each sequence a different active context is applied. For each combination of leaf node values and context we calculate the satisfaction degrees for the three root goals, *A*1, *B*1 and *C*1. A subset of the generated sequences that we think better illustrates how the calculated satisfaction degrees change as a consequence of alterations in the truth values of the leaf nodes and in the active contexts are presented in Table 3. For each sequence, Table 3 contains the values assigned to all leaf nodes (i.e. Input Values), the active context used, and the calculated satisfaction degrees for the root nodes (i.e. Output Values).

Initially (Seq. 1), neither of *C*5 and *C*4 are present in the model as conditions u_1 and u_2 are not in the active context. However, the AND-decomposed root goal *C*1 is satisfied, as the remaining two child nodes that exist in the run-time view are true. Additionally, the OR-decomposed root goal *B*1 is also satisfied as one of its child nodes (node *B*3) is true. Regarding the fuzzy nodes of the example, *A*5 satisfaction degree is equal to 50% (LowSat(A5) = 0.1, HighSat(A5) = 0.1) and *A*6 satisfaction degree is equal to 30% (LowSat(A6) = 0.5, HighSat(A6) = 0.0). As *A*4 is OR-decomposed to nodes *A*5 and *A*6, the following rules will be used for its membership degrees calculation:

$$1.0 : \text{LowSat}(A4) \leftarrow (1.0; \text{LowSat}(A5)) \tilde{\wedge} (1.0; \text{LowSat}(A6))$$
$$1.0 : \text{HighSat}(A4) \leftarrow (1.0; \text{HighSat}(A5))$$
$$1.0 : \text{HighSat}(A4) \leftarrow (1.0; \text{HighSat}(A6))$$

resulting in LowSat(A4) = $\top_{prod}(0.1, 0.5) = 0.05$, and HighSat(A4) = $\perp_{sum}(0.1, 0.0) = 0$ (see Eqs. (5) and (4)). In turn, the membership degrees of the AND-decomposed *A*1 fuzzy root node are calculated by the following rules:

$$1.0 : \text{HighSat}(A1) \leftarrow (1.0; \text{HighSat}(A2)) \tilde{\wedge} (1.0; \text{HighSat}(A3)) \tilde{\wedge} (1.0; \text{HighSat}(A4))$$
$$1.0 : \text{LowSat}(A1) \leftarrow (1.0; \text{LowSat}(A2))$$
$$1.0 : \text{LowSat}(A1) \leftarrow (1.0; \text{LowSat}(A3))$$
$$1.0 : \text{LowSat}(A1) \leftarrow (1.0; \text{LowSat}(A4))$$

resulting in LowSat(*A*1) = $\perp_{sum}(0.0, 0.8, 0.05) = 0.81$, as LowSat(*A*3)=0.8 because of the D^P contribution link from *B*2 and LowSat(*A*4)=0.05, and HighSat(*A*1)=$\top_{prod}(1.0, 0.0, 0.0)$=0, as HighSat(*A*2)=1.0 because of the S^P contribution link from *B*3. By combining the membership degrees of node *A*1 we calculate an overall satisfaction degree 19.3%.

Consequently (Seq. 2 and 3), nodes *C*5 and *C*4 are gradually added in the model as conditions u_1 and u_2 become true. In both cases, the calculated satisfaction degree for *A*1 changes only slightly from the previous case as a consequence of the addition of node *C*5 in the model which is true. The S^N contribution from *C*5 to *A*4 results in the increase of LowSat(A4) from 0.05 to 0.715, which increases LowSat(A1) from 0.81 to 0.943, and decreases the overall satisfaction degree for *A*1 to 18.7%. Additionally, in Seq. 3 *C*1 becomes false as *C*4, which is now part of the model, is false.

Finally, when *B*2 changes from false to true (Seq. 4), the satisfaction degree of *A*1 changes to 73.9% as *B*2 contributes positively *A*3, which is a child node of *A*1. Similarly, when the value of *A*6 is increased to 70% (LowSat(*A*6) = 0.0, HighSat(*A*6) = 0.5) in the next sequence (Seq. 5), the satisfaction degree of *A*1 is increased to 79.8%

9. Experiments and discussion

The performance of the proposed framework is evaluated by a series of experiments with randomly generated SEB models of varying size and complexity.

More specifically, we evaluate the proposed framework with respect to execution time and amount of memory required for models of different sizes in order to prove that time and memory requirements remain low even for large models, enabling thus its application to large models at run-time. Subsequently, we investigate the effect of membership function parameters to the reasoning process and give guidelines for selecting their values. Furthermore, we evaluate the sensitivity of the results to the weights assigned to contributions and prove that reasoning is stable to small variations of the weights.

9.1. Experiment setting

For the purposes of experimentation, we have designed a driver, which given certain parameters produces a randomly generated SEB model. Subsequently, we generated models with the following configurations:

- minimum total goal nodes in the model: 40–1000, with an interval of 20 nodes
- maximum number of child nodes per node: 4, 8, 12 ,16, 20

[1] The application and example input files can be downloaded from http://www.softlab.ntua.gr/~gechatz/seb/

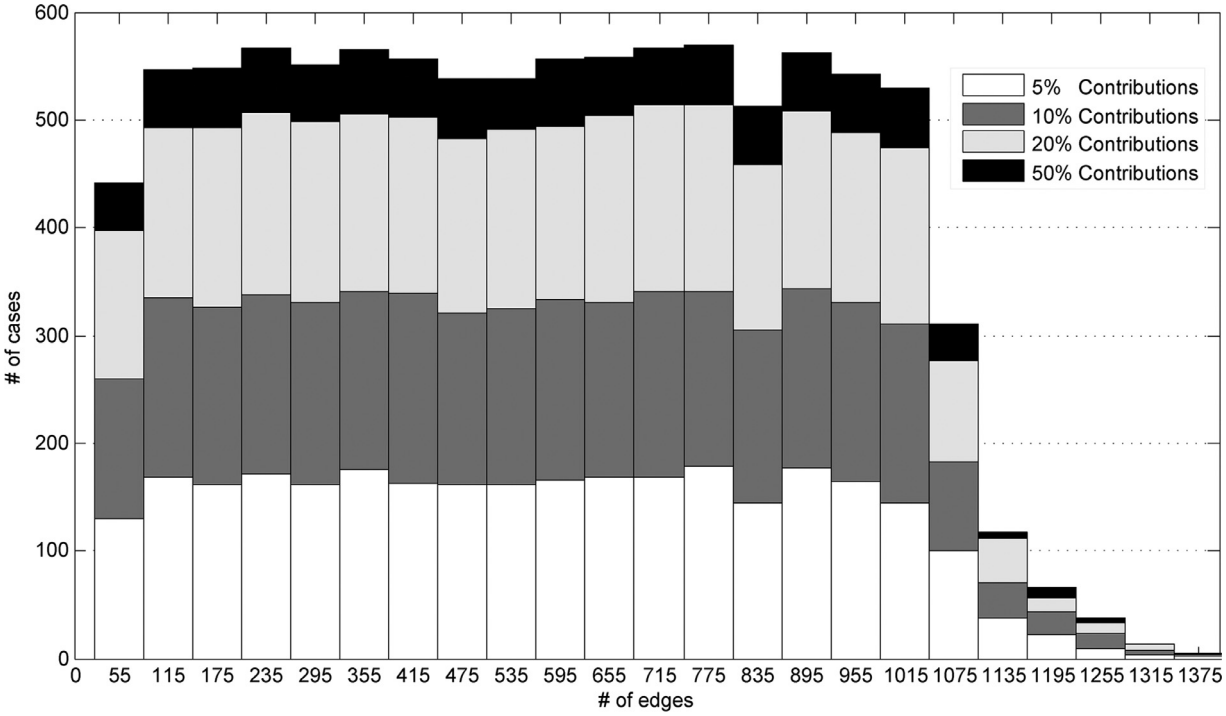


Fig. 9. Generated SEB models classification according to total number of edges and percentage of contribution links.

- AND vs. OR decomposition ratio: 1
- fuzzy vs. crisp number of nodes ratio: 1
- percentage of contribution links in the model: 5% , 10%, 20%, 50%
- contribution type (S^P, S^N, D^P, D^P) probability: 25 % per type

and also produced an initial assignment for the leaf nodes of each model, so as to produce the fuzzy facts required as input to the inference engine.

9800 models were generated in total, which were separated in 23 groups according to their total number of edges. We used the total number of edges in the SEB as the index for the complexity of the model, as edges (i.e. contribution and decomposition links) describe dependencies between nodes of the model, and hence add wf-rules to the rule-base, increasing thus its size.

Groups were further divided in 4 sub-groups according to the percentage of edges that correspond to contribution links. The number of models belonging to each group and sub-group are illustrated in Fig. 9. In particular, the first group (i.e. the first bar in the chart) contains all models that have a total number of edges in the interval [25, 85]. Each group is represented from the midpoint of the corresponding interval, e.g. for the first group the midpoint is 55. From the 441 models that belong to this group, 130 have 5% contribution links, 130 have 10% contribution links, 138 have 20% contribution links, and finally 43 have 50% contribution links.

Fig. 9 depicts that the distribution of models in the various groups are almost identical for the first 17 of them, while only a small fraction of the models fall into the last 6 groups. Hence, we only use the models that belong to the first 17 groups to evaluate the performance of the proposed framework.

9.2. Scalability

We generated the wf-rules for each SEB model that fall in one of the first 17 groups depicted in Fig. 9, we ran the reasoning process for each generated model, and measured a) the memory used by the reasoner, and b) the time required for the reasoning to complete.

The memory required for the reasoning increases linearly to the size of the models, with a value equal to 125KB for models with a mean number of edges equal to 55, and a value equal to 4MB for models with a mean number of edges equal to 1015.

In contrast, time has a polynomial growth rate with respect to the number of edges as this is illustrated in Fig. 10. For models of the same number of edges, as the percentage of contribution edges increases the time required to complete the reasoning increases to. This fact implies that the addition of contribution links increases the complexity of the rule base used by the inference engine. In any case however, time and memory requirements remain low even for large models.

9.3. Effect of membership function parameters

In this section we investigate the effect of a_L, b_L, b_H, a_H parameters to the values computed via the defuzzification method described in Section 7.4. More specifically, we investigate how the values of the parameters influence the satisfaction degree that will be used to represent the unknown case, and also the values that will represent satisfied/dissatisfied goals.

As discussed in Section 7.4 the values of the four parameters a_L, b_L, b_H, a_H determine the area under each membership function (Fig. 8), which is used to compute the centroid x-coordinate value C_x, given the values for LowSat(g) and HighSat(g) for a goal g. To evaluate the effect different membership functions (i.e. the parameters) have on the obtained results, we consider four indicative classes of membership functions as depicted in Fig. 11. For each membership function in Fig. 11, we compute the resulting defuzzified satisfaction degrees for various pairs of LowSat and HighSat values. The results are illustrated in Fig. 12, where each subfigure depicts the defuzzified satisfaction degree for the corresponding membership functions (e.g. Fig. 12(a) depicts the results for Fig. 11(a)).

Let us consider the C_x value (i.e the satisfaction degree) calculated for the following three extreme cases: *a)* LowSat(g) = 0 and HighSat(g) = 1 (fully satisfied) denoted as V_s; *b)* LowSat(g) = 1

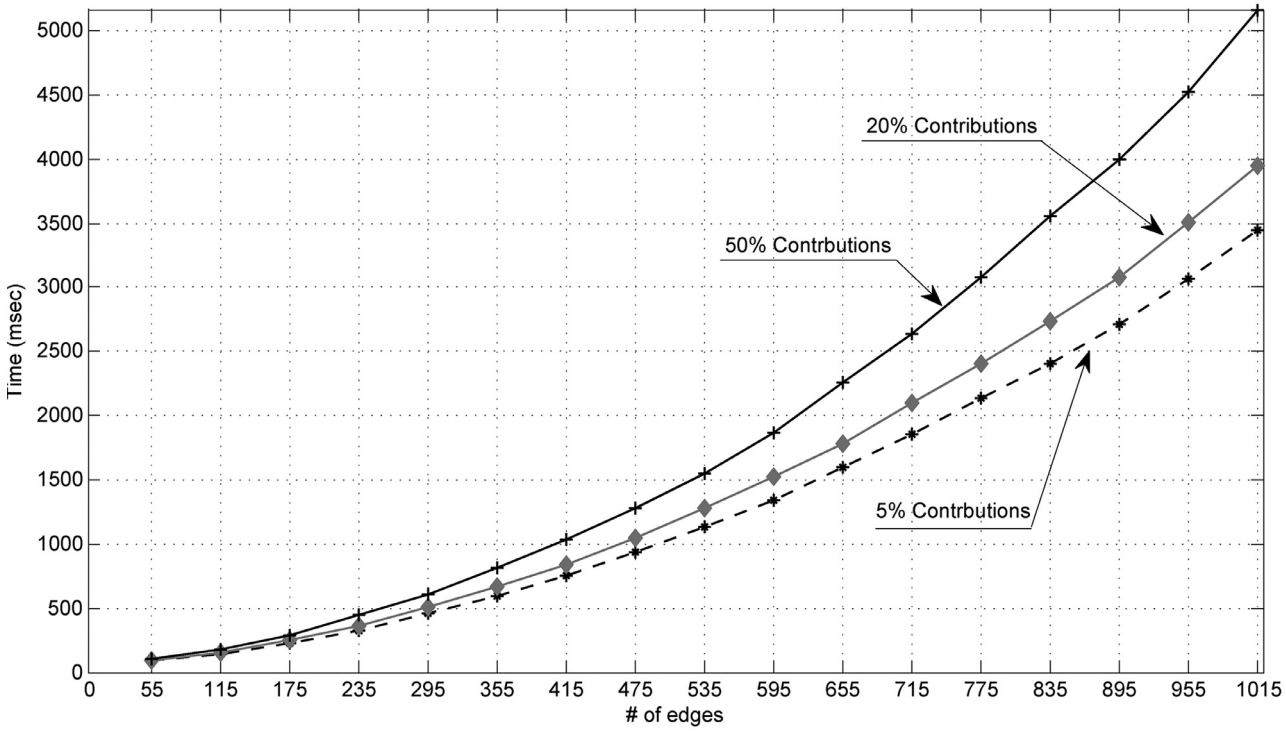


Fig. 10. Time required for reasoning to complete vs. the total number of edges in the SEB model.

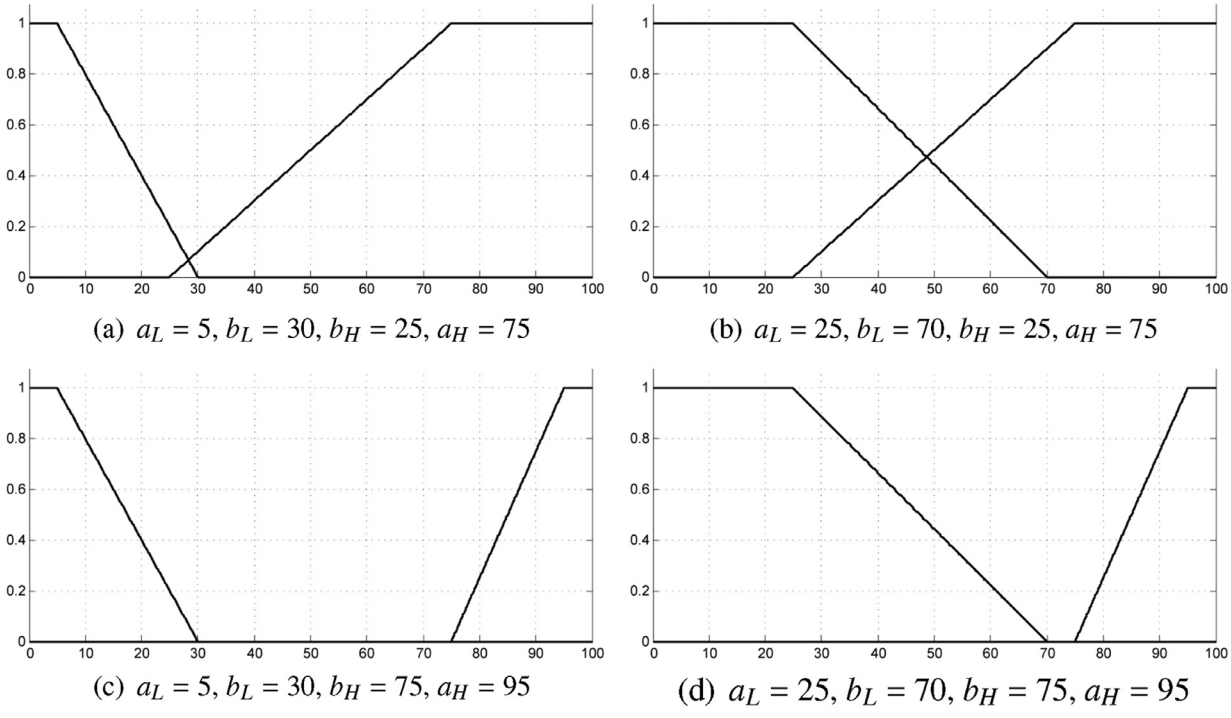
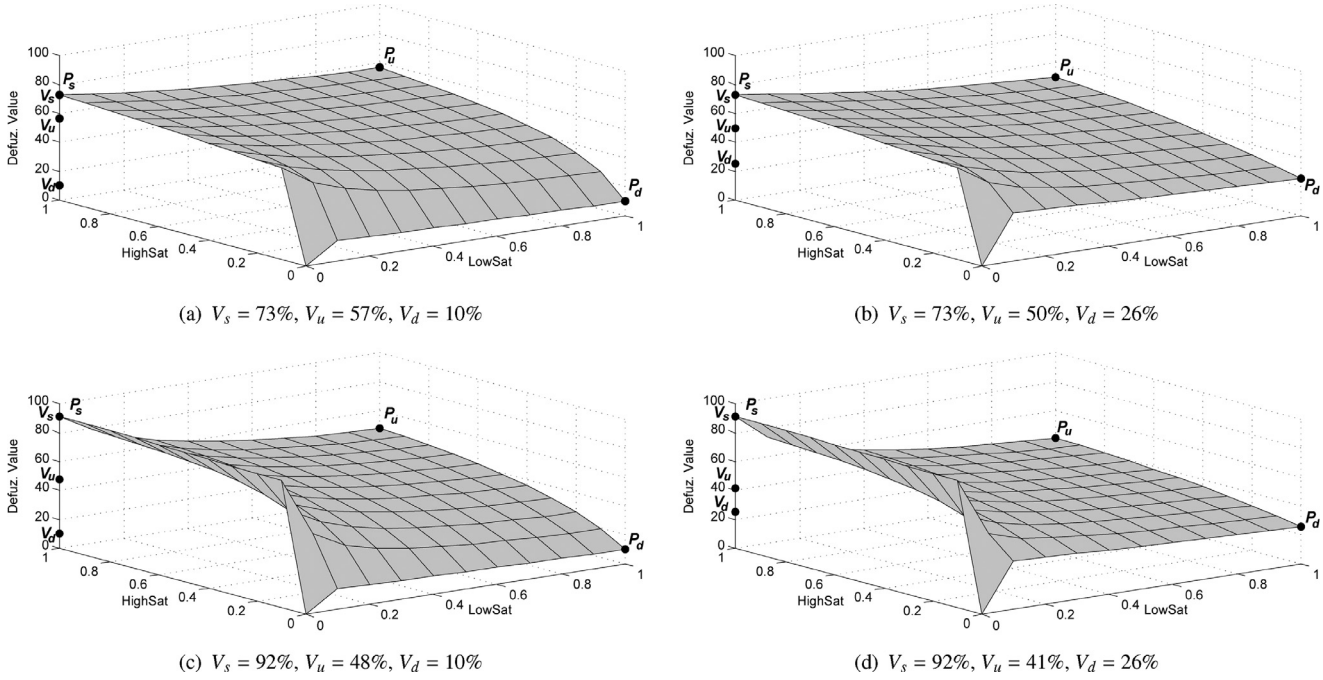


Fig. 11. Four indicative classes of membership functions.

and HighSat(g) = 1 (unknown) denoted as V_u; and *c)* LowSat(g) = 1 and HighSat(g) = 0 (fully denied) denoted as V_d.

The values of V_s, V_u and V_d are the z-coordinates of the points P_s, P_u and P_d respectively, depicted in Fig. 12. These values define two intervals in the z-axis (i.e. in the "Defuz. Value" space), namely interval $[V_d, V_u)$, and interval $(V_u, V_s]$. Goals with a satisfaction degree in the interval $[V_d, V_u)$ are goals that are not satisfied, while goals with satisfaction degrees in the interval $(V_u, V_s]$ are satisfied. Once the intervals are defined, sub-intervals can be chosen to represent classes of denial or satisfaction (e.g. "Highly Denied", "Highly Satisfied") according to the needs of the analysis. Hence, the values of the four parameters essentially define the lengths of the two intervals $[V_d, V_u)$, and $(V_u, V_s]$, and also the ability to define subclasses within the intervals, if required.

When the area under the LowSat membership function is approximately equal to the one under HighSat membership function



(a) $V_s = 73\%$, $V_u = 57\%$, $V_d = 10\%$

(b) $V_s = 73\%$, $V_u = 50\%$, $V_d = 26\%$

(c) $V_s = 92\%$, $V_u = 48\%$, $V_d = 10\%$

(d) $V_s = 92\%$, $V_u = 41\%$, $V_d = 26\%$

Fig. 12. Satisfaction degrees calculated via defuzzification for the membership functions of Fig. 10.

(Fig. 11(b) and (c)) the value for V_u is approximately equal to 50 % (Fig. 12(b) and (c)). When the area under HighSat is greater than the one under LowSat (Fig. 11(a)) the satisfaction degree calculated tends to increase and diverge from 50% ($V_u = 57\%$ in Fig. 12(a)). In contrast, i.e. when the area under LowSat is greater than the one under HighSat (Fig. 11(d)), the satisfaction degree calculated tends to decrease and diverge from 50% ($V_u = 41\%$ in Fig. 12(a))

Furthermore, a small area under LowSat membership function has as a consequence the increase of the length of the interval [V_d, V_u) as it decreases the value of V_d. For example, $V_d = 10\%$ for Fig. 11(a) and (c), while $V_d = 26\%$ for Fig. 11(b) and (d). In a similar manner, the smaller the area under HighSat the larger the value of V_s, and hence the larger the length of the interval (V_u, V_s]. For example, $V_s = 92\%$ for Fig. 11(c) and (d), while $V_s = 73\%$ for Fig. 11(a) and (b).

9.4. Stability

To exhibit that the proposed reasoning is stable with respect to reasonable variations of the weights on contribution links a modeler may introduce due to subjectivity on defining such weights, we conducted a series of experiments on how results are affected, when weights for a fraction of the contribution links are altered.

We utilized 16 binary trees of various sizes and then randomly selected a subset of the nodes that would be the target of a number of S^P contribution links. The number of contribution links added to a given model was the 50% of the total number of edges in that model, and the weight were randomly selected in the interval[2] [0.0, 0.625]. For each contribution added in the model, a source node was also added with an initial value set to true (HighSat = 1.0, LowSat = 0.0), while the initial values of all leaf nodes of the model were set to false, so as to record only the changes caused by the alterations of the weights assigned to contributions. For each case, we modified the 10%, 20% and 50% of the weights to the ± 10%, ± 20%, ± 40% and ± 60% of the initial value, and recorder the normalized percentage change of the value calculated for the HighSat node predicate.

From the results depicted in Fig. 13 we can notice that the growth rate of % Normalized Result Change is smaller in "10% Weight Value Change" than the one in "60% Weight Change" case. For example, when the value of the 20% of the weights is changed by 10% (point B in Fig. 13) the % Normalized Result Change increases to 3% from 2.61% which is the corresponding value when the 10% of the weights is changed by 10% (point A in Fig. 13), resulting in a 0.40% increase. In contrast, for the "60% Weight Value Change" case, the corresponding increase (increase between points D and C in Fig. 13) is approximately equal to 3%. Hence, the smaller the % Weight Value Change, the smaller the growth rate of the % Normalized Result Change.

Additionally, in all four cases the % Normalized Result Change, which is an index of how fast the results computed via the reasoner are altered vs the % of Weights Affected, is modified linearly to the % of Weights Affected. Hence, the calculated results are modified in a stable manner as the % of Weights Affected increases.

9.5. Threats to validity and discussion

There are two threats to the validity of the proposed method, namely the size of the models, and the contribution links' weights elicitation. The former is related to how goal models with many edges and nodes, like the ones used in the evaluation of the framework, are composed and maintained. In this paper we take the view that stakeholders cannot only define custom models, but they can also use predefined ones from repositories, created from experts or extracted utilizing text mining techniques from specifications and policy documents [31]. Using such repositories composite goal models of varying size and complexity can be created.

Additionally, even though in *qualitative* reasoning, label propagation becomes rapidly inconclusive as we move up in the softgoal refinement tree [32], a common problem encountered to almost every *quantitative* reasoning approach is how weights are assigned to contribution links. This problem, and whether it is

[2] The upper limit was set to 0.625 in order to ensure that when weights are changed to +60% of the initial value w_{init}, the new weight will still be less than or equal to 1 ($1.6w_{\text{init}} \leq 1 \Rightarrow w_{\text{init}} \leq 1/1.6 \Rightarrow w_{\text{init}} \leq 0.625$)

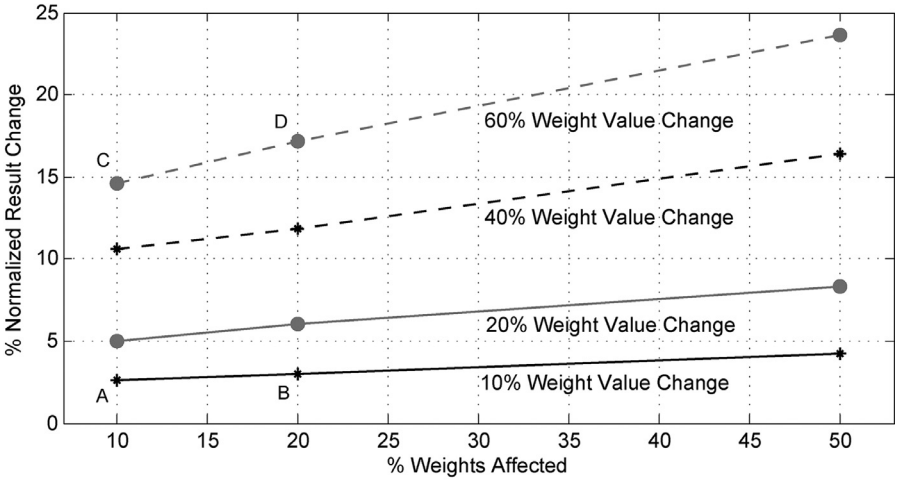


Fig. 13. % normalized result change vs. % of weights altered by ± 10%, ± 20%, ± 40% and ± 60% of the initial value.

easier to comprehend qualitative rather than quantitative labels is the subject of [13,33], where authors present results which verify that the use of numerical values increases the comprehension of the models even for non-experts, and they also propose an elicitation method.

Furthermore, the fuzzy reasoner utilized by the framework allows for weights in wf-rules to be calculated through a training process. This process can also be used in our case to elicit the weights in wf-rules generated from the goal model. Given the wf-rules, and a data set that contains for various contexts valid combinations of HighSat and LowSat membership degrees for the goals appear in the rules, i.e. both leaf and internal nodes, we can extract numerical values for the unknown weights. This training process for weighted fuzzy rules is described in detail in [14, p. 114], where authors define the problem of weighted fuzzy logic programs adaptation as an optimization problem aiming to determine the fuzzy atom weights of a set of wf-rules.

Moreover, the reasoning process presented in this paper can be applied even if some of the leaf nodes are nonmeasurable, by considering them as being unknown. However, the percentage of nodes that are unknown, and are used as input to the reasoner should be taken into account when reviewing the inferred satisfaction degrees, as a high percentage of unknown leaf nodes will tend to produce results of low trust.

Finally, because of the conditional elements, models may contain composite goals without child nodes (i.e. all child nodes are related to conditions that are false), which result in "incomplete" models. While these cases can be handled by the proposed framework by considering the composite goals as being unknown, we assume that models are "complete" and focus on the fuzzy reasoning technique.

Comparison to probabilistic approaches: probabilistic approaches like [4] are not equivalent to fuzzy ones but rather complementary as the former serves the purposes of elicitation rather than estimation [34]. Additionally, there are some key aspects of fuzzy logic, mainly related to expressiveness and interpretation of the results [34,35], that make fuzzy approaches more appropriate for: *a)* modeling vagueness of human reasoning and expertise as opposed to uncertainty; and *b)* denoting problems where an event has actually occurred (i.e. probability of the event is one) but we are vague on how this impacts higher level nodes, as opposed to problems where we have to assume the probability of occurrence of an event and how this probability affects the probability of satisfaction of the parent nodes.

This difference is important for modeling and reasoning purposes as in the case of probabilistic approaches the focus in on the uncertainty of events, while in our approach the focus is on the vagueness of how an already observed event affects system goals.

Membership functions: selecting the most appropriate membership functions is one of the most important problems of fuzzy controllers development. While there are some general guidelines proposed in the literature, and methods have been proposed for the elicitation of membership functions using genetic algorithms [36], membership function selection is mainly based on domain expertise and results interpretation needs [23]. We give some general guidelines regarding the membership function parameters in Section 9.3, however the proposed approach is not bound to the membership functions depicted in Fig. 7, and the same technique applies also to sigmoidal or trapezoidal-shaped ones. We only have to substitute Eqs. (8) and (7) with the equivalent formulas with integrals if the area under the combined membership function cannot be divided into a set of non-self-intersecting closed polygons.

Fuzzy operators: as it is depicted in Fig. 2 the reasoning process depends on the fuzzy operators $\Omega^{prod}_{\langle w_1,\cdots w_n\rangle}$ Eq. (5) and $\perp_{sum}$ Eq. (4). However, these are not the only operators supported by the reasoner. In contrast, instead of operator $\Omega^{prod}_{\langle w_1,\cdots w_n\rangle}$ operators $\Omega^{luk}_{\langle w_1,\cdots w_n\rangle}$ (Łukasiewicz *t*-norm) and $\Omega^{drastic}_{\langle w_1,\cdots w_n\rangle}$ (drastic *t*-norm) defined in [14] can be used. All three operators produce results that do not change dramatically for small variations of the weights as it is shown in [14, p. 112], and hence provide results which are stable to variations of the weights. Furthermore, in addition to operator $\perp_{sum}$ operators $\perp_{max}$ (maximum *s*-norm) and $\perp_{luk}$ (Łukasiewicz *s*-norm) can also be used in the analysis. The ability to select the operators provides an additional extension point of the proposed framework.

10. Related work

Approaches that utilize goal models for run-time analysis and adaptive software include [37] where the distinction between Design-time and Run-time Goal Models is made, [38] which proposes a set of extensions for the Tropos methodology related to failures and environment modeling, [39,40] where the notions of awareness (AwReqs) and evolution (EvoReqs) requirements are introduced as meta-requirements allowing to model requirements adaptation at run-time, and [21] that provides a general overview of how fuzzy goals can be handled as run-time abstractions. Also, authors in [41] use goal models for run-time analysis, however

in this case goal models are used as a mean to implement self-reconfigurable socio-technical systems, and authors apply a qualitative analysis in contrast to the quantitative analysis used in the proposed approach.

Furthermore, reasoning on goal models has been extensively studied in the literature mainly as a mean to detect contradictions or evaluate alternatives [42,43]. In [44] authors introduce a method for compliance verification for security requirements, expressed in FOL over secure Tropos models, using Datalog to check for contradictory requirements. Additionally, in [45] goal models with integer weights on nodes are used to describe tasks, while in [46] a genetic algorithm is introduced for run-time reasoning. The reasoning applied in this case provides a top-down analysis of the goal model, just like in [47] where authors aim at detecting inconsistencies and conflicts in contextual goal models, or [48] where authors apply the even swaps multi-criteria decision analysis method to determine the best solution among alternatives. A reasoning technique about contextual goal models is the subject in [11], where the idea of contextual goal models is introduced as a way to relate goals with context, and authors propose a design time reasoning technique in order to extract the combination of requirements that will be valid (i.e. there will be no contradictions), and will ensure the minimum cost for the system under development. A limitation of these approaches is that they can neither denote and handle non-functional requirements nor perform quantitative reasoning, as opposed to the inference mechanism utilized by the proposed framework.

A quite similar method to the proposed one that also performs quantitative reasoning over AND/OR goal trees is introduced in [4]. However, there are some key differences between the two approaches. First and foremost, authors in [4] propose a probabilistic reasoning process which outputs two distinct probabilities for each goal, one for the goal to be satisfied, and a second probability for it to be denied. However, those two values are not combined to a single quantitative result as is the case for our approach. Moreover, while weights on contribution links are used in both approaches, the semantics differ. In [4] the weight of a S^P contribution is defined as the conditional probability of the target node to be satisfied given that the source node is satisfied, while in the present approach the weight is the measure of increased belief in target satisfaction when the source node is satisfied. By comparing the results between [4] and our approach for the example goal model given in [4][3] we notice that in terms of values the results were very similar due to the fact that the formulae used to propagate the values are based on probabilistic sum s-norm and product t-norm which are the same equations used for fuzzy reasoning. However, their interpretation is different in [4] as they correspond to probabilities and not to truth values as in our approach. Furthermore, our framework can handle conditional contributions and is quite flexible as described in Section 9.5 allowing the use of various t-norms.

Another approach that uses quantitative reasoning over goal trees is the one presented in [49], where goal trees are utilized to describe software project management metrics, and an MLN reasoner is used to calculate the probabilities of root goals to be satisfied. A probabilistic framework is also introduced in [50] for the calculation of obstacles' probabilities that can result in certain goals' failure. Knowing these probabilities allows for the selection of an appropriate alternative countermeasure. Furthermore, a method that can assist decision making among alternatives is also presented in [51], where both qualitative and probabilistic goal reasoning techniques are applied to Business Intelligence Models, and also in [52] where goal trees are transformed to Dynamic Decision Networks (DDNs) allowing thus a probabilistic reasoning against goals models. Probabilistic reasoning applies to these cases is quite different from the reasoning applied in this paper, as we are interested in the satisfaction degree of a goal rather than its probability to be satisfied.

Additionally, Tanabe et al. in [53] introduce a technique for quantitative reasoning over goal trees with contribution values attached to the edges. Reasoning in this case aims at determining a total impact degree that expresses the influence of the deletion/addition of an element to the goal graph. The computation is based on a top-down analysis, however the technique cannot be applied to the ReqRV problem, where the nature of the problem demands a bottom-up approach as we already know the values for some of the leafs and the question is to what extend the roots are satisfied.

Quantitative goal models are also studied in [54] in order to evaluate alternative design decisions expressed in the KAOS goal modeling language which is a different problem from the one studied in this paper. A significant difference is that in [54] goals are annotated with objective functions and quality variables, with the former being utilized for the evaluation of the satisfaction degree of each goal according to the values of the latter. By defining specific probability distributions for the quality variables, a stochastic simulation can be used in order to compute the values of the objective functions for each alternative and subsequently select the one with the highest score.

11. Conclusion and future work

In this paper, we presented an approach that utilizes conditional weighted goal models to denote system goals related to requirements that should hold in the running system. The choice of goal models as the modeling notation for the proposed framework is based on the fact that it is a notation extensively used in requirements engineering to represent functional and non-functional requirements. Additionally, its well defined semantics enable the development of an automated algorithmic transformation process for generating conditional wf-rules from conditional and weighted goal models. These rules, in combination with facts representing observed or monitored system data, form a knowledge base that can be used by a fuzzy reasoner to calculate the satisfaction degrees for system goals when the system is altered or adapted.

In order to evaluate the performance of the proposed framework we conducted a series of experiments with randomly generated models of varying size and complexity. Using these models we evaluated the application of the proposed method with respect to execution time and amount of memory required for models of different sizes. The experimental results indicate that the amount of time required for the reasoning to complete remains small even for large models, and hence the proposed framework can be utilized at run-time. Furthermore, we evaluated the sensitivity of the results calculated from the proposed inference engine to the weights assigned to contributions and have shown that the reasoning process is stable with respect to small variations of the contribution link weights provided by the system analysts.

Finally, the work presented in this paper can be extended in a number of possible directions. One direction is to investigate the application of clustering techniques in order to compute collections of mutually independent sets of goal model nodes, which can be evaluated separately. This approach can be used to parallelize the goal model evaluation process, maximizing thus the overall performance of the proposed system. Additionally, by identifying independent sets of goal model parts, it would be feasible to run the reasoning only for parts of the model that are affected by the changes occurred, instead of applying the reasoning to the entire model, even for parts that remain unchanged. Another direction is

[3] Results are listed in http://www.softlab.ntua.gr/~gechatz/seb/

to consider different reasoning strategies applied to the crisp and the fuzzy parts of the model. For the crisp parts, a first order logic reasoner may suffice, while in the fuzzy parts a fuzzy reasoner can be used. This has also the potential to enhance the overall reasoning performance. A fourth direction would be to investigate further extensions on the goal modeling notation in order to include pre/post condition dependencies and temporal dependencies between goal model nodes. These extensions may be proven useful for enhancing the expressiveness of the modeling notation. Finally a fifth direction would be to annotate goals with actions, the execution of which can help towards goals' satisfaction. When goals failures are detected by the proposed framework, actions related to goals could be used to build a remediation plan that could be applied to the system in order to satisfy currently dissatisfied system goals.

Acknowledgments

This research has been co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) – Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund. We would also like to thank A. Chortaras for providing as with the implementation of his reasoner for weighted logic programs.

References

[1] A. van Lamsweerde, Goal-oriented requirements enginering: a roundtrip from research to practice, in: Proceedings of the 12th IEEE International Conference on Requirements Engineering (RE 2004), 6–10 September 2004, Kyoto, Japan, IEEE Computer Society, 2004, pp. 4–7.
[2] A.M. Sharifloo, P. Spoletini, LOVER: light-weight formal verification of adaptive systems at run time, in: C.S. Pasareanu, G. Salaün (Eds.), Proceedings of the 9th International Symposium on Formal Aspects of Component Software, FACS 2012, Mountain View, CA, USA, September 12–14, 2012, Lecture Notes in Computer Science, vol. 7684, Springer, 2012, pp. 170–187.Revised Selected Papers.
[3] Y. Falcone, M. Jaber, T. Nguyen, M. Bozga, S. Bensalem, Runtime verification of component-based systems, in: G. Barthe, A. Pardo, G. Schneider (Eds.), Proceedings of the 9th International Conference on Software Engineering and Formal Methods, SEFM 2011, Montevideo, Uruguay, November 14–18, 2011, Lecture Notes in Computer Science, vol. 7041, Springer, 2011, pp. 204–220.
[4] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, R. Sebastiani, Reasoning with goal models, in: S. Spaccapietra, S.T. March, Y. Kambayashi (Eds.), Proceedings of the Requirements Engineering Conference, ER, Lecture Notes in Computer Science, vol. 2503, Springer, 2002, pp. 167–181.
[5] D. Amyot, S. Ghanavati, J. Horkoff, G. Mussbacher, L. Peyton, E.S.K. Yu, Evaluating goal models within the goal-oriented requirement language, Int. J. Intell. Syst. 25 (8) (2010) 841–877.
[6] I.H. Krüger, M. Meisinger, M. Menarini, Interaction-based runtime verification for systems of systems integration, J. Log. Comput. 20 (3) (2010) 725–742.
[7] R. Calinescu, When the requirements for adaptation and high integrity meet, in: Proceedings of the 8th Workshop on Assurances for Self-adaptive Systems, ASAS '11, 2011, pp. 1–4.
[8] , Lecture Notes in Computer Science, vol. 7740, Assurances for Self-Adaptive Systems – Principles, Models, and Techniques, Springer, 2013.
[9] P. Sawyer, N. Bencomo, J. Whittle, E. Letier, A. Finkelstein, Requirements-aware systems: a research agenda for RE for self-adaptive systems, in: Proceedings of the 18th IEEE International Requirements Engineering Conference, RE 2010, Sydney, New South Wales, Australia, September 27–October 1, 2010, 2010, pp. 95–103.
[10] C. Ghezzi, Engineering evolving and self-adaptive systems: an overview, in: M. Broy, C. Leuxner, T. Hoare (Eds.), Software and Systems Safety – Specification and Verification, NATO Science for Peace and Security Series – D: Information and Communication Security, vol. 30, IOS Press, 2011, pp. 88–102.
[11] R. Ali, F. Dalpiaz, P. Giorgini, A goal-based framework for contextual requirements modeling and analysis, Requir. Eng. 15 (4) (2010) 439–458.
[12] D. Amyot, S. Ghanavati, J. Horkoff, G. Mussbacher, L. Peyton, E.S.K. Yu, Evaluating goal models within the goal-oriented requirement language, Int. J. Intell. Syst. 25 (8) (2010) 841–877.
[13] S. Liaskos, R. Jalman, J. Aranda, On eliciting contribution measures in goal models, in: M.P.E. Heimdahl, P. Sawyer (Eds.), Proceedings of the 2012 20th IEEE International Requirements Engineering Conference, RE, Chicago, IL, USA, September 24–28, 2012, IEEE Computer Society, 2012, pp. 221–230.
[14] A. Chortaras, G.B. Stamou, A. Stafylopatis, Definition and adaptation of weighted fuzzy logic programs, Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 17 (1) (2009) 85–135.
[15] M. Leucker, C. Schallhart, A brief account of runtime verification, J. Log. Algebric Program. 78 (5) (2009) 293–303.
[16] Y. Falcone, J.-C. Fernandez, L. Mounier, Runtime verification of safety-progress properties, in: S. Bensalem, D. Peled (Eds.), Runtime Verification, Lecture Notes in Computer Science, vol. 5779, Springer, 2009, pp. 40–59.
[17] P. Zave, M. Jackson, Four dark corners of requirements engineering, ACM Trans. Softw. Eng. Methodol. 6 (1) (1997) 1–30.
[18] R. Calinescu, C. Ghezzi, M.Z. Kwiatkowska, R. Mirandola, Self-adaptive software needs quantitative verification at runtime, Commun. ACM 55 (9) (2012) 69–77.
[19] A.K. Chopra, F. Dalpiaz, P. Giorgini, J. Mylopoulos, Reasoning about agents and protocols via goals and commitments, in: W. van der Hoek, G.A. Kaminka, Y. Lespérance, M. Luck, S. Sen (Eds.), Proceedings of the Autonomous Agents and Multiagent Systems, AAMAS, IFAAMAS, 2010, pp. 457–464.
[20] J. Mylopoulos, L. Chung, E.S.K. Yu, From object-oriented to goal-oriented requirements analysis, Commun. ACM 42 (1) (1999) 31–37.
[21] L. Baresi, L. Pasquale, P. Spoletini, Fuzzy goals for requirements-driven adaptation, in: Proceedings of the Requirements Engineering Conference, RE, IEEE Computer Society, 2010, pp. 125–134.
[22] Wikipedia.org, Centroid (accessed 24.05.2015).
[23] K.M. Passino, S. Yurkovich, Fuzzy Control, first ed., Addison-Wesley, Longman Publishing Co., Inc., Boston, MA, USA, 1997.
[24] T. Kalamatianos, K. Kontogiannis, P. Matthews, Domain independent event analysis for log data reduction, in: X. Bai, F. Belli, E. Bertino, C.K. Chang, A. Elçi, C.C. Seceleanu, H. Xie, M. Zulkernine (Eds.), Proceedings of the Computer Software and Applications Conference, COMPSAC, IEEE Computer Society, 2012, pp. 225–232.
[25] S. Zaman, F. Karray, Features selection for intrusion detection systems based on support vector machines, in: Proceedings of the 6th IEEE Consumer Communications and Networking Conference, CCNC 2009, IEEE, 2009, pp. 1–8.
[26] R. Ali, F. Dalpiaz, P. Giorgini, Location-based software modeling and analysis: tropos-based approach, in: Q. Li, S. Spaccapietra, E.S.K. Yu, A. Olivé (Eds.), Proceedings of the 27th International Conference on Conceptual Modeling, ER 2008, Barcelona, Spain, October 20–24, 2008, Lecture Notes in Computer Science, vol. 5231, Springer, 2008, pp. 169–182.
[27] P. Giorgini, F. Massacci, J. Mylopoulos, N. Zannone, Modeling security requirements through ownership, permission and delegation, in: Proceedings of the 13th IEEE International Conference on Requirements Engineering, 2005, pp. 167–176.
[28] S. Ingolfo, A. Siena, J. Mylopoulos, Establishing regulatory compliance for software requirements, in: M.A. Jeusfeld, L.M.L. Delcambre, T.W. Ling (Eds.), Proceedings of the 30th International Conference on Conceptual Modeling, ER 2011, Brussels, Belgium, October 31–November 3, 2011, Lecture Notes in Computer Science, vol. 6998, Springer, 2011, pp. 47–61.
[29] Object Management Group, Object Constaint Language (OCL) Version 2.0, http://www.omg.org/spec/OCL/2.0/PDF (accessed 24.05.2015).
[30] T.J. Ross, Fuzzy Logic with Engineering Applications, John Wiley & Sons, 2004.
[31] M.R.I. Nekvi, N.H. Madhavji, Impediments to regulatory compliance of requirements in contractual systems engineering projects: a case study, ACM Trans. Manag. Inf. Syst. 5 (3) (2015) 15:1–15:35.
[32] A. van Lamsweerde, Reasoning about alternative requirements options, in: A. Borgida, V.K. Chaudhri, P. Giorgini, E.S.K. Yu (Eds.), Conceptual Modeling: Foundations and Applications – Essays in Honor of John Mylopoulos, Lecture Notes in Computer Science, vol. 5600, Springer, 2009, pp. 380–397.
[33] S. Liaskos, S. Hamidi, R. Jalman, Qualitative vs. quantitative contribution labels in goal models: Setting an experimental agenda, in: J. Castro, J. Horkoff, N.A.M. Maiden, E.S.K. Yu (Eds.), Proceedings of the 6th International CEUR Workshop, i* Workshop 2013, Valencia, Spain, June 17–18, 2013, vol. 978, 2013, pp. 37–42.CEUR-WS.org
[34] L. Zadeh, Discussion: probability theory and fuzzy logic are complementary rather than competitive, Technometrics 37 (3) (1995) 271–276.
[35] H. Pan, D. McMichael, Fuzzy Causal Probabilistic Networks – A New Ideal and Practical Inference Engine, 1998.
[36] A. Homaifar, E. McCormick, Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms, IEEE T. Fuzzy Syst. 3 (2) (1995) 129–139.
[37] F. Dalpiaz, A. Borgida, J. Horkoff, J. Mylopoulos, Runtime goal models: keynote, in: R. Wieringa, S. Nurcan, C. Rolland, J. Cavarero (Eds.), Proceedings of the IEEE 7th International Conference on Research Challenges in Information Science, RCIS 2013, Paris, France, May 29–31, 2013, IEEE, 2013, pp. 1–11.
[38] M. Morandini, L. Penserini, A. Perini, Towards goal-oriented development of self-adaptive systems, in: Proceedings of the 2008 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2008, Leipzig, Germany, May 12–13, 2008, 2008, pp. 9–16.
[39] V.E.S. Souza, A. Lapouchnian, W.N. Robinson, J. Mylopoulos, Awareness requirements for adaptive systems, in: Proceedings of the 2011 ICSE Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2011, Waikiki, Honolulu , HI, USA, May 23–24, 2011, 2011, pp. 60–69.
[40] V.E.S. Souza, A. Lapouchnian, K. Angelopoulos, J. Mylopoulos, Requirements–driven software evolution, Comput. Sci. 28 (4) (2013) 311–329.
[41] F. Dalpiaz, P. Giorgini, J. Mylopoulos, Adaptive socio-technical systems: a requirements-based approach, Requir. Eng. 18 (1) (2013) 1–24.
[42] J. Horkoff, E.S.K. Yu, Comparison and evaluation of goal-oriented satisfaction analysis techniques, Requir. Eng. 18 (3) (2013) 199–222.
[43] J. Horkoff, E.S.K. Yu, Analyzing goal models: different approaches and how to choose among them, in: W.C. Chu, W.E. Wong, M.J. Palakal, C. Hung (Eds.), Proceedings of the 2011 ACM Symposium on Applied Computing, SAC, TaiChung, Taiwan, March 21–24, 2011, ACM, 2011, pp. 675–682.

[44] P. Giorgini, F. Massacci, J. Mylopoulos, N. Zannone, Requirements engineering for trust management: model, methodology, and reasoning, Int. J. Inf. Secur. 5 (4) (2006) 257–274.
[45] G. Chatzikonstantinou, M. Athanasopoulos, K. Kontogiannis, Towards a goal driven task personalization specification framework, in: Proceedings of the IEEE Ninth World Congress on Services, SERVICES 2013, Santa Clara, CA, USA, June 28–July 3, 2013, IEEE Computer Society, 2013, pp. 180–184.
[46] G. Chatzikonstantinou, M. Athanasopoulos, K. Kontogiannis, Task specification and reasoning in dynamically altered contexts, in: M. Jarke, J. Mylopoulos, C. Quix, C. Rolland, Y. Manolopoulos, H. Mouratidis, J. Horkoff (Eds.), Proceedings of the 26th International Conference on Advanced Information Systems Engineering, CAiSE 2014, Thessaloniki, Greece, June 16–20, 2014, Lecture Notes in Computer Science, vol. 8484, Springer, 2014, pp. 625–639.
[47] R. Ali, F. Dalpiaz, P. Giorgini, Reasoning with contextual requirements: detecting inconsistency and conflicts, Inf.Softw. Technol. 55 (1) (2013) 35–57.
[48] G. Elahi, E.S.K. Yu, Comparing alternatives for analyzing requirements trade-offs – in the absence of numerical data, Inf. Softw. Technol. 54 (6) (2012) 517–530.
[49] G. Chatzikonstantinou, K. Kontogiannis, I. Attarian, A goal driven framework for software project data analytics, in: C. Salinesi, M.C. Norrie, O. Pastor (Eds.), Proceedings of the 25th International Conference on Advanced Information Systems Engineering, CAiSE 2013, Valencia, Spain, June 17–21, 2013, Lecture Notes in Computer Science, vol. 7908, Springer, 2013, pp. 546–561.
[50] A. Cailliau, A. van Lamsweerde, Assessing requirements-related risks through probabilistic goals and obstacles, Requir. Eng. 18 (2) (2013) 129–146.
[51] J. Horkoff, D. Barone, L. Jiang, E.S.K. Yu, D. Amyot, A. Borgida, J. Mylopoulos, Strategic business modeling: representation and reasoning, Softw. Syst. Model. 13 (3) (2014) 1015–1041.
[52] N. Bencomo, A. Belaggoun, Supporting decision-making for self-adaptive systems: from goal models to dynamic decision networks, in: J. Doerr, A.L. Opdahl (Eds.), Proceedings of the 19th International Working Conference on Requirements Engineering: Foundation for Software Quality, REFSQ 2013, Essen, Germany, April 8–11, 2013, Lecture Notes in Computer Science, vol. 7830, Springer, 2013, pp. 221–236.
[53] D. Tanabe, K. Uno, K. Akemine, T. Yoshikawa, H. Kaiya, M. Saeki, Supporting requirements change management in goal oriented analysis, in: Proceedings of the Requirements Engineering Conference, RE, IEEE Computer Society, 2008, pp. 3–12.
[54] W. Heaven, E. Letier, Simulating and optimising design decisions in quantitative goal models, in: Proceedings of the Requirements Engineering Conference, RE, IEEE, 2011, pp. 79–88.