

Supporting Early Decision-Making in the Presence of Uncertainty

Jennifer Horkoff*, Rick Salay†, Marsha Chechik†, and Alessio Di Sandro†

*Department of Information Engineering and Computer Science, University of Trento, Italy

horkoff@disi.unitn.it

†Department of Computer Science, University of Toronto, Toronto, Canada

{rsalay,chechik,adisandro}@cs.toronto.edu

Abstract—Requirements Engineering (RE) involves eliciting, understanding, and capturing system requirements, which naturally involves much uncertainty. During RE, analysts choose among alternative requirements, gradually narrowing down the system scope, and it is unlikely that all requirements uncertainties can be resolved before such decisions are made. There is a need for methods to support early requirements decision-making in the presence of uncertainty. We address this need by describing a novel technique for early decision-making and tradeoff analysis using goal models with uncertainty. The technique analyzes goal satisfaction over sets of models that can result from resolving uncertainty. Users make choices over possible analysis results, allowing our tool to find critical uncertainty reductions which must be resolved. An iterative methodology guides the resolution of uncertainties necessary to achieve desired levels of goal satisfaction, supporting trade-off analysis in the presence of uncertainty.

I. INTRODUCTION

During the process of eliciting requirements, it is common to encounter uncertainties, including gaps in domain knowledge, disagreements between stakeholders, or uncertainty over requirement details. Early requirements elicitation typically uncovers a large space of alternative requirements and conflicting needs, making it necessary to support decision-making over alternatives in order to find acceptable trade-offs between conflicting requirements. Given the wide scope of early system analysis, it may not be feasible to resolve all uncertainty before decision-making occurs. Ignoring uncertainty forces implicit and premature decisions over the space of early requirements. Thus, methods and tools to (a) support early decision-making and trade-off analysis and (b) do so in the presence of uncertainty are needed. To support these needs, we make use of existing, established Requirements Engineering (RE) techniques: goal modeling and associated analysis [11], and the *MAVO* framework for formally capturing and reasoning over model uncertainty [20].

Goal models (e.g., NFR [3], KAOS [4], *i** [24]) have been widely studied in RE as a means to explicate system goals and high-level requirements. The ability to facilitate the analysis of alternative requirements makes goal models especially powerful for RE. For example, “what if?” analysis [3], [7], [11] uses the effects on and relationships between goals to determine the (degrees of) goal satisfaction resulting from a selection of alternative requirements. Such analysis helps modelers select

a viable set of requirements, making acceptable trade-offs between user needs.

Current goal model analysis procedures evaluate the effects of alternative requirements on user goals but do not account for model uncertainties discovered at design time. Much focus has been placed on capturing and reasoning over vague goals, using the softgoal concept to capture goals without clear-cut criteria for success. However, in addition to being vague, goals (and associated relationships) can be uncertain, for example, we might be uncertain about the presence or absence of an element. Ignoring uncertainty, i.e., treating the element as present, may lead us to eliminate viable alternatives or accept alternatives which do not sufficiently satisfy key goals.

The *MAVO* framework [20] has been introduced in order to explicitly and formally capture uncertainty in the contents of models. *MAVO* is language-independent, and our previous work, [18], [19], allowed us to record model changes, including change rationale, checking that these changes reduce uncertainty, and to propagate uncertainty reductions across traceability links between different models. In this work, we take the semantics of the target language (*i**) into account, allowing us to determine the satisfaction of goals, evaluate alternative solutions, make tradeoffs and find satisfactory solutions – even in the presence of model uncertainty.

Specifically, we facilitate analysis and decision-making in the presence of uncertainty by integrating goal model analysis with uncertainty expressed using *MAVO*. We describe a semi-automated method which finds sets of possible goal satisfaction analysis results, accounting for all of the ways in which model uncertainty may be resolved. Our approach determines which uncertainties must be resolved to achieve desired levels of goal satisfaction, allowing for targeted domain elicitation. We provide a methodology which supports an iterative reduction of uncertainty, moving towards “concrete” models in which uncertainties are resolved and desired levels of goal satisfaction are achieved. Thus, we allow users to make decisions over the space of early requirements even in the presence of uncertainty.

A variety of goal modeling languages and analysis procedures have been introduced (e.g., [3], [24], [4], [7]). In order to make our contribution concrete, we selected a particular language, *i** [24], and a particular method for qualitative analysis [11]. However, our approach can be generalized to

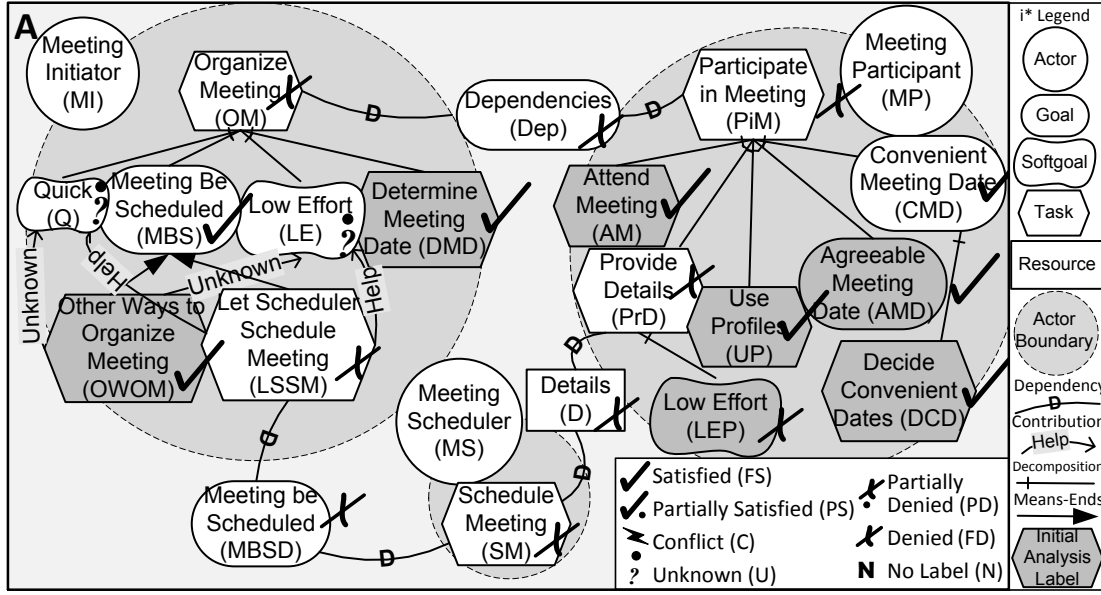


Fig. 1. A goal model for the Meeting Scheduler Example with analysis of the OWOM alternative.

any goal-oriented language with formal semantics. We do not model the intrinsic, run-time uncertainty of the environment, but the design-time uncertainty of the modeler, which can be eventually resolved. This type of uncertainty is best captured *possibilistically* rather than *probabilistically* as the statistics needed for the latter is typically not available at design time.

The rest of this paper is organized as follows: Sec. II describes a motivating scenario and notation background using the Meeting Scheduler example. We provide the necessary formal background and notation in Sec. III. Sec. IV describes how to compute and interpret analysis labels over uncertain models. Sec. V describes a methodology for analysis of uncertain models, including finding suggested uncertainty refinements given analysis choices. Sec. VI describes our tool support and experiences. We compare our approach with related work in Sec. VII. Sec. VIII summarizes the paper and discusses future work.

II. BACKGROUND AND MOTIVATING SCENARIO

We motivate our approach using a well-known meeting scheduler example¹. In this scenario, a Meeting Initiator has a variety of ways to schedule meetings, including using a (semi)automated Meeting Scheduler, and must choose between scheduling alternatives. However, in the early analysis of this scenario, we are uncertain about several aspects of the domain. What are some other ways for the Meeting Initiator to schedule a meeting? What information is needed from the Meeting Participant? Who determines possible meeting dates? Despite these uncertainties, we need to make decisions over the space of scheduling alternatives. We ask: Can we evaluate whether or not an alternative is viable even in the presence of uncertainty? and What uncertainties must be resolved in order to facilitate our decision-making process?

¹Parts of this scenario have appeared in previous work, e.g., [24], [18].

i* Language. Fig. 1 depicts an i* model, model A for the meeting scheduler capturing a simplistic view of the goals, actors and dependencies for this system. It does not yet explicitly capture our uncertainties concerning the scenario. The social aspect of i* is represented by *actors*, e.g., Meeting Initiator (MI for short, see Fig. 1 for shorthands), and Meeting Participant (MP). Actors depend upon each other for the accomplishment of *tasks*, e.g., Provide Details (PrD), and Determine Meeting Date (DMD), the provision of *resources*, e.g., Details (D), the satisfaction of *goals*, e.g., Meeting be Scheduled (MBS) and *softgoals*, goals without clear-cut satisfaction criteria, e.g., Low Effort (LE). Together, these are called *intentions*. Actor “boundaries” contain the intentions explicitly desired by these actors, e.g., MP wants to achieve Participate in Meeting (PiM).

The interrelationships between the intentions are depicted via three types of links: *decomposition*, representing intentions necessary to accomplish a task, e.g., to achieve PM the MP actor must satisfy AM, PrD, and satisfy the other three decomposition intentions; *means-end*, representing alternative tasks which can accomplish a goal, e.g., Other Ways to Organize Meeting (OWOM) or Let Scheduler Schedule Meeting (LSSM) will satisfy MBS; and *contribution*, showing the effects of softgoals, goals, and tasks on softgoals, e.g., LSSM helps Quick (Q). Positive/negative contributions representing evidence sufficient to satisfy/deny a softgoal are represented by *Make/Break* links, respectively. Weaker contributions are represented by *Help/Hurt* links. *Unknown* contribution links represent the presence of evidence with yet unknown polarity.

Goal Model Analysis. Model A allows the analyst to capture requirement options, and existing analysis procedures, e.g., [11], allow her to evaluate such alternatives. The analysis procedure starts with a question of the form “How effective is an alternative w.r.t. the goals in the model?”. In our example,

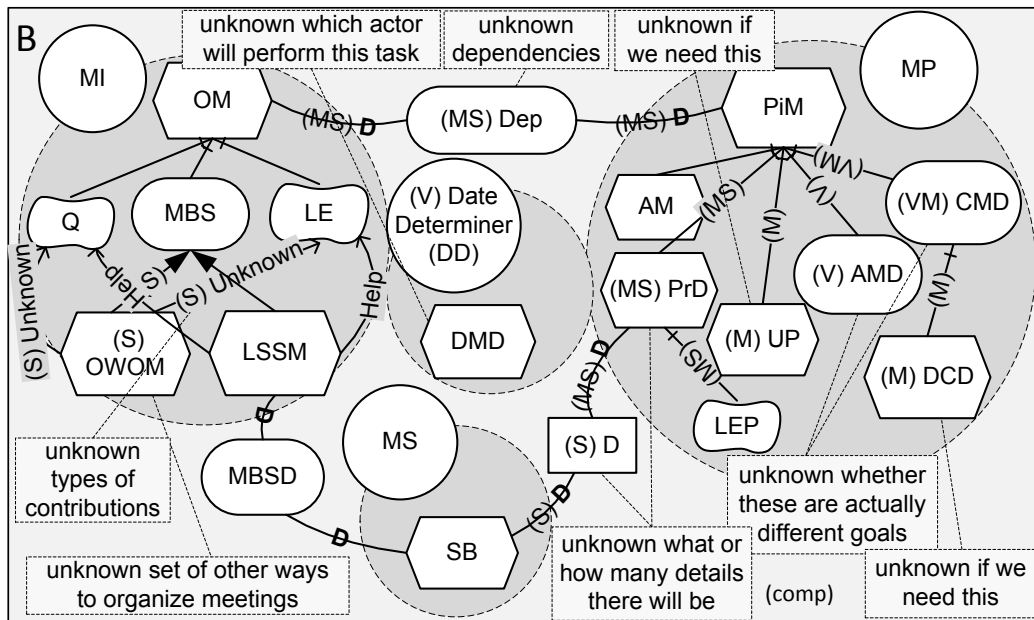


Fig. 2. A goal model for the Meeting Scheduler Example annotated with uncertainties informally (dashed notes) and formally (annotations).

there are two alternative ways to satisfy MBS: LSSM and OWOM. If we want to find the effects of selecting OWOM, we place *initial labels*, satisfying the task OWOM and denying LSSM in model **A**. For completeness, other initial labels, reflecting a more detailed analysis question, are chosen (shaded intentions in model **A**).

Initial labels are iteratively propagated through links using propagation rules for decomposition, means-ends, dependency, and contribution links, resulting in labels for other model elements, representing the cumulative level of positive or negative evidence. The propagation continues until no new labels can be produced. In our example, initial labels for OWOM and LSSM are propagated through the means-end (OR) link. By this rule, MBS is (fully) satisfied (*FS* or \checkmark , see legend in Fig. 1). Similarly, incoming labels for PiM in MP are propagated through the (AND) decomposition, this time selecting the minimum label, fully denied (*FD*).

Since there is a dependency relationship between **D** and **PrD**, a denied value for the latter implies a denied value for the former. Propagation through contribution links uses rules adopted from [3]. In order to capture partial evidence propagated through contribution links, goal model analysis captures *partial* positive or negative evidence towards goal satisfaction, via *partially satisfied (PS)* and *partially denied (PD)* labels, respectively. For example, the *FD* label for **LSSM** propagates a *PD* label through the *Help* link to **Q** and **LE**. Similarly, the *Unknown* label (*U*) represents the presence of evidence which could be positive or negative. For example, the *FS* value for **OWOM** propagates *U* through the *Unknown* links. In this case, *U* represents a missing piece of evidence that should be addressed by users.

As softgoals often have multiple incoming contribution links, multiple labels might apply at once, e.g., Q is both U

and *PD*. Such cases can be combined into a single label for subsequent propagation; possibly making use of the *Conflict* label, (*C*) indicating the presence of positive and negative evidence of roughly the same strength. In this paper, we combine evidence automatically: we select *U* when present, or *C* when evidence is both positive and negative, and otherwise select the minimum (most pessimistic) label. When an intention is the target of both a dependency link and another type of link (decomposition, means-ends, contribution), the propagation results from both types of links are combined via AND (taking the minimum label). The *N* (no label) symbol is used to explicitly indicate the absence of other labels.

Domain experts and analysts can determine whether the analysis results are sufficient or *viable*. In our example, the analysis indicates that the MI actor is unable to satisfy OM (*FD*), so OWOM is a non-viable alternative. In this case, the modeler would analyze other model alternatives (LSSM) or perform additional elicitation to find further alternatives.

Model Uncertainty. As mentioned, the scenario involves several points of uncertainty. We summarize this uncertainty using free-form notes in model **B** in Fig. 2. We can use the *MAVO* method described in [18] to formally express the uncertainties in our model, shown in the same figure.

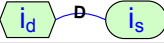
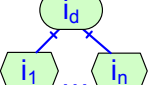
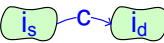
MAVO uses four types of annotations, each adding support for a different type of uncertainty in a model. The *May annotation* allows us to express uncertainty about the presence of an element in a model. Annotating it with M indicates that it “may exist”; otherwise, it “must exist”. E.g., in model **B**, we are unsure if the MP’s task DCD is present in the model. Uncertainty is reduced by removing the *May* annotation or eliminating the element altogether.

The *Abs annotation* allows a modeler to express uncertainty about the number of elements in the model. An S represents

TABLE I
ANALYSIS QUESTIONS OVER UNCERTAIN GOAL MODELS.

Q1	What are the analysis results given a particular analysis alternative in the goal model, considering model uncertainties?
Q2	Can viable choices be made over the set of results from Q1 ?
Q3	Can viable choices be achieved simultaneously? If so, find example uncertainty resolutions that achieve choices.
Q4	Given choices, what uncertainty reductions are forced? How can we target elicitation?
Q5	Are uncertainty reductions suggested by Q3 reasonable given the domain? If not, find further uncertainty reductions.

TABLE II
SELECTED PROPAGATION RULES.

Link Type		Original Rule	Example MAVO Rule
Dependency		$(v \in V) v(i_s) \Rightarrow v(i_d)$	$\forall t : \text{Task}, g : \text{Goal} \cdot (\text{Dep}(t, g) \wedge \text{FS}(g)) \Rightarrow \text{FS}(t)$
Decomposition		$(\bigwedge_{j=1}^n \text{FS}(i_j)) \Rightarrow \text{FS}(i_d)$... $(\bigvee_{j=1}^n \text{FD}(i_j)) \Rightarrow \text{FD}(i_d)$	$\forall t_1 \dots t_n : \text{Task}, g : \text{Goal} \cdot (\text{Decomp}(t_1 \dots t_n, g) \wedge \bigwedge_{j=1}^n \text{FS}(t_j)) \Rightarrow \text{FS}(g)$... $\forall t_1 \dots t_n : \text{Task}, g : \text{Goal} \cdot (\text{Decomp}(t_1 \dots t_n, g) \wedge \bigvee_{j=1}^n \text{FD}(t_j)) \Rightarrow \text{FD}(g)$
Contribution		$(c = \text{Make}) \text{FS}(i_s) \Rightarrow \text{FS}(i_d)$ $(c = \text{Help}) \text{FS}(i_s) \Rightarrow \text{PS}(i_d)$... $(c = \text{Unk}, v \in V) v(i_s) \Rightarrow U(i_d)$ $(c \in \{\text{Make}, \text{Help} \dots\}) U(i_s) \Rightarrow U(i_d)$	$\forall t : \text{Task}, g : \text{Goal} \cdot (\text{Make}(t, g) \wedge \text{FS}(t)) \Rightarrow \text{FS}(g)$ $\forall t : \text{Task}, g : \text{Goal} \cdot (\text{Help}(t, g) \wedge \text{PS}(t)) \Rightarrow \text{PS}(g)$... $\forall t : \text{Task}, g : \text{Goal} \cdot (\text{Unknown}(t, g) \wedge v(t)) \Rightarrow U(g)$ $\forall t : \text{Task}, g : \text{Goal} \cdot (c(t, g) \wedge U(t)) \Rightarrow U(g)$

a “set”; omitting the annotation means that the element is “particular”. E.g., the s-annotated operation OWOM in actor MI in model **B** indicates that there are some such operations but they are as yet unknown. Uncertainty is reduced by elaborating the content of s elements into a set of (possibly annotated) elements.

The *Var annotation* allows a modeler to express uncertainty about the distinctness of individual elements in the model by annotating an element as a “variable” (v); omitting the annotation means that the element is a “constant”. Uncertainty is reduced by merging variable elements with constants or other variables. E.g., while it is known that task DMD should be in model **B**, it is not yet clear which actor should perform it. Yet it must be assigned to *some* actor for *i** well-formedness. We thus put the task in a v-annotated actor, allowing it to merge with other actors and eventually be assigned to a “real” (constant) actor.

The *OW annotation* allows a modeler to explicitly state whether her model is incomplete (i.e., can be extended) (INC) or complete (the default). Here the annotation is at the level of the entire model rather than individual elements. Our model **B** is complete.

Analysis with Uncertainty. We want to be able to make early decisions over alternative requirements, even in the presence of uncertainty. Given our explicit recording of uncertainties in the modeling process, we want to know how their presence affects analysis results (Fig. 1), and how such uncertainties can be used to guide analysis and elicitation. Specifically, (**Q1**) how do the uncertainties in model **B** affect analysis results in model **A**? For example, if we are not sure about the presence of PrD (labelled with FD), perhaps the MP actor may be able to satisfy PiM (i.e., this task could be FS). If the MP actor can satisfy PiM, then the satisfaction of OM in the MI actor may actually be unknown (U). We wish to see all of the possible labels given possible uncertainty reductions in the model.

Given the answer to **Q1**, we wish to answer additional questions as listed in Table I. The rest of this paper describes a

method which takes uncertainty into account when analyzing goal models, allowing us to answer these questions.

III. PRELIMINARIES

In this section, we give the necessary formal background on goal model analysis and uncertainty via the *MAVO* framework.

A. Goal Model Analysis

The *forward propagation algorithm* [11] begins by applying initial labels corresponding to the analysis question to the model, queuing the labels. All labels in the queue are propagated through outgoing links, with multiple incoming labels resolved using human judgment as in [11] or, in our case, rules as described in Sec. II. Propagation ends when the label queue is empty (see [11] for a discussion of termination).

The above procedure uses six predicates over model elements corresponding to the six analysis labels ($V = \{\text{FS}(), \text{PS}(), \text{C}(), \text{U}(), \text{PD}(), \text{FD}()\}$), where the predicate holds if the label applies.

Initial Analysis Labels. A selected subset of intentions within a goal model are assigned initial analysis labels, e.g., FS(OWOM) and FD(LSSM) in Fig. 1.

Propagation Constraints. The procedure provides rules in order to facilitate a standard propagation of labels through goal model relationships (links). The Original Rule column of Table II presents selected propagation rules for the dependency, decomposition, and contribution relationships depicted in the Link Type columns. Dependency links propagate evidence unchanged, decomposition combines evidence using “AND” (minimum), while means-ends combines evidence using “OR” (maximum). The following order between labels is used: $(\text{FS} \geq \text{PS} \geq \text{U} \geq \text{C} \geq \text{PD} \geq \text{FD})$. Furthermore, $\text{FS}(i) \Rightarrow \text{PS}(i)$ and $\text{FD}(i) \Rightarrow \text{PD}(i)$, where i is an element in the model.

Means-end rules (omitted from Table II) are derived from decomposition rules by replacing \bigwedge with \bigvee and vice versa. Contribution links can weaken or negate evidence, or make evidence unknown, depending on the incoming label, $(v(i_s))$,

and the type of contribution link. For example, the second to last row of the table says that for an *Unknown* (Unk) contribution link, regardless of the incoming analysis label ($v(i_s)$), the *U* label is propagated ($U(i_d)$ holds).

B. Capturing Uncertainty with MAVO

MAVO [20] is an approach for adding uncertainty information as annotations in models. Although MAVO can be applied to any type of goal model, this exposition uses i^* models. A MAVO i^* model contains annotations on the model elements that together represent a set of different possible *concrete* (i.e., ordinary) i^* models that would resolve the uncertainty:

Definition 1 (MAVO i^* model): A MAVO i^* model M consists of an i^* base model, denoted $bs(M)$, and a set of annotations. $[M]$ denotes the set of i^* models called the *concretizations* of M . M is called *consistent* iff it allows at least one concretization, i.e., $[M] \neq \emptyset$.

Formalizing MAVO. We now describe how MAVO annotations formally characterize a set of i^* models – it is the foundation of the method for encoding and automatically reasoning with MAVO models. The i^* metamodel represents the set of well-formed i^* models and can be expressed as a First Order Logic (FOL) theory.

Definition 2 (i^* metamodel): The i^* metamodel is an FOL theory $T = \langle \Sigma, \Phi \rangle$, where Σ is the *signature* with sorts representing the entity types (e.g., Actor, Intention, Task) and predicates representing the relations (e.g., Task decomposes Goal); Φ is a set of sentences representing the i^* well-formedness constraints (e.g., multiplicities, dependencies between actors). The models that conform to T are the finite FO Σ -structures that satisfy Φ according to the usual FO satisfaction relation. We denote this set of models by $Mod(T)$.

Like the i^* metamodel, a MAVO i^* model also represents a set of models and thus can also be expressed as an FOL theory. Specifically, for a MAVO i^* model M , we construct a theory $FO(M)$ s.t. $Mod(FO(M)) = [M]$. We illustrate the construction of $FO(\mathbf{B})$ for MAVO i^* model \mathbf{B} in Fig. 2.

(1) Let $\mathbf{G} = bs(\mathbf{B})$ be the base model of model \mathbf{B} . We define a new MAVO i^* model \mathbf{B}_G with \mathbf{G} as its base model and its sole concretization, i.e., $bs(\mathbf{B}_G) = \mathbf{G}$ and $[\mathbf{B}_G] = \{\mathbf{G}\}$. We call \mathbf{B}_G the *ground* model of \mathbf{B} .

(2) To construct the FOL encoding of \mathbf{B}_G , $FO(\mathbf{B}_G)$, we extend the i^* metamodel to include a unary predicate for each

element in \mathbf{G} and a binary predicate for each relation instance between the elements in \mathbf{G} . Then, we add constraints to ensure that the only first order structure that satisfies the resulting theory is \mathbf{G} itself:

$$FO(\mathbf{B}_G) = \langle \Sigma \cup \Sigma_G, \Phi \cup \Phi_G \rangle \text{ (see Def. 2)}$$

where Σ_G and Φ_G are model \mathbf{G} -specific predicates and constraints, defined in Fig. 3. We refer to Σ_G and Φ_G as the *MAVO predicates* and *constraints*, respectively.

The FO structures that satisfy $FO(\mathbf{B}_G)$ are those i^* models that satisfy the constraint set Φ_G in Fig. 3. Assume that K is one such i^* model. The MAVO constraint *Complete* ensures that K contains no more elements or relation instances than \mathbf{G} . Now consider the actor \mathbf{MP} in \mathbf{G} . *Exists_{MP}* says that K contains at least one actor called \mathbf{MP} , *Unique_{MP}* – that it contains no more than one actor called \mathbf{MP} , and the clauses *Distinct_{MP-*}* – that the actor called \mathbf{MP} is different from all the other actors. Similar MAVO constraints are given for all other elements and relation instances in \mathbf{G} . These constraints ensure that $FO(\mathbf{B}_G)$ has exactly one concretization and thus $K = \mathbf{G}$.

(3) We construct $FO(\mathbf{B})$ from $FO(\mathbf{B}_G)$ by removing constraints corresponding to the annotations in \mathbf{B} . This constraint relaxation allows more concretizations and thus represents increasing uncertainty. If \mathbf{B} were incomplete, we could express this fact by removing the *Complete* clause from Φ_G , thereby allowing concretizations to be those i^* models that extend \mathbf{G} . Similarly, expressing the effect of the *M*, *S* and *V* annotations for an element E corresponds to relaxing Φ_G by removing *Exists_E*, *Unique_E* and *Distinct_{E-*}* clauses, respectively. E.g., removing the *Distinct_{DD-*}* clauses expresses the *V* annotation on the actor \mathbf{DD} (i.e., it may or may not be distinct from another actor).

IV. ANALYSIS OF GOAL MODELS WITH MAVO UNCERTAINTY

In this section, we describe our key contribution, the combination of goal model analysis with MAVO uncertainty annotations, “lifting” goal model analysis from conventional to uncertain goal models.

A. Correctness Condition for MAVO Goal Model Analysis

A MAVO goal model represents a set of goal models (i.e., its concretizations), whereas goal model analysis applies only to a single goal model. Thus, correct analysis results over a MAVO goal model should aggregate the analysis results over all of its concretizations:

Definition 3 (Correct Labeling): Let M be a MAVO goal model. M is *correctly labelled* iff for all intentions I in M , label v is assigned to I iff $\exists m \in [M], i \in I_m \cdot v(i)$, where I_m is the set of intentional elements mapped to I in m .

Thus, a label gets applied to an intention when there is a concretization that has that label on the intention. Model \mathbf{C} in Fig. 4 shows a correct analysis labeling of our example model from Fig. 2. Here we use a shorthand notation for sets of possible labels, e.g., $\{FS, FD, N\}$ in \mathbf{PiM} .

Σ_G has unary predicates $MP(\text{Actor})$, $AM(\text{Task})$, ..., and binary predicates $AMinMP(\text{Task}, \text{Actor})$, ...
 Φ_G contains the following sentences:
(Complete) $(\forall x : \text{Actor} \cdot MP(x) \vee MS(x) \vee DD(x) \vee \dots) \wedge (\forall y : \text{Task}, x : \text{Actor} \cdot in(y, x) \Rightarrow (AMinMP(y, x) \vee \dots)) \wedge \dots$
 MP :
(Exists_{MP}) $\exists x : \text{Actor} \cdot MP(x)$
(Unique_{MP}) $\forall x, x' : \text{Actor} \cdot MP(x) \wedge MP(x') \Rightarrow x = x'$
(Distinct_{MP-MS}) $\forall x : \text{Actor} \cdot MP(x) \Rightarrow \neg MS(x)$
(Distinct_{MP-DD}) $\forall x : \text{Actor} \cdot MP(x) \Rightarrow \neg DD(x)$
(Distinct_{MP-MI}) $\forall x : \text{Actor} \cdot MP(x) \Rightarrow \neg MI(x)$
 similarly for all other element and relation predicates

Fig. 3. The MAVO predicates and constraints for \mathbf{G} .

C. Determining Possible Analysis Labels

We can use our encoding to find analysis results given a particular alternative (set of initial labels) and model uncertainty, answering **Q1**. By Def. 3, a label can hold for an intention in a *MAVO* model iff there exists a concretization containing a corresponding concrete intention with that label. The following proposition shows we can use the extended encoding $FO^e(M)$ to check this condition.

Proposition 1 (Analysis Labeling): Let M be a *MAVO* i* model with an analysis label set assigned to each intention. M is *correctly labelled according to Def. 3* iff for all intentions I in M , label v is assigned to I iff $\Phi_M^e \cup (\exists i : \text{Intention} \cdot I(i) \wedge v(i))$ is satisfiable.

For each possible labeling of intention I with label v , this checks whether the labeling is consistent with Φ_M^e . If so, there must exist a concretization with this labeling. Checking this for all labellings ensures that condition in Sec. 3 is met. For example, for model **C**, to check whether PiM should have label *FS*, we ask whether $\Phi_C^e \cup (\exists i : \text{Intention} \cdot \text{PiM}(i) \wedge \text{FS}(i))$ is satisfiable. It is satisfiable since there is a concretization (see Sec. 3) in which PiM gets the propagated label *FS*. Performing this check for every possible combination of model element and analysis label allows us to find results across the entire model, as we do in model **C**.

V. REASONING METHODOLOGY

In our motivating scenario, we raised five analysis questions (Table I). We have shown how to answer **Q1**, finding possible analysis labels given uncertainty, in Sec. IV-C. In this section, we describe how the combination of goal model analysis and *MAVO* uncertainty can be used to answer **Q2-Q5**, providing an analysis methodology guiding users in asking the questions and interpreting the answers. We summarize our methodology in Fig. 5. Steps which require manual judgment are indicated by the presence of an actor.

Answering Q2: Making Choices. Given the set of analysis results over the uncertain model, users can select a subset of the results that they consider acceptable. We call this subset *choices*. In our example, the fully satisfied (✓) value is the most desired for PiM in Model **C**, while unknown (?) value is the most desired for OM (circled in model **C**).

Answering Q3: Checking Simultaneous Achievability of Choices. Once analysis choices have been selected, we can use the extended encoding to check if such choices are simultaneously achievable in one or more concretizations. This is done by encoding the desired label choices as constraints to require that concretizations simultaneously achieve these labels. For example, for model **C**, our choices consist of *U* for OM and *FS* for PiM and so we define the following constraints: (1) $\forall i : \text{Intention} \cdot \text{OM}(i) \Rightarrow U(i)$ and (2) $\forall i : \text{Intention} \cdot \text{PiM}(i) \Rightarrow \text{FS}(i)$. We call the set of all such constraints Φ_c and use it to check the existence of a concretization satisfying these choices.

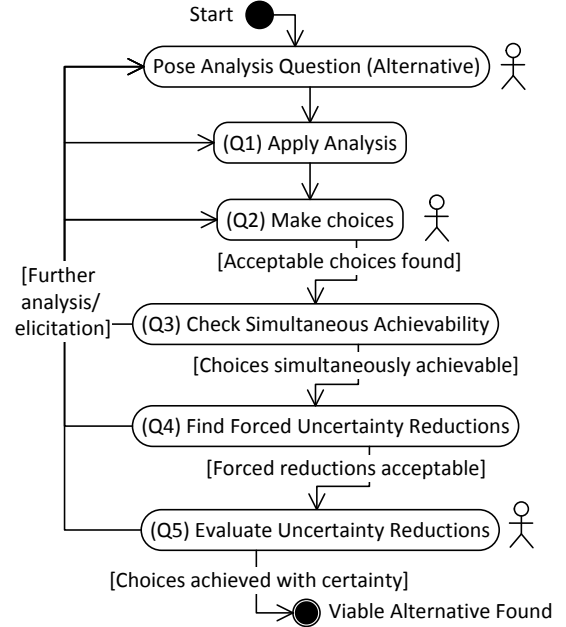


Fig. 5. Goal model analysis methodology with actors indicating manual tasks.

Proposition 2 (Choice Achievability Check): Given a *MAVO* model M and a constraint set Φ_c encoding a set of choices, this set of choices is simultaneously achievable iff $\Phi_M^e \cup \Phi_c$ is satisfiable.

For model **C**, checking the constraints $\Phi_C^e \cup \Phi_c$ shows that our label choices (*U* for OM and *FS* for PiM) are simultaneously achievable.

Answering Q4: Finding Forced Uncertainty Reductions.

Once the users check their choices for simultaneous achievability, we can use the approach defined in [21] to determine which uncertainties are *forced*, i.e., those which must be resolved in all concretizations in order to achieve analysis choices. In our example, the forced uncertainty reductions are those related to PrD. In this case, the decomposition link must be removed for our choices to be achieved. We indicate this in model **C** by making the link and the corresponding M uncertainty annotation red and bold. Such information allows users to target elicitation on a subset of uncertainties in the model. In our example, users judge this forced resolution to be acceptable, and continue with the analysis process.

Answering Q5: Evaluating Uncertainty Reductions. As an outcome of checking for simultaneous achievability in **Q3**, we produce a concrete model with uncertainties resolved such that choice values are achieved. Model **D** in Fig. 6 is a particular concretization for our example. It shows the meeting scheduler example after a set of uncertainty reductions (in bold) which allow achievement of the chosen analysis results. For example, the M-annotated task PrD and the uncertainty concerning UP have been removed when compared to model **C**.

In our example, users judge some, but not all, of the suggested uncertainty reductions in **D** to be consistent with the domain. For example, it is deemed acceptable that the MP

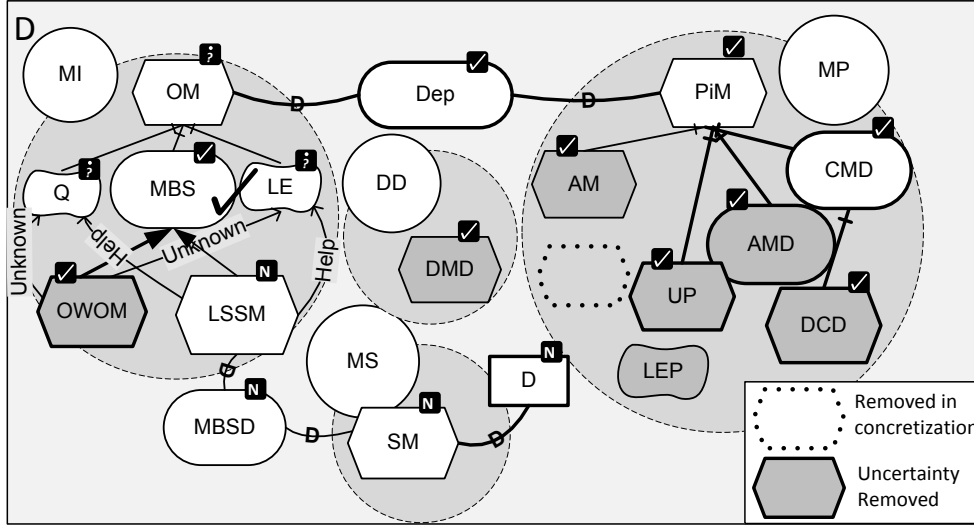


Fig. 6. One possible concretization achieving analysis choices in Model C in Fig. 4 (result of Q3).

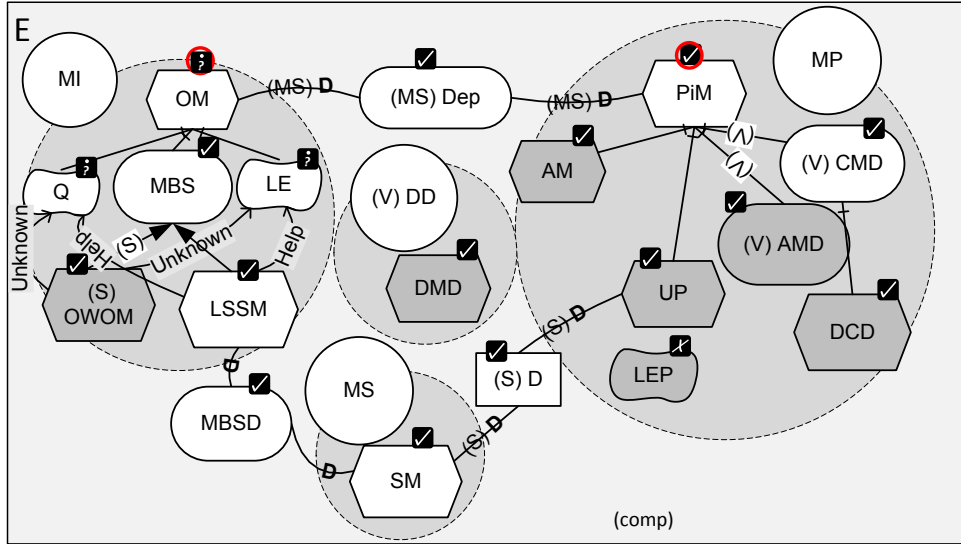


Fig. 7. Chosen and forced uncertainty reductions achieving analysis choices (results of Q5, reapplying Q1).

actor will not perform the task PrD (removed in D); instead, she can keep track of participant details using the UP task. In another example, it is deemed unacceptable for the dependency link from resource D to be removed from the model altogether. This link should now depend on UP instead of the removed PrD. Further uncertainties are accepted or rejected and the model is updated to reflect these decisions (uncertainties are removed, model elements are changed, analysis (Q1) is reapplied).

When iterating over analysis results, different constraints can be added in order to determine if concretizations satisfying them exist, producing one such concretization. For example, when finding concretizations for model C, we may want to specify that actors DD and MP cannot be merged together, expressed by the property Ψ . Checking $\Phi_C^e \cup \Phi_C \cup \{\Psi\}$ produces a concretization in which Ψ holds.

An iterative process of finding concrete models, constraining the problem and accepting or rejecting suggested uncertainty reductions will lead, in the optimal case, to a model in which choices are achieved with certainty and a viable set of alternative requirements is found. Otherwise, the model may not contain viable alternatives, prompting further elicitation. Alternative outcomes are discussed in Sec. VI.

In our running example, iterative analysis directed us to reduce targeted uncertainty in the model and the domain such that we could achieve our analysis choices with certainty. We show the final model resulting from this process, model E, in Fig. 7. In the model without uncertainty, A, we rejected the alternative that included OWOM. Yet considering model uncertainty, as we did in model C, made this alternative viable. We have determined the viability of this alternative without the need to remove all model uncertainty. Applying

this methodology iteratively can enable exploring the viability of other alternatives in the model (e.g., satisfying LMSM), allowing for a final selection between alternative requirements.

VI. EXPERIENCE AND DISCUSSION

Implementation. We have implemented the automatable questions in our methodology (**Q1**, **Q3**, **Q4**), building on top of Model Management Tool Framework (MMTF) [17]. Our implementation automatically converts an i^* model with uncertainty annotation $\langle \Sigma_M, \Phi_M \rangle$, together with the i^* metamodel $\langle \Sigma, \Phi \rangle$, analysis constraints $\langle \Phi_M \cup \Phi_P \rangle$, and initial labels $\langle \Phi_I \rangle$, into their corresponding FOL encoding.

Our implementation uses an SMT solver, specifically, Z3 version 4.3.1². It answers **Q1** by iteratively calling the solver, checking each possible label for each intention. We answer **Q3** by adding the user choices to the encoding $\langle \Phi_C \rangle$ and checking for satisfiability. The result is a suggested concrete model. **Q4** is answered using the implementation outlined in [21], integrated into the MMTF Framework.

Experience. Answering questions **Q1**, **Q3**, and **Q4** for the illustrated alternative in the Meeting Scheduler example took 20.72, 0.18, and 7.31 seconds, respectively, using a PC with Intel Core i7-2600 3.40 GHz x 4 cores and 8 GB RAM. We have applied our approach to three additional alternatives for the Meeting Scheduler and to a larger example, Info. A description of these models, analysis results, SMT encodings, and running times can be found online³.

Alternative Analysis Paths. Our scheduler example described a successful analysis path through our methodology, leading to a viable analysis alternative with certainty.

However, there may be less successful outcomes which correspond to back-edges in Fig. 5. We claim that even those scenarios that do not directly lead to a viable alternative can help eliminate alternative sets of requirements, reduce uncertainty, and refine the model.

In one scenario, the set of possible analysis values may not yield viable choices, (i.e., **Q2** is answered negatively). For example, if the analysis in model **D** in Fig. 6 yielded only negative values (PD or FD) for both PiM and OM, users could evaluate further alternatives in the model (e.g., LSSM), or perform further elicitation to discover other requirement alternatives.

In another scenario, if users are able to find sufficient choice values over uncertain results but these values are not simultaneously achievable (**Q3** is answered negatively), users must revise their choices, or evaluate further alternatives. Similarly, if critical uncertainty reductions (**Q4**) are not consistent with the domain, options include relaxing choice values, finding an alternative set of choice values making different trade-offs, or evaluating further alternatives.

If the reductions in uncertainty evaluated as part of **Q5** are not consistent with domain knowledge, users can articulate the reasons for such inconsistencies and either revise the model

and *MAVO* uncertainties to reflect this understanding, or add more specific constraints to the SMT encoding (see Sec. V for an example).

Backward Reasoning. In this paper, we have illustrated our technique using *forward analysis*, i.e., placing labels on leaf elements and asking “what if?” questions. It is also possible to conduct *backward analysis* by placing analysis labels on root elements and asking if it is possible to achieve them and if so, how? [8]. Our current implementation can support backward analysis by removing the constraint which assigns the N label to all leaf intentions not explicitly assigned an initial label. In this case, the set of possible analysis labels presented when answering **Q1** includes possible labels which satisfy backward constraints and labels made possible via uncertainty. Future work will focus on making the sets of possible labels comprehensible to the user, both in forward and backward analysis, e.g., understanding how they can be used to visualize consistent solutions over the whole model.

VII. RELATED WORK

Uncertainty in RE. Several approaches consider uncertainty in requirements, often as part of an overall strategy for managing uncertainty in software development (e.g., [12], [16]). The *MAVO* approach adopted in this work takes a different view on uncertainty in RE, allowing users to capture uncertainty on top of their preferred RE modeling language [20].

Much of the investigation of uncertainty in RE concerns adaptive systems. Such systems aim to respond to run-time uncertainty by specifying functional adaptations as part of RE (see [22] for overview). *MAVO* is aimed to represent uncertainty in the *content*, or structure, of requirements models arising as part of elicitation and design, ideally resolved as part of the software development process, and does not explicitly handle run-time or environmental uncertainty.

Uncertainties in software development are often considered as part of risk management, (e.g., [13]). Although certain risk factors (e.g., an unknown budget) may motivate the presence of model uncertainty, *MAVO* is not explicitly intended to capture risks.

Partial Modeling. May uncertainty in *MAVO* is related to various modal extensions to *behavioural* modeling formalisms. For example, Modal Transition Systems (MTSs) [14] allow introduction of uncertainty about transitions on a given event, whereas Disjunctive Modal Transition Systems (DMTSs) [15] add a constraint that at least one of the possible transitions must be taken in the refinement. Concretizations of these models are Labelled Transition Systems (LTSs). MTSs and DMTSs have been used to capture some forms of uncertainty in early design models [23]. The *MAVO* approach allows specification of more uncertainty types, although it is applicable only to *structural* models.

Goal Model Analysis. Several goal model analysis procedures have been proposed, facilitating decision-making in RE (see [10] for a survey). In this paper, we have focused on qualitative satisfaction analysis. Quantitative approaches (e.g., [7]) are

²<http://z3.codeplex.com/>

³<http://www.cs.utoronto.ca/~jenhork/AnalyzingUncertainGM>

appropriate for later, detailed requirements analysis, where reliable metrics can be found, and are less applicable to possibilistic, early uncertainties. Other approaches consider uncertainty in goal models through varying contexts (e.g., [1]) or by analyzing risk (e.g., [2]). These approaches capture uncertainty inherent in the domain, for example, a change in user context or failure of a component, while our approach focuses on evolving uncertainty arising as part of requirements elicitation, ideally resolved before implementation.

Further approaches have applied SAT solving to goal models, using interpolation to automatically refine operational goals [5]. This technique is not aimed for decision-making or trade-off analysis but to automatically find goal operationalizations which satisfy safety and liveness goals.

Related work has explored the application of search based (SB) techniques to the problem of requirement selection, finding non-dominated sets of optimal requirements [25]. Yet application of SB techniques requires quantitative utility functions mapping all alternatives to all objectives, which are difficult to elicit with accuracy in early RE. Our approach focuses on qualitative analysis over goal models with uncertainty which are more feasible for early decision making.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we have motivated the need for support of early decision-making in the presence of uncertainty. We provide such support via an explicit consideration of uncertainty as part of goal model analysis. We have provided a tool-supported methodology, allowing users to answer five analysis questions using uncertain goal models. The result of this process is a reduction of necessary uncertainties, accepting or rejecting alternative requirements with certainty. This work is the first example application of the *MAVO* uncertainty framework [20] taking semantics of a target language into account (in our case, such semantics has been provided in [11], [8]).

While we chose i^* as a particular goal modeling framework and [11] as a particular analysis algorithm, our approach can be applied more generally to work with other goal modeling frameworks (e.g., NFR [3] or KAOS [4]) or other qualitative goal model analysis approaches (e.g., [3], [7]), especially if such approaches come with their own formal semantics, translatable into FOL. We are currently working to improve the usability of *MAVO* labels (e.g., representing uncertainty over link types [6]).

We plan to further evaluate our technique by applying it to industrial RE case studies. The visual presentation of multiple analysis labels should be evaluated and improved through user studies. This work should be further integrated with goal model visualization techniques, helping users understand why the tooling produces no solution for a model [9]. Finally, we are currently considering the consequences of analysis over incomplete models, in a open world.

ACKNOWLEDGMENTS

This work was supported in part by ERC advanced grant 267856, “Lucretius: Foundations for Software Evolution”, www.lucretius.eu.

REFERENCES

- [1] R. Ali, F. Dalpiaz, and P. Giorgini. A Goal-Based Framework for Contextual Requirements Modeling and Analysis. *Requir. Eng.*, 15(4):439–458, Nov. 2010.
- [2] Y. Asnar, V. Bryl, and P. Giorgini. Using Risk Analysis to Evaluate Design Alternatives. In *Proc. of AOSE’07*, pages 140–155, 2007.
- [3] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.
- [4] A. Dardenne, A. V. Lamsweerde, and S. Fickas. Goal-Directed Requirements Acquisition. *Sci. of Comp. Prof.*, 20(1-2):3–50, 1993.
- [5] R. Degiovanni, D. Alrajehy, N. Aguirre, and S. Uchitely. Automated goal operationalisation based on interpolation and sat solving. In *Proc. of ICSE’14*, 2014.
- [6] M. Famelis and S. Santosa. MAV-Vis: A Notation for Model Uncertainty. In *Proc. of MiSE’13*, pages 7–12, 2013.
- [7] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani. Reasoning with Goal Models. In *Proc. of ER’02*, volume 3084 of *LNCS*, pages 167–181, 2002.
- [8] J. Horkoff and E. Yu. Finding Solutions in Goal Models: An Interactive Backward Reasoning Approach. In *Proc. of ER’10*, volume 6412 of *LNCS*, page 59, 2010.
- [9] J. Horkoff and E. Yu. Visualizations to support interactive goal model analysis. In *Proc. of REV’10*, pages 1–10, 2010.
- [10] J. Horkoff and E. Yu. Analyzing Goal Models: Different Approaches and How to Choose among Them. In *Proc. of SAC’11*, pages 675–682, 2011.
- [11] J. Horkoff and E. Yu. Interactive goal model analysis for early requirements engineering. *Requirements Engineering (accepted with minor revisions)*, 2014.
- [12] H. Ibrahim, B. H. Far, A. Eberlein, and Y. Daradkeh. Uncertainty Management in Software Engineering: Past, Present, and Future. In *Proc. of CCECE’09*, 2009.
- [13] S. Islam and S. H. Houmb. Integrating Risk Management Activities into Requirements Engineering. In *Proc. of RCIS’10*, pages 299–310, 2010.
- [14] K. G. Larsen and B. Thomsen. A Modal Process Logic. In *Proc. of LICS’88*, 1988.
- [15] P. Larsen. The Expressive Power of Implicit Specifications. In *Proc. of ICALP’91*, volume 510 of *LNCS*, pages 204–216, 1991.
- [16] J. Noppen, P. van den Broek, and M. Akşit. Software Development with Imperfect Information. *J. Soft Computing*, 12(1):3–28, 2008.
- [17] R. Salay, M. Chechik, S. Easterbrook, Z. Diskin, P. McCormick, S. Nejati, M. Sabetzadeh, and P. Viriyakattiyaporn. An Eclipse-Based Tool Framework for Software Model Management. In *Proc. of OOPSLA Eclipse Wrksp.*, pages 55–59, 2007.
- [18] R. Salay, M. Chechik, and J. Horkoff. Managing Requirements Uncertainty with Partial Models. In *Proc. of RE’12*, pages 1–10, 2012.
- [19] R. Salay, M. Chechik, J. Horkoff, and A. Sandro. Managing requirements uncertainty with partial models. *Requirements Engineering*, 18(2):107–128, 2013.
- [20] R. Salay, M. Famelis, and M. Chechik. Language Independent Refinement Using Partial Modeling. In *Proc. of FASE’12*, volume 7212 of *LNCS*, 2012.
- [21] R. Salay, J. Gorzny, and M. Chechik. Change Propagation Due to Uncertainty Change. In *Proc. of FASE’13*, pages 21–36, 2013.
- [22] P. Sawyer, N. Bencomo, J. Whittle, E. Letier, and A. Finkelstein. Requirements-Aware Systems: A Research Agenda for RE for Self-adaptive Systems. In *Proc. of RE’10*, pages 95–103, 2010.
- [23] S. Uchitel and M. Chechik. Merging Partial Behavioural Models. In *Proc. of SIGSOFT FSE’04*, pages 43–52, 2004.
- [24] E. Yu. Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In *Proc. of RE’97*, pages 226–235, 1997.
- [25] Y. Zhang, A. Finkelstein, and M. Harman. Search based requirements optimisation: Existing work and challenges. In *Proc. of REFSQ’08*, pages 88–94, 2008.