2010 18th IEEE International Requirements Engineering Conference

Integrating Preferences into Goal Models for Requirements Engineering

Sotirios Liaskos,
School of Information Technology
York University
Toronto, ON, Canada
liaskos@yorku.ca

Sheila A. McIlraith, Shirin Sohrabi, John Mylopoulos
Department of Computer Science
University of Toronto
Toronto, ON, Canada
sheila, shirin, jm@cs.utoronto.ca

***Abstract*—Requirements can differ in their importance. As such the priorities that stakeholders associate with requirements may vary from stakeholder to stakeholder and from one situation to the next. Differing priorities, in turn, imply different design decisions for the end system. While elicitation of requirements priorities is a well studied activity, though, the modeling and reasoning side of prioritization has not enjoyed equal attention. In this paper, we address this by extending a traditional goal modeling notation to support the representation of optional and preference requirements. In our extension, optional goals are distinguished from mandatory ones. Then, quantitative prioritizations of the former are constructed and used as criteria for evaluating alternative ways to achieve the latter. A state-of-the-art preference-based planner is utilized to efficiently search for alternatives that best satisfy the given preferences. This way, analysts can acquire a better understanding of the impact of high-level stakeholder preferences to low-level design decisions.**

***Keywords*-requirements engineering, preferences, variability**

I. Introduction

In Requirements Engineering (RE), goal-oriented techniques [1] have enjoyed significant attention due to their ability to bridge the gap between what stakeholders want (goals) and the means (actions/tasks/plans) by which these goals can be achieved. However, current goal-oriented modeling frameworks ([2], [3]) treat goals as mandatory requirements that must be fulfilled by any proposed solution. In this respect, such frameworks cannot accommodate optional ("nice-to-have") requirements that might be posed by stakeholders.

Consider, for instance, the classic "Meeting Scheduling" problem. While *Schedule a Meeting* is a mandatory goal in that the stakeholders' main problem is not solved unless this goal is fulfilled, goals such as *Use 5th Floor Rooms*, *Send Regular Reminders* or *Ensure Participation of Important Participants* are optional, in that there might be ways to solve the *Schedule a Meeting* problem without meeting several or even any of these goals. Moreover, there is a variety of types of such optional goals such as non-functional requirements (e.g., *Keep Secretary Unburdened*), low-level solution details (e.g., *Use Room with Projector*) or even more complex temporal properties of desired solutions (e.g., *Confirm Room before Announcing Meeting*). Furthermore, stakeholders are understood to often prioritize over such goals, when addressed with a particular situation in a given context. There are situations, for example, in which to *Send Regular Reminders* is not as important as to *Keep Secretary Unburdened* while in different situations the opposite might actually be true.

In an example like this, what exactly do optional requirements and preferences mean, and how do they impact our understanding of the requirements problem? If we are given a set of optional and preferred requirements, what constitutes a good solution for mandatory goals and how can we obtain it from a potentially vast set of possibilities? Generally speaking, the problem of modeling and reasoning about preferences for stakeholder goals remains largely unexplored.

The aim of this paper is to address precisely this problem. We introduce a framework for both specifying optional requirements and preferences, and also for using them to select specifications that fulfill mandatory requirements while best satisfying optional requirements and preferences. We begin by building on our previous work on goal-oriented variability modeling and analysis ([4], [5]). We propose a distinction between mandatory and optional goals and show how alternative ways to fulfill the mandatory goals decides the fulfillment or non-fulfillment of the optional ones. Then, by assigning weights of importance to optional goals of interest, as we would do in, for example, a quantitative requirements prioritization application such as the Analytic Hierarchy Process (AHP) [6], we obtain a preference function to be optimized. Our work adopts a powerful preference-based planner [7] to automatically obtain alternative plans for fulfilling mandatory goals, and also optimize the preference function. The formal underpinnings for our proposal are based on the Planning Domain Definition Language (PDDL) [8] a popular formal language for specifying AI planning problems, extended to support the definition of Hierarchical Task Networks (HTN) [9]. This formalization both allows us to have clear semantics for optionality and preference and offers us the benefit of using powerful HTN and PDDL-

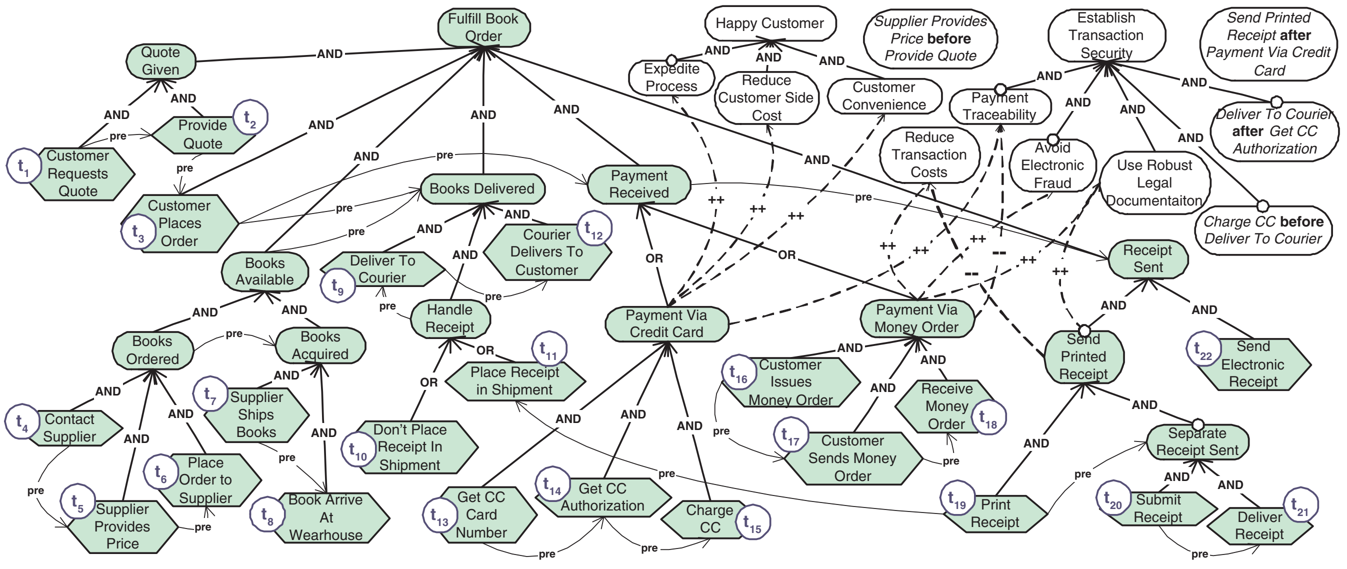


Figure 1. A goal model

based reasoning tools such as the one we actually adopt.

The rest of the paper is organized as follows. In Section II we present the goal language and show how we add temporal constructs to it. Then, in Section III, we show how that language is extended to support various types of optional goals and preferences. In Section IV we provide a sketch of the semantics of our visual representations in PDDL and HTN and in Section V we show how the planner allows us to explore alternatives that best satisfy the given preferences. In Section VI we discuss the performance of the proposed reasoning task and in Section VII we show how we can further enrich the underlying formal specification, to which the visual goal models are automatically translated, in order to support more advanced reasoning. In Section VIII we take a look at related literature and conclude in Section IX.

II. Goal Models

Goal models ([2], [10]) have been found to be effective in concisely capturing large numbers of alternative sets of low-level tasks, operations, and configurations that can fulfill high-level stakeholder goals. The characterization of a large space of such alternatives has been shown to be useful for evaluating alternative designs during the analysis process [11], for customizing designs to fit individual user characteristics [12], or even for coping with the large space of configurations of common desktop applications [13].

In Figure 1, a (simplified) goal model for a hypothetical wholesale book seller is depicted. At this point we would like to focus on the AND/OR decomposition that consists of shaded shapes, and ignore for the moment the white oval shapes at the top right part of the diagram. That decomposition model shows alternative ways by which the process of fulfilling a book order can be performed, including handling quotes to customers, placing the necessary order to the supplier, receiving a payment and sending a receipt. The model primarily consists of *goals* and *tasks*. Goals – the ovals in the diagram – are generally defined as states of affairs or conditions that one or more actors of interest would like to achieve [10]. Thus, *Payment Received* is an example of a goal. Tasks, on the other hand, – the hexagonal shapes – describe particular activities that the actors perform in order to fulfill their goals, e.g., *Print Receipt*. In the interest of conciseness, in the figure, tasks have been annotated with a literal of the form t_n, e.g. t_2 is the task *Provide Quote*. In the rest of the paper, we will make frequent use of these literals to refer to the tasks in the figure.

Goals and tasks are connected with each other via AND- and OR-decompositions as well as via *break* ($\xrightarrow{--}$) links. By AND-decomposing a goal into other goals or tasks, we indicate that the satisfaction of each of its children is necessary for the decomposed goal to be fulfilled. If the goal is OR-decomposed into other goals or tasks, then the satisfaction of one of these goals or tasks suffices for the satisfaction of the parent goal. Further, the break link ($\xrightarrow{--}$) indicates that satisfaction of the origin causes denial (non-satisfaction) of the destination – non-satisfaction of the origin does not imply anything about the destination.

In our work, the *order* in which goals and tasks are satisfied and performed respectively is relevant. To express constraints over satisfaction ordering we use a *precedence link* ($\xrightarrow{pre}$). A precedence link drawn from a goal or task to another goal or task, indicates that satisfaction/performance of the target of the link cannot begin unless the origin is satisfied or performed. Thus, the precedence link from *Customer Places Order* (task t_3 in the diagram) to *Payment Received* indicates that unless the former is performed, none of the tasks below the latter can be performed. The intended use of the precedence link is for representing indicative constraints, that is constraints that are not the desire of any stakeholder but rather properties of the domain. For example, the courier company cannot possibly deliver an order to a customer unless they first pick it up from the merchant.

Alternative solutions that satisfy the requirements problem posed by the root goal come in the form of ordered sequences of leaf level tasks, which we call *plans*. A plan for the root goal is a sequence of leaf level tasks that altogether satisfy the AND/OR structure of the root goal and possible break and precedence links. For example the sequence: $[t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_{10}, t_9, t_{12}, t_{13}, t_{14}, t_{15}, t_{19}, t_{20}, t_{21}, t_{22}]$ is a plan for the root goal, involving credit card payment as well as printing and separate submission of a receipt. Notice that it satisfies both the AND/OR structure and the precedence links. Sequence: $[t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_{10}, t_9, t_{12}, t_{13}, t_{14}, t_{15}, t_{19}, t_{21}, t_{20}, t_{22}]$ however is not a plan, because submission of the receipt (t_{20}) occurs after its delivery (t_{21}), which is physically impossible and violates the corresponding precedence link. A sequence such as $[t_1, t_2, t_3, t_{10}, t_9, t_{12}]$ is not a plan either because it does not satisfy the AND/OR decomposition: neither the books are made available nor payment and receipt are arranged in any way.

In Figure 1, we have attempted to model some interesting dilemmas that can be common in merchant transactions. Some of the dilemmas come in the form of OR-decompositions. For instance there are several ways by which the payment can be received and in the diagram we have included the options to use credit card or money order. Each of these options has a different quality: credit cards are faster and more convenient but money orders constitute a more robust legal document and allows for transactions protected from internet fraud. More OR-decomposition based alternatives can be added to the diagram that have been omitted in the interest of space: alternative ways to handle and pay for the order to the supplier, alternative arrangements for courier pick-up and delivery or more payment options such as bank transfers are examples of such alternatives.

In addition to the OR-decompositions, variability exists in the diagram from a temporal point of view. Precedence constraints do not completely restrict the order by which tasks can be performed. For example, credit card authorization (task t_{14}) must be performed after the card number is known (task t_{13}) but it may or may not precede delivery of the shipment to the courier (task t_9) or even placement of the order with the supplier, depending on, for example, the degree of trust the bookseller has for a particular customer. Similarly a quote can be provided to the customer before or after the corresponding price has been given by the supplier, depending on whether the bookseller is willing to assume the risk of not accounting for a higher supplier price, or the supplier being unable to supply the books.

The choice that stakeholders will make in all the above dilemmas will depend on their *preferences* in a given situation and context. Such preferences can be priorities over high-level qualities (such as customer convenience versus maintaining robust legal documentation) or low level constraints posing desired operational details (e.g. deliver to courier before getting credit card authorization). All these types of desires need both to be modeled and to be combined in a way that each of them influences the final choice based on its relative importance. In the next section, we show how we can represent such desires as optional goal structures and then how we can combine them into objective functions to be optimized.

III. Optional Goals and Preferences

A. Introducing Optional Goals

The AND/OR goal decomposition model we discussed in the previous section represents alternative solutions for fulfilling the root goal. These solutions came in the form of plans. We consider the root goal – and subsequently the entire AND/OR decomposition structure – to be *mandatory*, in that no plan is acceptable if that root goal (i.e. *Fulfill Book Order*) is not satisfied by it. However, there are also goals whose fulfillment by a particular plan is desired but whose non-fulfillment does not make the plan totally unacceptable. In the bookseller example, we may choose to fulfill the book order through a payment approach that is inconvenient for the customer – e.g. money order – but still accept the solution because, for example, some other goal may benefit from it, such as to maintain robust legal documents. Such goals are represented as *optional goals*.

In Figure 1, optional goals appear as white non-shaded elements on the top right of the diagram. Some optional goals, such as *Happy Customer*, are decomposed into other optional goals. The kinds of links used in the optional decomposition structures are taken from the kinds also used in the mandatory decomposition: AND, OR, as well as break links. The difference however from the mandatory decomposition is that $\xrightarrow{pre}$ links are not relevant, that is optional goals cannot be origins or targets or such links.

With the exception of optional goals that reflect temporal constraints, which we discuss later, optional goals and any of their possible subgoals are connected with the mandatory decomposition through break links, whose meaning we discussed in the previous section, as well as an additional kind of link, the *make* link. The make links ($\xrightarrow{++}$) mean that satisfaction of the origin of the link implies satisfaction of its destination. This way, we can model how certain alternative subgoals and tasks in the mandatory decompositions affect the satisfaction of the optional goals.

Going back to our example, the optional goal *Establish Transaction Security* is AND-decomposed into five other goals, including *Payment Traceability*, *Avoid Electronic Fraud* and *Use Robust Legal Documentation*. These goals receive, in turn, make and break links from the mandatory structure. Thus, *Payment Traceability* is achieved through using a credit card, hence the make link. However, if the payment is sent via a money order, this causes denial of the goal. However, money orders come at no cost to the book seller, hence a make link to the optional goal *Reduce Transaction Costs*. Printing receipts on the other hand breaks the goal to reduce transaction costs.

B. Optional Subgoals

Subgoals of either the mandatory or optional decompositions can be designated as optional too, in a manner very similar to that of the optional features in feature models [14]. To visually depict this particular kind of optional goal, we augment the entry point to the subgoal with a small circle. This style of annotation originates within feature modeling. For simplicity and intuitiveness, we restrict this annotation to children of AND-decompositions. Thus, the refined definition of the AND-decomposition is then that all AND-subgoals that are not designated as optional need to be fulfilled for the parent goal to be fulfilled. For example the optional goal *Establish Transaction Security* which is AND-decomposed into five subgoals can be satisfied even when only *Use Robust Legal Documentation* is satisfied, as all other subgoals are marked as optional. Similarly the goal *Receipt Sent* of the mandatory decomposition, can be achieved only by the goal *Send Electronic Receipt* as the goal *Send Printed Receipt* is optional.

Notice that optional subgoals in the mandatory decomposition introduce more variability to the one that already exists due to the presence of OR-decompositions. For example, the fact that the goal *Send Printed Receipt* has been designated as an optional subgoal implies more alternatives for fulfilling the root goal: those that involve printing of a receipt and those that do not.

C. Optional Temporal Constraints

In addition to optional goals that express preferences regarding the high-level quality of solutions, we can express optional goals that refer to low-level temporal constraints over the sequences of tasks that constitute a plan. Again, these constraints are not mandatory. Their satisfaction is desired but does not necessarily eliminate from consideration plans that do not satisfy them.

The difference between optional temporal constraints and other optional goals is that the former are not connected with the mandatory decomposition through make or break links. The connection with the mandatory decomposition is instead realized through being appropriately formalized, as we discuss later.

Back in Figure 1, the optional goal *Supplier Provides Price before Provide Quote* means that in a plan in which *Provide Quote* is performed we prefer that *Supplier Provides Price* precedes it in that plan. Similarly, *Send Printed Receipt after Payment Via Credit Card* indicates that if the goal *Payment Via Credit Card* has been satisfied in a plan (through performance of its subtasks) then we prefer that the goal *Send Printed Receipt* is also satisfied later in the plan.

D. Specifying Preferences

Optional goals aren't always of equal importance. In a particular situation or context or for a particular stakeholder a subset of optional goals can become more relevant than the others. To see this, let us return to our bookseller example. If an order is from a loyal and important customer, the book seller is mainly interested in keeping them happy, while reducing the possibility of electronic fraud – which may affect their relationship negatively. Meanwhile, they may also wish to maintain low transaction costs but at a lower priority. On the other hand, due to the size of the order, they also think that if a credit card is used, it is preferable to be charged before the order is actually shipped. These are four optional goals that the stakeholder may be interested in seeing satisfied when taking orders from a particular customer, while deeming the other optional goals less important or completely irrelevant. The list of relevant optional goals may include both roots and subgoals of optional decompositions, as well as optional temporal constraint goals.

Subsequently, the optional goals that are deemed to be most relevant are prioritized subject to their relevant importance using numerical weights in the real interval [0,1]. In the example situation described above, the weights we assign to our optional goals are seen on Table I.

Happy Customer	0.6
Avoid Electronic Fraud	0.2
Reduce Transaction Costs	0.1
Charge CC before Deliver To Courier	0.1

Table I
PRIORITIZING OPTIONAL GOALS

Assigning varying weights to optional goals constitutes a preference specification, in that optional goals with higher weight are understood as more preferred or important than those with lower weight or those that are not mentioned at all. As far as eliciting the priority weights is concerned, different ways of doing so have been proposed and used in practice, most prominently AHP [6]. Our proposal is concerned with modeling and reasoning about preferences and is not bound to a particular method for weight acquisition.

E. Preferred Plans

Given a preference specification as discussed above, each plan of the mandatory decomposition satisfies the preference to a different degree. Given a plan, to calculate the degree by which the plan satisfies the preference, we simply add up the weights of the optional goals that are satisfied by the plan. For example, plan:

$[t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_{13}, t_{14}, t_{15}, t_{10}, t_9, t_{12}, t_{22}]$ satisfies two of the four relevant optional goals of Table I, namely *Happy Customer* and *Charge CC before Deliver To Courier*. The former is satisfied through the inclusion of t_{13}, t_{14} and t_{15}, which satisfy the goal *Payment Via Credit Card*, which, in turn, satisfies all AND-subgoals of optional goal *Happy Customer* through the make links. The latter is satisfied because when the task t_9 (*Deliver To Courier*) is performed the task t_{15} (*Charge CC*) has already been performed in the plan. The degree of preference satisfaction for this plan is therefore 0.7.

Should we place t_{15} after t_9 though, as in plan:

$[t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_{10}, t_9, t_{12}, t_{13}, t_{14}, t_{15}, t_{22}]$ the latter temporal constraint would not be satisfied and the score for the new plan would be 0.6. Should we further change the payment method to money order, through plan:

$[t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_{10}, t_9, t_{12}, t_{16}, t_{17}, t_{18}, t_{22}]$ we would not be able to satisfy the important goal *Happy Customer* either, due to absence of make links to it. We would instead satisfy *Avoid Electronic Fraud*, *Reduce Transaction Costs*, worth of 0.2 and 0.1 respectively. This is due to the inclusion of tasks that satisfy *Payment Via Money Order* and the corresponding links towards those two optional goals. Thus the new score for the preference totals 0.3.

As we discussed earlier, different stakeholders in different contexts and situations have an interest in satisfying a different subset of optional goals, to which they also give different priorities. The bookseller example of Table I reflects the desires of a hypothetical order from a loyal and important customer; thus the corresponding preferred plans may involve choices which presume trust and strongly aim at customer satisfaction. In a different situation, a new and unknown customer may place a very small order. In this case, the book seller's priorities are to *Reduce Transaction Costs*, and, to a lesser extent, to maintain *Happy Customer*. Assume, further, that each of these two optional goals is given importance weight 0.7 and 0.3, respectively. Plans that satisfy this preference specification with a good score are ones that do not involve a printed receipt and require payment to be sent through a money order.

The goal of the reasoning mechanisms we will present below is to find plans of the mandatory decomposition that best satisfy a given preference specification. We do this by appropriately formalizing our visual goal models and preference specifications and using an efficient preference-based planner. We describe the details in the following sections.

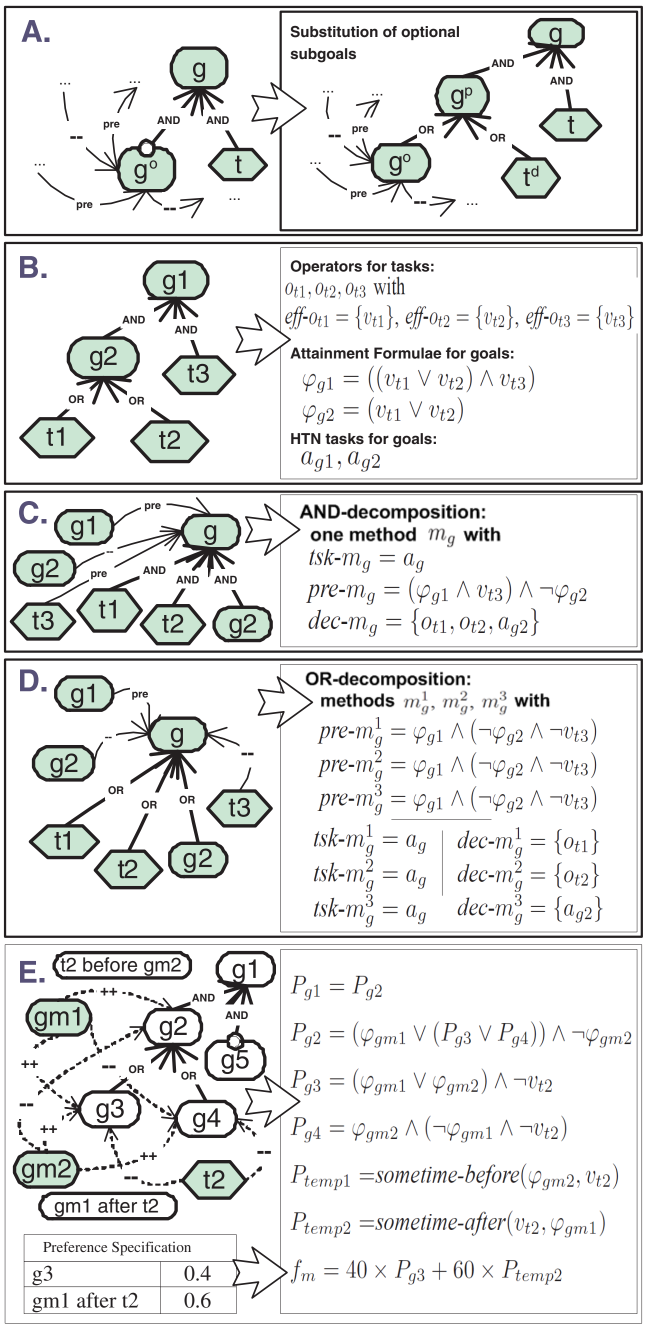
IV. Formalizing Optional Goals and Preferences

We define the semantics of our diagrammatic language as well as the preference specification via translation to a hierarchical task network (HTN) [9] and PDDL3.0 [8], two popular specification languages for planning problems. The HTN task decomposition formalism presents a superset of the AND/OR goal models we discussed above. As such, in our translation, the part that corresponds to the mandatory goal decomposition is translated to an HTN specification, while the part that corresponds to the optional goals and preferences is translated into PDDL3.0. In this paper we provide a sketch of the translation, with details appearing in a longer technical report.

A. HTN and PDDL basics

The core of an HTN description consists of a set of operators $o \in O$, a set of HTN tasks $a \in A$, a set of methods $m \in M$ as well as a set of domain predicates $v \in V$. Operators model primitive low level actions that can be performed in the domain. For each operator o, we define a precondition formula $pre\text{-}o$, which shows what needs to be true in order for performance of the operator to be possible, as well as an effect formula *eff-o* which shows what becomes true or false upon performance of the operator. Those formulae are logical expressions over domain predicates. Furthermore, HTN tasks constitute characterizations of higher level activity. Note that HTN tasks are different from tasks in the goal model and we will distinguish them by referring to them as "HTN tasks" rather just "tasks". HTN tasks cannot be "performed" but recursively reduced into other tasks and eventually into operators. Different reduction possibilities are modeled through methods. In defining a method m we define a parent task for the method $tsk\text{-}m$ as well as a set of child tasks $dec\text{-}m$. Thus, the method shows how the performance of an HTN task $tsk\text{-}m = a$ can be reduced to (i.e. realized by) the performance of a set $dec\text{-}m = a_1, a_2, \ldots$ of other lower-level tasks or operators. Methods also have preconditions $pre\text{-}m$, which mean that the method cannot be used unless $pre\text{-}m$ is satisfied.

From the PDDL3.0 constructs we use only the ones that relate to the specification of *PDDL preferences*. PDDL preferences are different from the goal-level preferences we introduced above: the former are constraints while the latter are, as we saw, priority specifications. We will distinguish between the two by using the terms "PDDL preference" and plain "preference", respectively. While PDDL preferences can take many forms, in our work we are interested in PDDL preferences that appear as temporal constraint formulae over the sequence of operators that comprise a plan. Those are logical formulae, enriched with temporal operators such as *sometime*(ϕ), *sometime-before*(ϕ_1, ϕ_2) and *sometime-after*(ϕ_1, ϕ_2), with semantics based on Linear Temporal Logic [8]. Expressions of priorities amongst PDDL preferences is possible through the definition of PDDL *metrics*. Metrics are functions whereby different PDDL preference constraints can be combined and assigned a weight. The simplest way of doing so, which we adopt here, is by constructing a linear combination $f_m = w_1 \times P_1 + w_2 \times P_2 + \ldots + w_n \times P_n$, where f_m is the metric function, P_i


A.
Substitution of optional subgoals
g
g^o
t
g^p
t^d
pre
AND
OR
B.
g1
g2
t1
t2
t3
Operators for tasks:
o_{t1}, o_{t2}, o_{t3} with
$eff\text{-}o_{t1} = \{v_{t1}\}$, $eff\text{-}o_{t2} = \{v_{t2}\}$, $eff\text{-}o_{t3} = \{v_{t3}\}$
Attainment Formulae for goals:
$\varphi_{g1} = ((v_{t1} \vee v_{t2}) \wedge v_{t3})$
$\varphi_{g2} = (v_{t1} \vee v_{t2})$
HTN tasks for goals:
a_{g1}, a_{g2}
C.
AND-decomposition:
one method m_g with
$tsk\text{-}m_g = a_g$
$pre\text{-}m_g = (\varphi_{g1} \wedge v_{t3}) \wedge \neg\varphi_{g2}$
$dec\text{-}m_g = \{o_{t1}, o_{t2}, a_{g2}\}$
D.
OR-decomposition:
methods m_g^1, m_g^2, m_g^3 with
$pre\text{-}m_g^1 = \varphi_{g1} \wedge (\neg\varphi_{g2} \wedge \neg v_{t3})$
$pre\text{-}m_g^2 = \varphi_{g1} \wedge (\neg\varphi_{g2} \wedge \neg v_{t3})$
$pre\text{-}m_g^3 = \varphi_{g1} \wedge (\neg\varphi_{g2} \wedge \neg v_{t3})$
$tsk\text{-}m_g^1 = a_g$
$tsk\text{-}m_g^2 = a_g$
$tsk\text{-}m_g^3 = a_g$
$dec\text{-}m_g^1 = \{o_{t1}\}$
$dec\text{-}m_g^2 = \{o_{t2}\}$
$dec\text{-}m_g^3 = \{a_{g2}\}$
E.
t2 before gm2
gm1 after t2
gm1
gm2
g3
g4
g5
++
--
$P_{g1} = P_{g2}$
$P_{g2} = (\varphi_{gm1} \vee (P_{g3} \vee P_{g4})) \wedge \neg\varphi_{gm2}$
$P_{g3} = (\varphi_{gm1} \vee \varphi_{gm2}) \wedge \neg v_{t2}$
$P_{g4} = \varphi_{gm2} \wedge (\neg\varphi_{gm1} \wedge \neg v_{t2})$
$P_{temp1} = sometime\text{-}before(\varphi_{gm2}, v_{t2})$
$P_{temp2} = sometime\text{-}after(v_{t2}, \varphi_{gm1})$
$f_m = 40 \times P_{g3} + 60 \times P_{temp2}$

Preference Specification	
g3	0.4
gm1 after t2	0.6

Figure 2. Translation Rules by example

are PDDL preference formulae and w_i the corresponding weights. Then, given a plan, the more of the constituent PDDL preferences P_i are satisfied by the plan, the lower (or higher depending on how optimization is defined) the value of the metric f_m will be, to a degree that depends on the weights w_i of the individual preferences.

B. Translating to HTN and PDDL

We now show how our visual representations of goals and preferences can be translated into HTN and PDDL. In the interest of simplicity and space we sketch how the translation is possible based on the examples of Figure 2.

Roughly, the mandatory decomposition is translated into a set of HTN operators, tasks and methods, while the set of optional goals as well as the preferences are translated into PDDL preference constraints and metrics. To translate the mandatory decomposition we first transform the goal model in order to eliminate the optional subgoals. As seen in frame (A.) of Figure 2, for each subgoal g^o that is optional, we introduce one new goal g^p and one new task t^d. The goal g^p is then OR-decomposed into the original g^o and t^d, and takes the place of g^o in the original tree. Possible $\xrightarrow{pre}$ and $\xrightarrow{--}$ links stay connected with g^o. The newly introduced task t^d is a "dummy" task that is ignored when it appears in plans.

The goal model that results after eliminating optional subgoals is then translated into an HTN specification as follows. First, each leaf level task t is translated into an operator o_t. A domain predicate v_t is also created and added as the effect *eff*-o_t of the introduced operator, signifying that, when it is true, the operator has been performed. For each goal g, on the other hand, including the root, we introduce HTN task a_g. For each goal we also introduce a formula that is grounded on leaf level tasks; we call it attainment formula ϕ_g. The structure of ϕ_g reflects the structure of the AND/OR subtree rooted to goal g and is used for the definition of preconditions. These steps can be seen in the example of frame (B.) of Figure 2.

For each goal g we also introduce one or more HTN methods. In particular, if g is AND-decomposed, one method is introduced that connects the corresponding HTN task a_g with the set of HTN tasks or operators that correspond to each of its subgoals (see frame (C.) of Figure 2). If the goal, however, is OR-decomposed into n subgoals, then n methods are introduced, each connecting the HTN task that corresponds to the parent goal with the HTN task or operator that corresponds to each OR-subgoal, as seen in frame (D.) of Figure 2. The preconditions of the introduced operators and methods are defined based on the $\xrightarrow{pre}$ and $\xrightarrow{--}$ links that the corresponding task or goal receives. Thus, as seen in frames (C.) and (D.) of the figure, the conjunction of all $\xrightarrow{pre}$ links must be satisfied and the disjunction of all $\xrightarrow{--}$ links must *not* be satisfied, for the operation or method to be considered.

Finally, each optional goal is translated into a PDDL preference formula that reflects the AND/OR decomposition structure of the goal. In addition, if the goal receives $\xrightarrow{++}$ and $\xrightarrow{--}$ links, a disjunction of the former and a negated conjunction of the latter are appropriately included in the logical formula. Optional goals that are temporal constraints are appropriately translated into PDDL temporal formulae using the temporal modalities such as *sometime-before*(ϕ_1, ϕ_2) we discussed above. Once the optional goals are translated into PDDL preferences, the actual requirements preference specification, such as the one on Table I, is translated into a PDDL metric. The treatment of optional goals and preferences is exemplified in frame (E) of Figure 2.

C. Templates for Optional Temporal Constraints

The translation of the visual goal and preference language into PDDL can be performed automatically using the rules sketched above, thereby allowing analysts to make use of the reasoning capabilities of the HTN-PDDL planner without having to be exposed to elements of these formal languages. An exception to this is the use of optional temporal constraints that can theoretically be of arbitrary length and expressiveness in natural language. This would imply that a complex optional temporal constraint would need to be manually translated into the appropriate PDDL constraints.

Nevertheless, our understanding from experimenting with modeling several domains using our framework is that two most common and useful temporal constraints are simple expressions of precedence (*before*) and response (*after*) between goals and tasks. It is therefore possible to use natural language templates to facilitate automated translation for a large number of practical optional constraints. An example of such a template is of the form "*goal1/task1* before/after *goal2/task2*". This way automatic translation into the corresponding PDDL *sometime-before*(ϕ_{g2}, ϕ_{g1}) and *sometime-after*(ϕ_{g2}, ϕ_{g1}) expressions is possible. The examples of optional temporal constraints in Figure 1, such as for instance "Supplier Provides Price before Provide Quote" or "Send Printed Receipt after Payment Via Credit Card", are actually written using such a template.

V. Reasoning about Preferences

By translating the goal formalisms to HTN and PDDL it is possible to use an HTN preference-based planner to identify plans that optimize for the priorities amongst optional goals and preferences. To this end, we employ HTNPlan-P [7], an extension of the popular SHOP2 HTN planner [9] that supports the optimization of PDDL-based preferences. The input to the planner is an HTN domain specification and a set of PDDL preferences and metrics, as well as a problem specification which involves definition of initial conditions (for the domain predicates) and the planning goal which is normally a top level HTN task.

The planner searches through recursive reductions of the top level HTN task into subtasks or operators, with the objective of finding a sequence of operators that satisfies the various precondition and effect constraints while optimizing the metric function that encodes the prioritized PDDL preferences. The search is performed in a best-first, incremental manner and is guided by several heuristics. Thus, the planner continues to output plans with improved metric values until the search space has been exhausted. Branch-and-bound search is applied for excluding search directions that are guaranteed not to provide a better metric value, dramatically reducing the search space and making the planner very usable for our application.

In our context, the planner is given as input the translated domain theory and preference specification in HTN and PDDL, respectively, as well as a problem specification with empty initial conditions and the HTN task that corresponds to the root goal of the mandatory decomposition. The planner returns a sequence of operators that suffice for the performance of the given HTN task, as described above. Each operator can be directly mapped to a task of the goal model, meaning that the plan that the planner returns directly corresponds to a plan of the goal model.

Returning to our bookseller example, given the preference specification in Table I, the planner returns the plan:

$$[t_4, t_5, t_6, t_7, t_8, t_1, t_2, t_3, t_{13}, t_{14}, t_{15}, t_{22}, t_{19}, t_{11}, t_9, t_{12}]$$

as optimal, with score 0.7. The above satisfies the goal to have *Happy Customer* which is achieved through the convenience of using credit cards. Delivery to courier (t_9) also happens after the credit card is charged (t_{15}). However, the other constituents of the priority specification, such as avoidance of electronic fraud could not be met at the same time leading to a compromised score. Should the preference specification instead reflect the priorities associated with a smaller order for an unknown customer, i.e., prioritizing the reduction of transaction costs (e.g., by 0.7) while still attempting to maintain a happy customer (e.g., by 0.3), then the planner would return a plan where money order instead of credit card is selected and receipts are not printed.

Through further exploring this process with several examples, we found that it could support very well an iterative process for preference acquisition, whereby the result of an initial preference specification triggers the introduction of additional or removal of existing constraints. For instance, after studying the above plan, the bookseller may suggest that the order to the supplier (t_6) should not precede the customer confirming placement of the order (t_3). Thus, a temporal constraint goal such as *Customer Places Order Before Contact Supplier* can be added to the list and a process for redistributing weights may follow. The process may repeat until satisfactory plans are returned. We believe that through such a practice of iteratively specifying preferences, acquisition of a better understanding of the domain as well as the impact of high-level stakeholder desires to low level design decisions can be achieved.

VI. Performance Evaluation

While HTNPlan-P is known to perform remarkably well for fairly large and complex hierarchical tasks networks [7], we performed additional performance evaluation steps in order to understand the planner's usability on practical goal and preference modeling tasks. Thus, in addition to the bookseller domain (34 mandatory elements in total, 21 tasks, 13 optional goals) we also tried goal models from a case study in the nursing domain (24 mandatory elements,

Model	Size	First Result	Best Result
Nursing	24	0.04s	0.1s
ATM	28	0.06s	0.13s
Bookseller	34	0.09s	0.14s
Mtg. Sched.	53	0.04s	2.0s

Table II
PERFORMANCE RESULTS FOR PRACTICAL GOAL MODELS

11 tasks, 7 optional goals), an extended meeting scheduling example (53 mandatory elements, 32 tasks, 13 optional goals) as well as an ATM example (28 mandatory elements, 21 tasks, 4 optional goals).

The results can be seen in Table II. The first column ("First Result") shows the time it takes for the planner to return the first plan (which may be suboptimal) and the second column ("Best Result") shows the time required to generate the optimal plan. Times are in seconds. From the preferences that we tried for each domain, the worst cases are reported.

Observe that the running time rarely exceeds one second. The significance of this result is that, at least for this class of problem sizes, our framework can be used in applications where quick results are demanded due to usability or other constraints, such as on-line iterative preference analysis or even requirements-driven run-time software adaptation.

VII. ADDING EXPRESSIVENESS

To this point we have presented an approach to specifying goal models and preferences in a high-level graphical way, usable by requirements analysts who have no knowledge of the underlying HTN and PDDL-based representation and reasoning mechanism. However, in doing so, we are not availing ourselves of the full modeling and reasoning capabilities afforded by HTN, PDDL3.0 and HTNPlan-P. Some of these further capabilities can be exploited by appropriately extending the result of the automated translation of the goal models into the HTN and PDDL specifications. In this section we describe two capabilities we found particularly useful in the context of requirements engineering: the explicit representation of state, and the ability to represent classes of objects and to differentiate and reason about different individuals within those classes.

A. Representing State

As we saw, PDDL supports the definition of domain predicates, whereby the state of the environment or other interesting facts about the domain can be modeled. These predicates can be used in a variety of roles. Returning to our bookseller example, such predicates might include description of the state of the book stock e.g. *stock(low)*, *stock(zero)* and *stock(ok)*, qualities of the client e.g. *client(loyal)*, *client(new)* or qualities of the order e.g. *order(major)*, *shipmentDistance(large)*.

These predicates can be used in the specification of preconditions to HTN-PDDL operators and methods, in the specification of effects of operators, and as conditions upon which preferences are predicated. For example, it is compelling to condition the optional goal *Deliver to Courier after Get CC Authorization* on the truth of the predicate *order(major)*. Referring to the literals we have established in Figure 1 the corresponding PDDL preference could be updated to:

$$\textit{always(order(major)}\Rightarrow\textit{sometime-after}(v_{t_{14}}, v_{t_9}))$$

where *always* is a PDDL temporal construct with the conventional Linear Temporal Logic semantics [8]. This will make the temporal preference relevant only if *order(major)* is true for the problem instance being examined.

B. Representing Classes of Objects

In many requirements engineering applications, it is useful not only to represent and reason about state but further to distinguish classes of objects and to specify goals and preferences over individuals within these classes. Returning to our bookseller example, we may, for instance define Customers, Orders, Employees or Delivery Companies as distinct object classes, different instances of which can then be declared in problem definitions.

Operators can then be defined as n-ary predicates whose arguments range over objects of different types. Thus, the task *Deliver To Courier* in Figure 1 can be translated to an operator *deliverToCourier(Employee, Company, Order)* where the *Employee* and *Company* are variables describing the particular employee of the bookseller that performs the delivery, the particular courier company that is called to perform the delivery and the individual order that is being delivered, respectively. The domain predicates can be extended appropriately to express the state of the system as for example in *shipment-in-progress(Company, Order)* which represents that a particular company is currently shipping a particular order instance.

Preferences can then be represented at a similar level of specificity, if desired. For example, *"always ship rush orders by UPS"* or *"for any order placed by Orange Books Inc, payment should be received before couriering"* illustrate the specialization of preferences to individual objects. Preferences can similarly be aggregated over groups of objects *"all orders of a particular client should be handled by the same delivery company"* or *"deliver all rush orders within 5 business days"*. Ideally such preferences would be satisfied for all individuals of a type of object, but when this is not possible, plans that maximize the number of individuals for which these preferences are satisfied would be preferred.

By exploiting the expressive power of HTN and PDDL in this way, we can more accurately illustrate the impact that stakeholder preferences have on aspects that are particular to a specific domain instance.

VIII. RELATED WORK

The need to include stakeholder attitudes and preferences, as well as optionality in requirements modeling has been demonstrated by Jureta et al. [15] through reference to the nature of the linguistic matter that serves the communication between stakeholders and analysts. Requirements have indeed been traditionally understood as having varying degrees of importance amongst stakeholders. Approaches for eliciting these degrees of importance range from simple empirical division of requirements into "must-have" and "nice-to-have" (e.g. [16], [17]), to more elaborate quantitative prioritization techniques, such as the Analytic Hierarchy Process (AHP) [6], multi-criteria analysis methods [18] or more advanced techniques based on machine learning [19].

However, while those approaches address priority elicitation, the problem of modeling and reasoning about priorities and alternative solutions has not received as much attention. Some efforts that attempt to reason about different requirements priorities (e.g. [20]) focus mostly on identifying combinations of coarse-grained features of the end-system rather than high-level stakeholder goals or behavioral characteristics of possible solutions.

In the requirements engineering literature, a constraint language for selecting scenario instances from generic use-cases has been proposed [22], while elsewhere [23], requirements alternatives are computed through reasoning about partial satisfaction of quality goals. Preference matrices have also been used for measuring various qualities of a requirements specification [24]. However, these approaches do not focus on how optionality and preference (versus hard-constraints) can be modeled or they do not put emphasis on both high-level quality goals of stakeholders and low-level temporal characteristics of solutions, as well as the connection between the two. Nevertheless, Jureta et al. propose a comprehensive approach for modeling preference and optionality by introducing a novel requirements modeling framework called Techne [25], founded on the CORE ontology [15]. Techne's graph structures offer an expressive way to model mandatory and optional requirements, preferences among them, domain assumptions and arguments for/against any part of a requirements model. The underlying semantics of optionality and preference are the same in the two proposals. However, Techne's framework is strictly qualitative (no numerical weights) and does not accommodate precedence links, nor is it offering specific algorithms for finding solutions to requirements problems. As such, Techne constitutes a general (and rich) framework for goal-oriented requirements modeling languages which has yet to be instantiated. Our proposal, on the other hand, offers a concrete language and reasoning support for discovering solutions to requirements problems involving priorities and preferences.

Our inclusion of optional subgoals in the mandatory decomposition triggers a question about the relationship between goal models and feature models (e.g. [14], [26]) in terms of their expressiveness and semantics. However, the difficulty of intuitively relating the meaning of both the two base concepts (feature vs. goal) and their possible decompositions, as well as the inclusion of a temporal dimension to the goal models seem to obstruct a trivial and direct comparison of the two notations on such a basis. Exploring this relationship is certainly part of our future research plans.

IX. Conclusions

We presented a goal-based framework for modeling and reasoning about optional and preferred requirements. Our main contributions are an approach for modeling optionality and preference in traditional goal models, as well as a method for automatically identifying design alternatives for given preference specifications. We begin by introducing in the traditional goal modeling notation the distinction between mandatory and optional goals. The former represent alternative solutions to the core requirements problem, while the latter express desired qualities or temporal ordering constraints of such solutions. A preference specification is then a definition of priorities amongst a selection of optional goals, through assignment of numerical weights, similar to the kind of result that popular quantitative priority elicitation techniques produce. The goal model with the preference specification is then translated into a popular language for defining planning problems, and a state-of-the-art planner is used to identify alternatives of the mandatory decompositions that best satisfy the specified preferences. The formalization output can further be extended to allow for more advanced reasoning. In terms of performance, we show that the tool performs very well in several realistic goal and preference models that we constructed.

Through experimentation with models representing different domains we found that the task of reasoning about preferences and alternatives allows better understanding of the connection between the stakeholder attitudes and alternative behavioral designs. This makes it particularly useful for applications such as exploring alternative designs during earlier requirements stages, supporting a priority elicitation activity through directly showing the implication of certain prioritizations or, potentially, customizing software-systems through connecting the high-level design descriptions obtained through the tool into configurations of variation points in the software itself [13].

Several directions for future work emerge from the effort presented in this paper. Firstly, we would like to apply our framework to even larger problems in order to assess its scalability both in terms of model comprehensibility and in terms of reasoning performance. Secondly, while we expect that current numerical requirements prioritization techniques are applicable to our approach, we would certainly like to perform an empirical exploration in order to better understand how the particular flavor of optional goals (high level qualities or temporal constraints) influences the priority elicitation process. Finally, by looking closely at current variability representation techniques (e.g., [14], [26], [21]) we plan to look at enhancements to our language that would make it an even stronger variability representation tool.

References

[1] A. van Lamsweerde, "Requirements engineering in the year 00: a research perspective," in *Proceedings of the 22nd International Conference on Software Engineering (ICSE'00)*. New York, NY, USA: ACM, 2000, pp. 5–19.

[2] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition," *Science of Computer Programming*, vol. 20, no. 1-2, pp. 3–50, 1993.

[3] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani, "Reasoning with goal models," in *Proceedings of the 21st International Conference on Conceptual Modeling (ER'02)*, London, UK, 2002, pp. 167–181.

[4] S. Liaskos, A. Lapouchnian, Y. Yu, E. Yu, and J. Mylopoulos, "On goal-based variability acquisition and analysis." in *Proceedings of the 14th IEEE International Conference on Requirements Engineering (RE'06)*, Minneapolis, MN, 2006, pp. 76–85.

[5] S. Liaskos, S. A. McIlraith, and J. Mylopoulos, "Towards augmenting requirements models with preferences," in *Proceedings of the 24th International Conference on Automated Software Engineering (ASE'09)*, Auckland, New Zealand, 2009, pp. 565–569.

[6] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing requirements," *IEEE Software*, vol. 14, no. 5, 1997.

[7] S. Sohrabi, J. A. Baier, and S. McIlraith, "HTN planning with preferences," in *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, Pasadena, CA, USA, 2009, pp. 1790–1797.

[8] A. Gerevini and D. Long, "Plan constraints and preferences in PDDL3," Univ. of Brescia, Tech. Rep. Technical Report, Department of Electronics for Automation, 2005.

[9] D. Nau, Y. Cao, A. Lotem, and H. M. noz Avila, "SHOP: Simple Hierarchical Ordered Planner," in *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, Stockholm, Sweden, 1999, pp. 968–973.

[10] E. S. K. Yu and J. Mylopoulos, "Understanding "why" in software process modelling, analysis, and design," in *Proceedings of the 16th International Conference on Software Engineering (ICSE'94)*, Sorrento, Italy, 1994, pp. 159–168.

[11] J. Mylopoulos, L. Chung, S. Liao, H. Wang, and E. Yu, "Exploring alternatives during requirements analysis," *IEEE Software*, vol. 18, no. 1, pp. 92–96, 2001.

[12] B. Hui, S. Liaskos, and J. Mylopoulos, "Requirements analysis for customizable software: A goals-skills-preferences framework," in *Proceedings of the 11th IEEE International Requirements Engineering Conference (RE'03)*, Monterey Bay, California, 2003, pp. 117–126.

[13] S. Liaskos, A. Lapouchnian, Y. Wang, Y. Yu, and S. Easterbrook, "Configuring common personal software: a requirements-driven approach." in *Proceedings of the 13th IEEE International Conference on Requirements Engineering*, Paris, France, 2005, pp. 9–18.

[14] K. Czarnecki and U. W. Eisenecker, *Generative Programming - Methods, Tools, and Applications*. Addison-Wesley, 2000.

[15] I. Jureta, J. Mylopoulos, and S. Faulkner, "Revisiting the core ontology and problem in requirements engineering," in *Proceedings of the 16th IEEE International Conference on Requirements Engineering (RE'08)*, Barcelona, Spain, 2008, pp. 71–80.

[16] K. Beck, *Extreme Programming Explained*. Addison Wesley, 1999.

[17] D. Clegg and R. Barker, *Case Method Fast-Track: A RAD Approach*. Addison Wesley, 1994.

[18] H. P. In, D. Olson, and T. Rodgers, "Multi-criteria preference analysis for systematic requirements negotiation," in *Proceedings of the 26th Annual International Computer Software and Applications Conference (COMPSAC'02)*, 2002, pp. 887–892.

[19] P. Avesani, C. Bazzanella, A. Perini, and A. Susi, "Facing scalability issues in requirements prioritization with machine learning techniques," in *Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE'05)*, Paris, France, 2005, pp. 297–306.

[20] H. Zhang, S. Jarzabek, and B. Yang, "Quality prediction and assessment for product lines." in *Proceedings of the 15th International Conference on Advanced Information Systems Engineering (CAiSE'03)*, Klagenfurt, Austria, 2003, p. 681.

[21] G. Halmans and K. Pohl, "Communicating the variability of a software-product family to customers." *Software and System Modeling*, vol. 2, no. 1, pp. 15–36, 2003.

[22] A. G. Sutcliffe, N. A. M. Maiden, S. Minocha, and D. Manuel, "Supporting scenario-based requirements engineering," *IEEE Transactions on Software Engineering*, vol. 24, no. 12, pp. 1072–1088, 1998.

[23] R. Sebastiani, P. Giorgini, and J. Mylopoulos, "Simple and minimum-cost satisfiability for goal models," in *Proceedings of the 16th Conference On Advanced Information Systems Engineering (CAiSE'04)*, Riga, Latvia, 2004, pp. 20–35.

[24] H. Kaiya, H. Horai, and M. Saeki, "AGORA: attributed goal-oriented requirements analysis method," in *Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'02)*, Essen, Germany, 2002, pp. 13 – 22.

[25] I. J. Jureta, A. Borgida, N. A. Ernst, and J. Mylopoulos, "Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling," in *Proceedings of the 18th IEEE International Requirements Engineering Conference (RE'10)*, Sydney, Australia, 2010.

[26] P.-Y. Schobbens, P. Heymans, and J.-C. Trigaux, "Feature diagrams: A survey and a formal semantics," in *Proceedings of the 14th IEEE International Conference on Requirements Engineering (RE'06)*, Minneapolis, MN, 2006, pp. 139–148.