

Representing and reasoning about preferences in requirements engineering

Sotirios Liaskos · Sheila A. McIlraith ·
Shirin Sohrabi · John Mylopoulos

Received: 11 January 2011 / Accepted: 23 July 2011 / Published online: 12 August 2011
© Springer-Verlag London Limited 2011

Abstract 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 system to be. While elicitation of requirement priorities is a well-studied activity, modeling and reasoning with prioritization has not enjoyed equal attention. In this paper, we address this problem by extending a state-of-the-art goal modeling notation to support the representation of preference (“nice-to-have”) requirements. In our extension, preference 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. To generate solutions, an existing preference-based planner is utilized to efficiently search for alternatives that best satisfy a given set of mandatory and preferred requirements. With such a planning tool, analysts can acquire a

better understanding of the impact of high-level stakeholder preferences on low-level design decisions.

Keywords Requirements engineering · Goal modeling · Preference specification

1 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 [2–4] treat goals as mandatory requirements that must be fulfilled by any proposed solution. In this respect, such frameworks cannot accommodate preference (“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 all preferences, in that there might be ways to solve the *Schedule a Meeting* problem without meeting several or even any of the preference goals. Moreover, there is a variety of types of such preference goals, such as non-functional requirements (e.g., *Keep Secretary Unburdened*), 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. In

Extended version of the paper titled *Integrating Preferences into Goal Models for Requirements Engineering* that appears in the Proceedings of the 18th International Requirements Engineering Conference, Sydney, Australia, 2010, pp. 135–144 [12].

S. Liaskos (✉)
School of Information Technology,
York University, Toronto, Canada
e-mail: liaskos@yorku.ca

S. A. McIlraith · S. Sohrabi · J. Mylopoulos
Department of Computer Science, University of Toronto,
Toronto, Canada
e-mail: sheila@cs.toronto.edu

S. Sohrabi
e-mail: shirin@cs.toronto.edu

J. Mylopoulos
e-mail: jm@cs.toronto.edu

some situations, for instance, the preference *Send Regular Reminders* is not as important as to *Keep Secretary Unburdened* while in others the opposite might actually be true.

In an example such as this, what exactly do preference requirements and priorities between them mean and how do they impact our understanding of the requirements problem? If we are given a set of preference requirements with priorities defined between them, what constitutes a valid 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 stakeholders in the presence of mandatory goals remains largely unexplored.

The aim of this paper is to address precisely this problem. We introduce a framework for both specifying preference requirements and priorities among them, and also for using them to select specifications that fulfill mandatory requirements while best satisfying preference requirements and priorities. Our proposed framework builds on our previous work on goal-oriented variability modeling and analysis [5–7]. This work is extended to distinguish between mandatory and preference goals, and we show how alternative ways to fulfill mandatory goals determine the fulfillment or non-fulfillment of preference ones. Then, by assigning weights of importance to preference goals—using, for example, a quantitative requirement prioritization scheme such as the Analytic Hierarchy Process (AHP [8])—we obtain a preference function to be optimized. Our work adopts a powerful preference-based planner [9] to automatically obtain alternative plans for fulfilling mandatory goals and also optimize the preference function. The formal underpinnings for our proposal are based on PDDL 3.0 [10], a widely used formal language for specifying AI planning problems, extended to support the definition of hierarchical task networks (HTNs—[11]). This formalization both allows us to have clear semantics for preferences and priorities among them and offers us the benefit of using powerful HTN- and PDDL-based reasoning tools such as the one we actually adopt.

The paper extends our earlier publication on the topic [12] over four directions. Firstly, we provide the complete semantics of the visual language into HTN and PDDL 3.0 specifications, including the temporal templates that we are using to allow easier construction of temporal constraints. Secondly, we show through examples how exploring the impact of preferences to the selection of alternatives can facilitate understanding of the domain, validation of our model thereof, as well as communication of the attitudes and priorities of the stakeholders. Thirdly, we offer a more detailed account on how the HTN/PDDL specification that comes out of the translation procedure can be enriched to allow for instance-level reasoning about preferences.

Finally, we offer more details and experiences from our applications as well as experimental results on the performance of the reasoning tool and how it is affected by the size of the goal model.

The rest of the paper is organized as follows. In Sect. 2, we present the goal language and show how we add temporal semantics to it. Then, in Sect. 3, we show how that language is extended to support various types of preference goals and priorities. In Sect. 4, we provide the detailed semantics of our visual representations in PDDL and HTN, while in Sect. 5, we show how the planner allows us to explore alternatives that best satisfy given preferences while fulfilling mandatory goals. In Sect. 6, we show how we can further enrich the underlying formal specification, to which visual goal models are automatically translated, in order to support more advanced reasoning. Then, in Sect. 7, we provide reflections from our applications of the approach in different example domains, and in Sect. 8, we discuss the performance of the proposed reasoning task. Finally, in Sect. 9, we take a look at related literature and conclude in Sect. 10.

2 Goal models

Goal models [2, 13] 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 requirements. The capture of a large space of such alternatives has been shown to be useful for evaluating alternative designs during the analysis process [7], for customizing designs to fit individual user characteristics [6] or even for coping with the large space of configurations of common desktop applications [14].

In Fig. 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 and left parts 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 [13]. Thus, *Payment Received* is an example of a goal. Tasks, on the other hand—the hexagonal shapes—describe particular activity that actors perform in order to fulfill their goals, e.g., *Print Receipt*. For 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 of the figure.

Goals and tasks are connected with each other using AND- and OR-decompositions. By AND-decomposing a goal into other goals or tasks, we imply 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. Furthermore, subgoals of AND-decompositions can be designated as *optional*, in a manner very similar to that of the optional features in feature models [15]. To visually signify this particular kind of optional goals, we use a small circular annotation on their top side, a decoration borrowed exactly from feature modeling. Thus, the refined definition of the AND-decomposition is that all AND-subgoals that are not designated as optional need to be fulfilled for the parent goal to be fulfilled. For example, goal *Receipt Sent* can be achieved by the task *Send Electronic Receipt* (t_{23}) alone as the goal *Printed Receipt Sent* is optional. The use of optional subgoals allows goal analysts to avoid having to represent optionality through OR-decompositions where one of the two OR-subgoals expresses non-satisfaction of the parent goal—a rather awkward modeling practice, which is however necessary if optional subgoals are absent.

In our work, the *order* in which goals and tasks are satisfied and performed, respectively, is relevant. To express constraints in satisfaction ordering, we use the *precedence link* (\xrightarrow{pre}). The precedence link is drawn from

a goal or task to another goal or task, meaning 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* means that unless the former is performed, none of the tasks below the latter can be performed. In addition to the (positive) precedence link, the *negative precedence link* (\xrightarrow{npr}) can be used to denote that satisfaction/performance of the target of the link cannot begin if the origin is already satisfied or performed. The intended use of the precedence links 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:

$[t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_{11}, t_{10}, t_{13}, t_{14}, t_{15}, t_{16}, t_{20}, t_{21}, t_{22}, t_{23}]$

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:

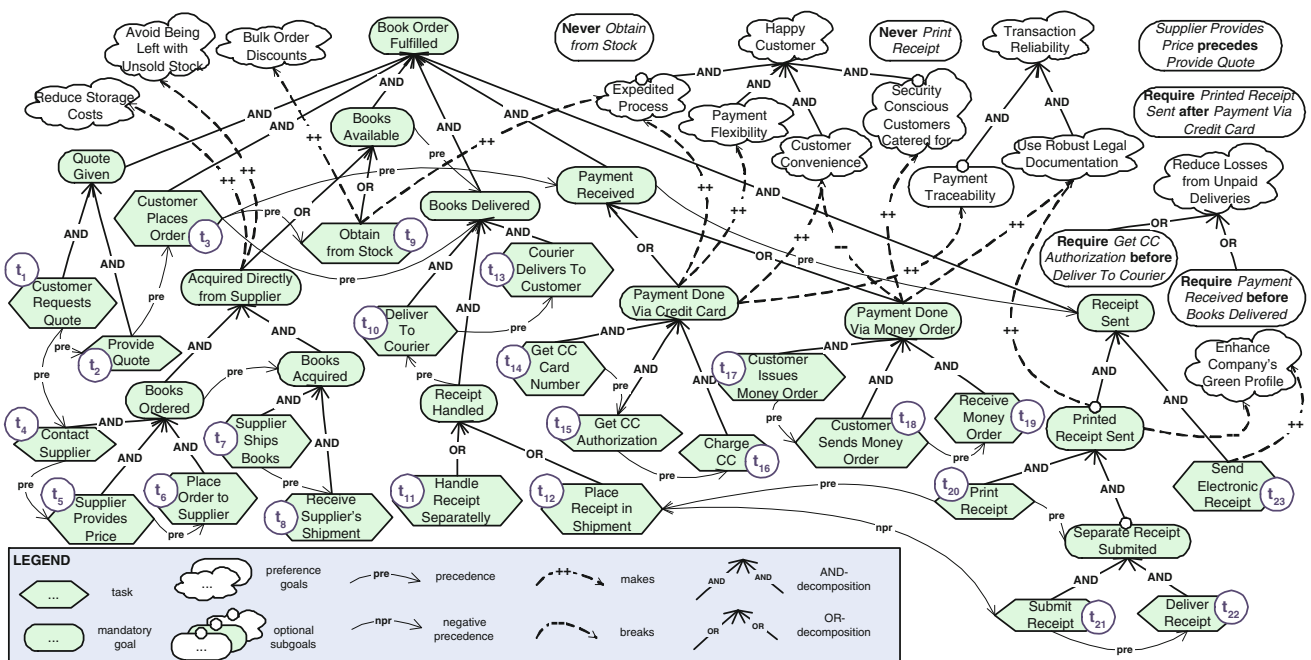


Fig. 1 A goal model

$[t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_{11}, t_{10}, t_{13}, t_{14}, t_{15}, t_{16}, t_{20}, t_{22}, t_{21}, t_{23}]$

however is not a plan, because submission of the receipt (t_{21}) is happening after its delivery (t_{22}), which is physically impossible and therefore violates the corresponding precedence link. A sequence such as $[t_1, t_2, t_3, t_{11}, t_{10}, t_{13}]$ 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.

The visual formalism we use for our goal models can be seen as a subset of a strategic rationale i^* diagram [4], whereby OR-decompositions correspond to alternative means–ends relationships. Thus, Fig. 1 shows the goals of the bookseller, in which some of the leaf-level tasks are subject to delegation to other actors (customer, supplier, etc)—for simplicity, we have omitted the corresponding i^* actor and dependency elements. Two aspects that are additional to the core i^* constructs are the optional subgoals, which, as we saw, can be seen as “syntactic sugar”, as well as the precedence annotations, which have been attempted before for the purpose of preparing for subsequent formalization processes [16, 17].

In Fig. 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 and optional subgoals. 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. Thus, credit cards are faster, more convenient and allow customers the flexibility to pay at a later time. Money orders, on the other hand, constitute a more robust legal document and are preferred by security conscious customers. On the other hand, the goal *Printed Receipt Sent* has been designated as optional subgoal, which implies more alternatives for fulfilling the root goal: those that involve printing of a receipt and e.g., allow for a better documented transaction and those that do not, which support a more environmentally friendly transaction. More alternatives can be added to the diagram that have been omitted for 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 effect of OR-decompositions and optional subgoals, 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_{15}) must be performed after the card number is known (task t_{14}) but it may or may not precede delivery of the shipment

to courier (task t_{10}) or even placement of the order to the supplier (t_6), depending on e.g., the degree of trust of the bookseller to a particular customer and/or supplier. 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 take a risk of not accounting for a higher supplier price or inability to supply.

The choice that stakeholders will make in all the above dilemmas will depend on their priorities at a given situation and context. Such priorities can be over high-level qualities (such as customer convenience versus maintaining robust legal documentation) or constraints posing desired sequences of goal satisfaction (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 relevant importance. In the next section, we show how we can represent such desires as preference structures and then how we can combine them into objective functions to be optimized.

3 Preferences and priorities

3.1 Introducing preference 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., *Book Order Fulfilled*) 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 *preferences* or *preference goals*—below we use the terms interchangeably.

Preferences are also goals in a sense that stakeholders may desire their fulfillment—though not always in the same degree of desirability. In Fig. 1, preferences appear as white non-shaded elements on the top right and left of the diagram. Preference goals can be hard goals as in the mandatory decomposition but they can also be *soft goals*. Soft goals, as opposed to hard goals, do not have a clear-cut criterion to be used in order to decide whether they are

satisfied or not. Thus, *Happy Customer* is a typical soft-goal, as there is no precise and objective way to know when a customer is actually happy, but argue about its satisfaction based on e.g., relevant evidence or approximations [18]. Some preferences, such as *Happy Customer*, are AND- and OR-decomposed into other preference goals forming decomposition trees just like the mandatory goals—but without precedence links, which cannot be origins or targets of preferences.

With the exception of preferences that reflect temporal constraints, which we discuss later, preference goals and any of their possible subgoals are connected with the mandatory decomposition through *break* and *make* links, directly adopted from i^* . The break link ($\overset{-}{\rightarrow}$) means that satisfaction of the origin of the link causes denial (non-satisfaction) of the destination. The make link ($\overset{+}{\rightarrow}$) means that satisfaction of the origin implies satisfaction of its destination. In both cases though, non-satisfaction of the origin does not imply anything about the destination.

Using break and make links, we can model how certain alternative subgoals and tasks in the mandatory decomposition affect the satisfaction of the preference goals. Going back to our example, the preference goal *Happy Customer* is AND-decomposed into four other goals: *Expedited Process*, *Payment Flexibility*, *Customer Convenience*, and *Security Conscious Customers Catered for*. These goals receive, in turn, make, and break links from the mandatory structure. Thus, *Customer Convenience* is achieved through making the payment via credit card, hence the make link. However, if the payment is sent through money order, this causes denial of the goal. Nevertheless, payment through money order may be desired by customers who fear internet fraud, hence a make link to the preference goal *Security Conscious Customers Catered for*. Printing receipts on the other hand breaks the goal to *Enhance Company's Green Profile*.

3.2 Temporal preferences

In addition to preferences that express quality desires of the solutions, we can express preference goals that refer to temporal constraints over the sequences of goals and tasks

of the mandatory decomposition. Again, however, these constraints are not mandatory in a sense that their satisfaction is desired without necessarily making unacceptable plans that do not satisfy them. They are represented as hard goals since, as temporal constraints over other hard elements, their satisfaction criterion is precise.

As opposed to other preferences, temporal preferences are not connected with the mandatory decomposition through make or break links. The connection with the mandatory decomposition is instead performed through being appropriately formalized, as we discuss later.

To maintain simplicity and to ensure that reasoning about temporal preferences is possible using the infrastructure we introduce below, we formulate the constraints using templates. Table 1 depicts seven templates that we found to be useful for the construction of almost every temporal property that came up in our applications. In the table, $[X]$ and $[Y]$ denote goals or tasks of the mandatory decompositions.

Using templates, construction of simple temporal properties is possible by users with limited knowledge of temporal logic. Back in Fig. 1, all preferences that are temporal constraints are expressed using such templates. Thus, the preference *Supplier Provides Price precedes Provide Quote* means that in a plan in which *Provide Quote* and *Supplier Provides Price* are both performed, we prefer that the latter has preceded in the same plan. Similarly, **require** *Printed Receipt Sent after Payment Done Via Credit Card* means that if the goal *Payment Done Via Credit Card* has been satisfied in a plan (through performance of its subtasks) then we prefer that the goal *Printed Receipt Sent* is also satisfied later in the plan.

3.3 Priorities over preferences

Preferences are not always equally desired or important. In a particular situation or context or for a particular stakeholder, a subset of preferences can become more relevant than the others. To see that, let us return to our bookseller example. If an order is small and from a frequent customer, the bookseller is mainly interested in keeping them happy, while ensuring at the same time transaction reliability—

Table 1 Templates for temporal preferences and their meaning

	Template	Intuitive meaning
1.	“ sometime $[X]$ ”	$[X]$ must be eventually satisfied
2.	“ never $[X]$ ”	$[X]$ must never be satisfied
3.	“ $[X]$ precedes $[Y]$ ”	if $[X]$ and $[Y]$ are both satisfied $[X]$'s satisfaction must precede
4.	“ require $[X]$ before $[Y]$ ”	if $[Y]$ is satisfied then $[X]$'s satisfaction must have preceded
5.	“ never $[X]$ before $[Y]$ ”	if $[Y]$ is satisfied then $[X]$'s satisfaction must not precede
6.	“ require $[X]$ after $[Y]$ ”	if $[Y]$ is satisfied then $[X]$'s satisfaction must follow
7.	“ never $[X]$ after $[Y]$ ”	if $[Y]$ is satisfied then $[X]$'s satisfaction should never follow

thought this is not as important given that the order is small. Meanwhile, they also wish to enhance the company's "green" profile that is to project an environmentally friendly image to customers—but again with a lower priority.

Thus, from the entire set of preferences, these three are the ones that the stakeholder may be interested to see satisfied when taking an order of this size from the particular type of customer, while deeming the other preferences less important or completely irrelevant. Subsequently, these preferences that are thought of as 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 preferences are seen in Table 2.

The bookseller may also state that certain constraints on how goal fulfillment is sequenced are relevant for her case. Thus, she may believe that before a quote is provided to the customer the supplier must have provided the supply price to ensure that selling price will be within the desired profit margin. At the same time, but less importantly for a small order and a loyal customer, she also thinks that payment should be received before shipment of the product is performed. She also mentions it would be good but not vital to print a receipt. Finally, she mentions that stock should preferably not be used. Again, these preferences can acquire a weight that expresses their relevant importance as seen in Table 3.

Through this weighted representation, the relative desirability of each of the preferences that are deemed relevant for a particular situation can be modeled and compared. But where do the numbers come from? Our proposal is concerned with modeling and automated reasoning about preferences without being bound to a particular method for priority acquisition. As we will see, the requirements prioritization literature offers a wealth of approaches for elicitation of quantitative expressions of priority among higher-level requirements concepts. In addition to those, our representation and reasoning

Table 2 Prioritizing preferences

<i>Happy Customer</i>	0.6
<i>Transaction Reliability</i>	0.3
<i>Enhance Company's Green Profile</i>	0.1

Table 3 Prioritizing temporal preferences

Require Payment Received before Books Delivered	0.3
Supplier Provides Price precedes Provide Quote	0.4
Sometime Print Receipt	0.2
Never Obtain from Stock	0.1

framework opens the opportunity for iterative revision of an initial set of preferences and priorities thereof, through following repeated cycles of priority specification and testing against the mandatory decomposition. As we discuss below, such an exercise can help the acquisition of a better understanding of the domain and the impact of stakeholder attitudes to lower-level design decisions.

3.4 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 preferences that are satisfied by the plan. For example, plan:

$$[t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_{14}, t_{15}, t_{16}, t_{11}, t_{10}, t_{13}, t_{23}]$$

satisfies two of the three relevant preferences of Table 2, namely *Happy Customer* and *Enhance Company's Green Profile*. The former is satisfied through the inclusion of t_{14} , t_{15} , and t_{16} , which satisfy the goal *Payment Done Via Credit Card*, which, in turn, satisfies all non-optional AND-subgoals of preference *Happy Customer* through the make links. The preference *Enhance Company's Green Profile* is satisfied, because the receipt is not printed (task t_{20}); thus, the negative contribution to that preference goal is avoided, while at the same time the goal receives a make link from task t_{23} . However, no positive contribution link points to the mandatory subgoal of *Transaction Reliability*. Thus, the total score of the particular plan is $0.6 + 0.1 = 0.7$. Should we request printing and submission of a receipt as in plan:

$$[t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_{14}, t_{15}, t_{16}, t_{11}, t_{10}, t_{13}, t_{20}, t_{21}, t_{22}, t_{23}]$$

while the preference *Enhance Company's Green Profile* is now hurt and thus not satisfied in the priority specification, the preference goal *Transaction Reliability* is now satisfied exactly due to the fact that a receipt is printed implying a positive contribution to *Use Robust Legal Documentation*. Thus, the score for this plan, based on the weights of Table 2, would be $0.6 + 0.3 = 0.9$, making it more preferred than the previous one.

For the priority specification of Table 3, the order by which tasks appear in the plan is critical in calculating its score. Thus, plan:

$$[t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_{11}, t_{17}, t_{18}, t_{19}, t_{10}, t_{13}, t_{23}]$$

satisfies only the first and the last of the four components of the priority specification as the payment is received (t_{19}) before book delivery is complete (at t_{13}), and, as desired, the stock is also not used. However, neither the supplier provides price (t_5) before the quote is given to the customer

(t_2) nor is a receipt printed (t_{20}). Thus, the score is 0.4. Should we ensure proper ordering of the tasks *Provide Quote* (t_2) and *Supplier Provides Price* (t_5) as in:

[$t_1, t_4, t_5, t_2, t_3, t_6, t_7, t_8, t_{11}, t_{17}, t_{18}, t_{19}, t_{10}, t_{13}, t_{23}$]

the second component of the priority specification of Table 3 is also satisfied yielding score $0.1 + 0.3 + 0.4 = 0.8$.

As we discussed earlier, different stakeholders in different contexts and situations have an interest in satisfying a different subset of preferences, to which they also give different priorities. The bookseller example of Table 2 reflects the desires of a hypothetical order, where the customer is frequent and the order is small; 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 large order. In this case, the book seller's priorities are to establish *Transaction Reliability* and, to a lesser extent, to maintain *Happy Customer*. Depending on the specific weightings, the plans may involve payment through money order or also printed receipts, depending on the relevant importance of the goal to maintain a green profile. In addition, such an order case may require certain temporal constraints, such as ensuring that the payment is received before the product is shipped.

As we will see, quality and temporal preferences can be combined in a priority specification through an iterative process of automated optimal plan calculation and refinement. Calculating such optimal plans of the mandatory decomposition that best satisfy a given priority specification is possible through the reasoning mechanisms we present below. These mechanisms presume formalization of the visual goal models and the preference specifications into a planning problem specification. In the next section, we discuss this in detail.

4 Formalizing preference goals and priorities

We define the semantics of our diagrammatic language as well as the preference and priority specification via translation to a hierarchical task network (HTN) [11] and PDDL3.0 [10], two popular languages for specifying 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 preference goals and priorities is translated into PDDL3.0. As we will see, by choosing these languages as the basis of our semantics definition, we avail ourselves to the use of a powerful planner to perform

efficient reasoning about solutions that best satisfy given priority specifications. In the rest of the section, we introduce the subsets of the languages we will be interested in, followed by the details of the translation.

4.1 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 $pre-o$, which shows what needs to be true, so that performance of the operator is possible, as well as an effect formula $eff-o$ which shows what becomes true or false upon performance of the action that is modeled by the operator. Those formulae are logical expressions over domain predicates in V . 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 than just “tasks”. HTN tasks cannot be “performed” but recursively reduced into other tasks and/or operators. Different reduction possibilities are modeled through methods. For a method m , we define a parent task for the method $tsk-m$ as well as a set of child tasks $dec-m$. Thus, the method shows how the performance of an HTN task $tsk-m = a$ can be reduced to (i.e., substituted by) the performance of a sequence $dec-m = a_1, a_2, \dots$ of other lower-level tasks or operators. Methods also have preconditions $pre-m$, which mean that the method cannot be used unless that precondition is satisfied.

The HTN domain, which consists of a set of operators and methods describing possible agent actions and allowable combinations thereof, is complemented with an HTN problem specification, which allows the description of a particular problem instance. The problem specification consists of a list of predicates I that are initially (before the performance of any operator) true as well as a list G of high-level HTN tasks that need to be satisfied. The HTN planner reads the domain and problem specifications, and through recursively trying different substitutions of the HTN tasks into subtasks, it searches for allowable sequences of operators that starting from the initial condition I lead to the achievement of all HTN tasks in G . These sequences are HTN plans.

From the PDDL3.0 constructs, we use only the ones that relate to the specification of *PDDL preferences*. 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 of the form *is-violated*(φ), where φ is a logical formula enriched with temporal

operators such as *always*(ϕ), *sometime*(ϕ), *sometime-before*(ϕ_1, ϕ_2), and *sometime-after*(ϕ_1, ϕ_2), with semantics based on Linear Temporal Logic [10]. The synergy between HTN and PDDL is achieved through the fact that these formulae ϕ, ϕ_1, ϕ_2 , etc can be grounded on HTN domain predicates of V . In this way, PDDL preferences can be used to express evaluation conditions of plans from domains written in HTN. The details about how this synergy is achieved can be found in [9].

Expressions of priorities among 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 of the form $f = w_1 \times is-violated(\phi_1) + w_2 \times is-violated(\phi_2) + \dots + w_n \times is-violated(\phi_n)$, where f is the metric function, ϕ_i are PDDL preference formulae, w_i are numerical weights, and *is-violated*(\cdot) is a function that returns 0.0 if ϕ is satisfied and 1.0 otherwise—thus it is an expression of penalty. Given a plan, the more of the constituent PDDL preferences are satisfied by the plan, the lower the value of the metric f will be, to a degree that depends on the weights w_i of the individual preferences. Thus, specification of PDDL preferences and metrics thereof are included as part of the planning problem specification, in order to instruct the planner to find solutions that achieve the specified goals starting from the given initial condition while, at the same time, optimizing the given metric.

4.2 Translating to HTN and PDDL

We now show how our visual representations of goals and priorities can be translated into HTN and PDDL 3.0. Roughly, the mandatory decomposition is translated into a set of HTN operators, tasks, and methods, while the set of preference goals as well as the priorities are translated into PDDL 3.0 preference constraints and metrics. To facilitate comprehension of the rules, we will use the small translation examples of Fig. 2.

4.2.1 Eliminating optional subgoals

To translate the mandatory decomposition, we first transform the goal model in order to eliminate the optional subgoals. As seen in frame (A) of Fig. 2, for each subgoal g^o that is designated as optional in the goal model, 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{npr} links stay connected with g^o . The newly introduced task t^d is a “dummy” task that is removed from the plans

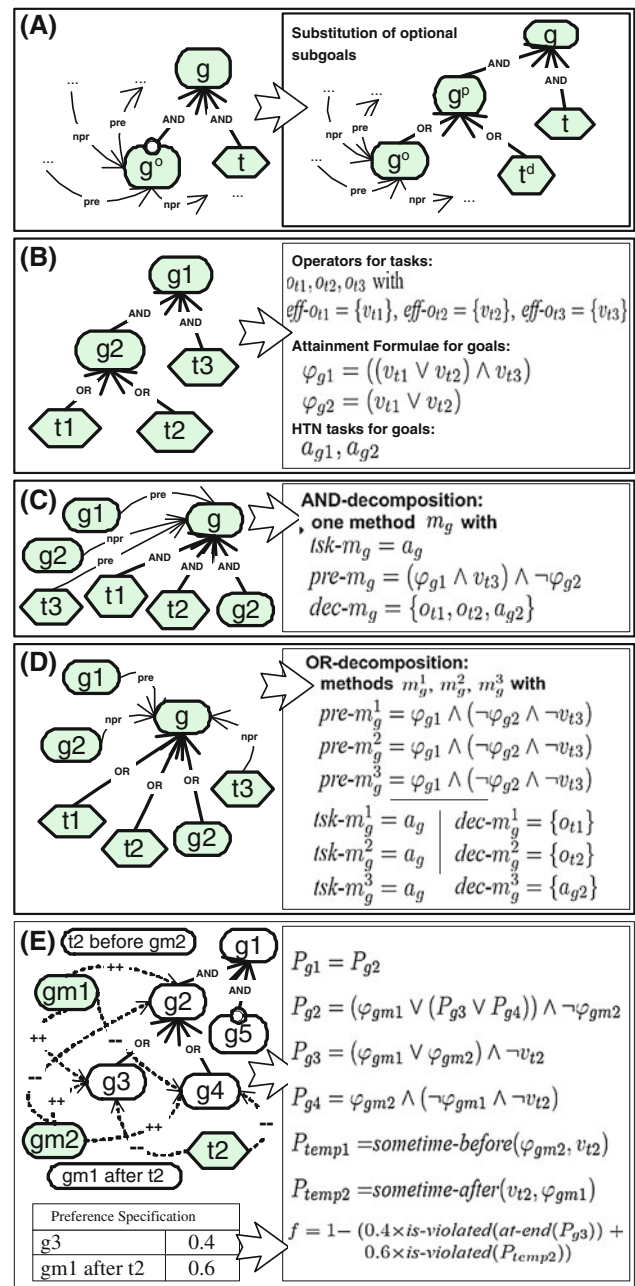


Fig. 2 Translation rules by example

that the planner returns—in the rest of our presentation, we will assume that this post-processing step has preceded when presenting example plans. The mandatory decomposition is now ready to be translated into an HTN specification.

4.2.2 Translating the mandatory decomposition

To construct the HTN specification, we work as follows:

- For each leaf-level task t :

- Introduce an HTN domain predicate v_t . Call this the *task performance predicate*.
- Introduce an HTN operator o_t .
- Set the effect of o_t to be $eff-o_t = v_t$
- Set the operator precondition $pre-o_t = \phi_t^{pre}$, where ϕ_t^{pre} is the *precondition formula* for task t , which we will define below.

Intuitively the task performance predicate v_t represents the fact that the task t has been performed, hence its position as an effect.

- For each hard goal g of the goal model:
 - Introduce an HTN task a_g .
 - Introduce the *attainment formula* ϕ_g of g , grounded on task performance predicates v_t and reflecting the structure of the subtree rooted on g . To construct ϕ_g work as follows. Depending on whether g is OR- or AND-decomposed into mandatory subgoals g_1, g_2, \dots replace g with $(g_1) \vee (g_2) \vee \dots$ or $(g_1) \wedge (g_2) \wedge \dots$, respectively. Recursively repeat replacement each g_i with the conjunction or disjunction of g_i 's children, depending on decomposition type. At any point in the recursion, if a child of the goal in consideration is a task use the task performance predicate v_t in the replacement. Terminate the recursion when all individuals in the formula are task performance predicates—and no more replacements are possible. The above steps for generating operators, task performance predicates, HTN tasks and attainment formulae can be seen in the example of frame (B) of Fig. 2.

- Depending on g 's decomposition type, introduce one or more HTN methods. More specifically:
 - If g is AND-decomposed into goals g_1, g_2, \dots and tasks t_1, t_2, \dots , introduce one method m_g with:

- $tsk-m_g = a_g$,
- $pre-m_g = \phi_g^{pre}$, and
- $dec-m_g = \{a_{g1}, a_{g2}, \dots, o_{t1}, o_{t2}, \dots\}$,

where a_{g1}, a_{g2}, \dots and o_{t1}, o_{t2}, \dots are the HTN tasks and operators that correspond to goals g_1, g_2, \dots and tasks t_1, t_2, \dots , respectively. Formula ϕ_g^{pre} is the precondition formula for goal g which we introduce below.

- If g is OR-decomposed into n goals or tasks $h_1, h_2, \dots, h_i, \dots, h_n$ introduce n methods m_g^i , each corresponding to each child h_i and with:
 - $tsk-m_g^i = a_g$,
 - $pre-m_g^i = \phi_g^{pre}$, and
 - $dec-m_g^i = a_{h_i}$, if h_i is a goal or $dec-m_g^i = o_{h_i}$ if h_i is a task. In the above, ϕ_g^{pre} is again the

precondition formula, which we introduce below. Notice that, this time, each HTN task is decomposed into exactly one HTN task or operator.

In frames (C) and (D) of Fig. 2, the above translation of decompositions into HTN methods is exemplified.

- As we saw above, a precondition formula ϕ_h^{pre} is defined and used as an operator precondition $pre-o_h$ if h is a task or as method precondition(s) if h is a goal. For every goal or task h , we construct this precondition formula ϕ_h^{pre} as follows. Let N be the set of all goals and tasks h_e for which $h_e \xrightarrow{mpr} h$, P the set of all goals and tasks h_p for which $h_p \xrightarrow{pre} h$. The precondition formula ϕ_h^{pre} for goal or task h is then defined as:

$$\phi_h^{pre} = \bigwedge_{h_p \in P} \phi_{h_p} \wedge \left(\bigwedge_{h_e \in N} \neg \phi_{h_e} \right)$$

where ϕ_{h_p}, ϕ_{h_e} are attainment formulae or task performance predicates associated with h_p, h_e depending on whether they are goals or leaf-level tasks, respectively.

4.2.3 Translating preferences and priorities

Let us now see how preferences and priorities thereof at the goal level are translated into PDDL preferences and metrics. An example can be seen in frame (E) of Fig. 2. Recall that a priority specification is a set of elements $h_1, h_2, \dots, h_i, \dots$ each with an assigned weight w_{h_1}, w_{h_2}, \dots (e.g., Table 2). Items h_i can be either (a) simple preference goals or (b) preferences expressed as temporal constraints over goals or tasks of the mandatory decomposition. We discuss how we translate each h into a PDDL preference ϕ_h in each of these cases below:

- If h is a simple preference goal, then let $\phi_h = at-end(P_h)$ —where the temporal operator $at-end(\cdot)$ is true if the operand is true in the final state. P_h is calculated as follows. Let P and N be the sets of all elements h_p and h_e of the mandatory decomposition such that $h_p \xrightarrow{++} h$ and $h_e \xrightarrow{-} h$, respectively. Then, P_h is constructed as follows—with P_{ch} being explained below:

$$P_h = \left(\bigvee_{h_p \in P} \phi_{h_p} \vee P_{ch} \right) \wedge \bigwedge_{h_e \in N} \neg \phi_{h_e}$$

The term P_{ch} is calculated recursively as follows. If h is OR-decomposed into subgoals h_1, h_2, \dots then $P_{ch} = P_{h_1} \vee P_{h_2} \vee \dots$. Similarly, if h is AND-decomposed into subgoals h_1, h_2, \dots then $P_{ch} = P_{h_1} \wedge P_{h_2} \wedge \dots$ —excluding subgoals with the optional subgoal designation. The terms P_{h_i} are the corresponding preference subformulae for each

Table 4 Template Semantics

Template	PDDL 3.0 preference
“sometime h_1 ”	<i>sometime</i> (ϕ_{h_1})
“never h_1 ”	<i>always</i> ($\neg\phi_{h_1}$)
“ h_1 precedes h_2 ”	<i>sometime-before</i> ($\phi_{h_1} \wedge \phi_{h_2}, \phi_{h_1} \wedge \neg\phi_{h_2}$)
“require h_1 before h_2 ”	<i>sometime-before</i> (ϕ_{h_2}, ϕ_{h_1})
“never h_1 before h_2 ”	(cf. “ h_2 precedes h_1 ”)
“require h_1 after h_2 ”	<i>sometime-after</i> (ϕ_{h_2}, ϕ_{h_1})
“never h_1 after h_2 ”	(cf. “ h_1 precedes h_2 ”)

subgoal h_i . If h is not decomposed, the term P_{ch} is *false*. Thus, recursive construction of P_{ch} can be performed starting from the leaf level and moving toward the root. Note also that possible temporal constraints that could appear at the leaf of these preference hierarchies are excluded from this calculation and added manually as separate prioritization elements—this to avoid nesting of temporal operators.

– If h is a temporal constraint over elements h_1, h_2, \dots of the mandatory decomposition then translation depends on the template that is being used. The intuitive meaning of the seven templates we introduced in Table 1 in terms of PDDL preferences can be seen in Table 4.

On the table, ϕ_{h_1} and ϕ_{h_2} are, as above, HTN attainment formulae for goals or HTN performance predicates for tasks. In reading the above, one must keep in mind that attainment formulae are satisfied one at a time and in a monotonic way, i.e., once true they do not become false within the context of a plan.

4.2.4 Constructing the metric

The PDDL metric is constructed by first reading the priority table (e.g., Tables 2, 3) and formulating it as a linear combination, where each constituent preference goal is translated into the appropriate PDDL 3.0 preference formula as described above.

Recall that a PDDL metric is of the form: $w_1 \times is-violated(\varphi_1) + w_2 \times is-violated(\varphi_2) + \dots + w_n \times is-violated(\varphi_n)$, where φ_i are PDDL preference formulae and w_i numerical weights. Note that the function *is-violated*(φ) (which, again, equals to 0.0 if φ is satisfied and 1.0 otherwise) must be used to comply with PDDL metric formation rules. Thus, given the preference table containing preference elements h_1, h_2, \dots, h_n , the corresponding PDDL metric is:

$$f = \sum_i(w_i) - \sum_i(w_i \times is-violated(\phi_{h_i}))$$

where each P_{h_i} is the PDDL preference formula that corresponds to preference element h_i .

To see the rationale behind the first term, notice that the PDDL metric construction rule that all constituent terms must be in the form *is-violated*(φ) implies that the metric yields a penalty rather than a reward value. To reverse that, in the above formula, we subtracted the linear combination from the sum of all weights. This way, the more constituent preferences are satisfied the higher the score.

4.2.5 The planning problem

The last step of the translation is the identification of the planning problem. As we saw, the HTN planning problem consists of a set I of domain predicates in V that are true in the initial state, as well as a list G of high-level tasks that need to be satisfied. In our case, we set empty initial conditions $I = \{ \}$, while the HTN goal is set to the root goal of the mandatory decomposition, that is $G = \{a_{g_r}\}$ where a_{g_r} is the HTN task representing the root goal g_r .

5 Reasoning about preferences

5.1 Integrating the planner

By translating the goal formalisms to HTN and PDDL—scripts have been developed and extensively used for the purpose—it is possible to use an HTN preference-based planner to identify plans that optimize for the priorities among preference goals. To this end, we employ HTNPlan-P [9], an extension of the popular SHOP2 HTN planner [11] 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, as we saw, includes the definition of initial conditions for the domain predicates and the planning goal, which is normally a top-level HTN task.

The HTNPlan-P planner searches through recursive reductions of the top-level HTN task into subtasks or operators, with the objective of finding a sequence of ground operators (actions) that satisfies the various preconditions and effect constraints while optimizing the metric function that encodes the prioritized PDDL preferences. The search for a suitable plan can be understood as a variant of a sequential plan optimization problem with the task decomposition serving to constrain the legal action sequences that the planner should consider. The optimization of preferences is achieved by branch-and-bound heuristic search through this induced search space. HTNPlan-P employs several heuristics that have been tailored to this task. In particular, in order to deal with temporally extended preferences using heuristic search, preferences using LTL temporal modalities are transformed into final-state

preferences by exploiting a correspondence between LTL and finite state automata. The planner performs an incremental, best-first search guided by an inadmissible heuristic. Partial plans that have no prospect of being better than the current best plan are pruned. In this way, the search space can be pruned and searched in its entirety, leading to a proof of optimality of the result. The combination of the expressiveness supported by HTNPlan-P and the effectiveness and sophistication of the plan generation approach make this tool very amenable to our purposes.

In our context, the planner is given as input the translated domain theory and priority specification in HTN and PDDL, respectively. The planner returns a sequence of operators that suffice for the performance of the top-level HTN task that corresponds to the root goal, 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.

It is interesting to understand why we use a planner and when it may be necessary in rendering solutions to goal requirements problems. Solutions in goal models are typically specified as configurations of tasks with no explicit temporal dependencies. In such systems, stakeholder requirements can be modeled without specification of system dynamics, and solutions to requirements can be determined using automated constraint solvers. Giorgini et al., for example, employ such a system for statically reasoning about partial goal satisfaction [3]. Here, we argue that some requirements are inherently temporal and require specification of certain aspects of system dynamics. Indeed, several efforts for modeling such temporal relations in goal models can be found in the literature [2, 16, 17]. In simple cases where goal or task orderings do not rely on any notion of state or can be modeled without explicit pre- and post-conditions, a constraint solver can be argued to be adequate to efficiently compute a solution. However, in our case, both the expressiveness needs and the requirement for satisfaction or optimization of temporal constraints over goals and tasks necessitate the use of a planner. By having defined the semantics of our preference language in PDDL, we are able to use HTNPlan-P as described above, as well as any efficient PDDL-compliant HTN planners that will emerge in the future.

5.2 Finding preferred plans

Returning to our bookseller example, given the preference specification of Table 2, the planner returns the plan:

$$[t_1, t_2, t_3, t_{14}, t_{15}, t_{16}, t_{23}, t_{20}, t_{21}, t_{22}, t_9, t_{11}, t_{10}, t_{13}]$$

as optimal, with score 0.9. The above satisfies the goal to have *Happy Customer*, which is achieved through the convenience of using credit cards. Transaction reliability is

also met by submitting printed receipt. The latter, though, causes the preference *Enhance Company's Green Profile* to be compromised. If we reverse the weights of that last goal with the goal *Transaction Reliability*, the result does not contain a printed receipt:

$$[t_1, t_2, t_3, t_{14}, t_{15}, t_{16}, t_{23}, t_9, t_{11}, t_{10}, t_{13}]$$

Similarly, the priority specification over temporal preferences of Table 3 gives the following plan with maximum score 1.0:

$$[t_1, t_4, t_5, t_2, t_6, t_7, t_8, t_3, t_{14}, t_{15}, t_{16}, t_{23}, t_{20}, t_{21}, t_{22}, t_{11}, t_{10}, t_{13}]$$

Observe how the payment is received (at t_{16}) before the books are delivered (at t_{13}), the price is provided by the supplier (t_5) before a quote is given to the customer (t_2), stock is not used (t_9), and eventually a receipt is printed (t_{20}).

But what if would like to combine quality and temporal preferences in one unique priority specification? The example of the next section shows how this can be done through a process of gradual refinement and trial using the tool.

5.3 Gradual specification and refinement

The visualization that the planner gives us of how preference specifications are interpreted into operational designs can be used for iterative refinement and validation of both the preferences and the goal model. Thus, after looking at an initial result, the stakeholders may wish to revise their specification by adding new constraints or relaxing the existing ones. This may lead to a new preference specification, which will, in turn, yield more plans for further investigation and revision of the preferences or even the goal model. Such an exercise can be useful in the early requirements stage, when stakeholders need to explore and understand different solutions for their goals under different envisioned circumstances which, in turn, affect their priorities in different ways.

To see an example of this process, assume that the bookseller wants to find a good plan for serving a local customer. At first, they do not specify any preferences other than that they would like to increase *Customer Convenience* and that they do not want to use any stock as they do not have infrastructure for that. This results in a preference with two components. The planner returns the following plan for this preference:

$$[t_1, t_2, t_3, t_{14}, t_{15}, t_{16}, t_{23}, t_4, t_5, t_6, t_7, t_8, t_{11}, t_{10}, t_{13}]$$

After looking at the output, the bookseller asks whether it is necessary that shipment (t_{10}) must happen after charging the credit card (t_{16}). For the particular kind of customer, she is ready to ship the product once the credit

card authorization has been performed. So she asks whether a scenario where the product is shipped before charging the credit card and after acquiring credit card authorization is possible and at what utility cost. In that case, to the original components of the preference specification, we add two more components that require delivery to courier to be performed before charging the credit card (t_{10} **precedes** t_{16}) but after acquiring an authorization (t_{15} **precedes** t_{10}). We distribute weights equally among the four preference components. The planner is indeed able to find a plan that satisfies all components:

$[t_1, t_2, t_3, t_{14}, \mathbf{t}_{15}, t_4, t_5, t_6, t_7, t_8, t_{11}, \mathbf{t}_{10}, \mathbf{t}_{16}, t_{13}, t_{23}]$

The bookseller is now asking whether, in addition, a printed receipt can be inserted in the shipment. To see what this would entail, we add one more component to the preference specification, namely an existential property on task *Place Receipt in Shipment* (**sometime** t_{12}). This time, the planner cannot satisfy all components and returns this plan:

$[t_1, t_2, t_3, t_{14}, t_{15}, t_4, t_5, t_6, t_7, t_8, \mathbf{t}_{16}, t_{23}, t_{20}, \mathbf{t}_{12}, \mathbf{t}_{10}, t_{13}]$

The plan does involve placing a receipt inside the shipment (t_{12}) but it does not allow delivery to courier (t_{10}) before the credit card is charged (t_{16}). Clearly, by looking at the model, the business rule that no receipt can be considered unless full receipt of payment is performed (expressed as a precedence link between the corresponding high-level goals) makes the desire to ship before charging the credit card and the desire to place a receipt inside the shipment to conflict. “So how can I avail a printed receipt to my customer without violating our previous temporal desires?” the bookseller may ask. We can express this query by replacing the lastly added existential preference for t_{12} with one that requires that just a receipt is printed (**sometime** t_{20}). The planner returns the following plan with maximum score (i.e., all components are satisfied):

$[t_1, t_2, t_3, t_{14}, t_{15}, t_4, t_5, t_6, t_7, t_8, t_{11}, \mathbf{t}_{10}, t_{16}, t_{13}, t_{23}, t_{20}, \mathbf{t}_{21}, t_{22}]$

Thus, the printed receipt can be sent separately if the precedences are to be satisfied. From here, the bookseller may reject this solution as expensive by adding an absence component to the tasks that refer to submission of separate receipt (t_{21} , t_{22}) and observing what the planner returns then. If she did that however, she would realize that the planner would not find an optimal solution, meaning that she will need to weight the importance of shipping before fully charging the credit card, sending an included receipt and not sending a separate receipt. Or she would need to reconsider the rule that no receipt is provided before the customer is charged, which, as we saw, is expressed as a precedence in the mandatory decomposition.

Thus, by following this progressive refinement of the preference specification, we can support an iterative process for preference acquisition, whereby the result of an initial preference specification is used to trigger introduction of additional or removal of existing constraints. As we discuss later, this practice of iteratively specifying preferences led to (a) improvement in the representation and correction of modeling errors and (b) acquisition of a better understanding of the domain we were investigating. Model improvement came in the form of, for example, correcting invalid precedences or adjusting preference versus mandatory characterizations. In terms of domain understanding, we felt that this gradual refinement and reasoning exercise triggered deeper and more detailed thinking about the domain. We return to these experiences later in the paper.

6 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 how a basic object model can be created within HTN/PDDL, how action parameters and domain predicates can be used to allow for richer representation of the domain and its states, and how preferences can be consequently enriched to support more expressive reasoning at an instance level.

6.1 Representing types

In many requirements engineering applications, it is useful to distinguish types of objects and to specify goals and preferences over individuals within these types. Returning to our bookseller example, we may, for instance define Customers, Orders, Employees or Delivery Companies as distinct types, different instances of which can then be declared in HTN problem definitions.

In particular, we use domain predicates such as *isCustomer(o)* or *isEmployee(o)* in order to represent that o is an object of the corresponding type. Hierarchies of such types can be constructed using *HTN axioms*. In the simplest case, axioms come in the form $a \Leftarrow \varphi$ where a is an n -ary domain predicate and φ a logical expression of such predicates. In the frame (A) of Fig. 3, such axioms are

shown. Thus, axioms such as $isCompany(X) \Leftarrow isSupplier(X)$ introduce a specialization relationship, while the use of more complex logical formulae on the right-hand side allows for more elaborate type definitions. Note that the type and object structures we define in HTN must also be defined in PDDL, which actually happens in a very similar way.

6.2 Adding operator and method parameters

At a second stage, HTN/PDDL operators and HTN methods can be redefined to include parameters of the specified types. Thus, the action *deliverToCourier*, derived from task t_{10} of the goal model can be extended to specify whose company's order it is, which courier company should be used and what the quantity is, resulting in an operator of the form *deliverToCourier(Company, Courier, Quantity)*. In frame (B) of Fig. 3, the operator with its pre- and post-conditions (effects) can be seen. Obviously, the parameters of the operator are bound to the parameters of the pre- and post-conditions. Thus, the operator *deliverToCourier(theCube, hDL, large)* will require that *reserved(theCube, large)* is already true and that the 3-ary domain predicate *deliveredToCourier(theCube, hDL, large)* will become true right afterward, where *theCube*, *hDL* and *large* are concrete objects whose definition we describe below.

Higher-level methods can also contain parameters and can, moreover, serve as ways by which parameters of operators or other methods can be synchronized (i.e., bound to the same objects). For instance, the HTN task *quoteGiven*, derived from the corresponding goal in the goal model, can be parameterized to also specify the customer to which this is done by writing *quoteGiven(Customer)*. Through defining the appropriate method, as seen in frame (B) of Fig. 3, this HTN task is further decomposed into *customerRequestsQuote(Customer)* and *provideQuote(Customer)* which are HTN operators. Use of the method implies that the task and the operators are bound to the same *Customer* object. Methods also have parameter-specific preconditions—in the previous example, the method is considered only if *goodCustomer(Customer)* is satisfied.

Alternatively, the planner itself may search for bindings that satisfy given conditions. Consider the method that decomposes the task *booksDelivered(Customer, Quantity)* into the operators/tasks *deliverToCourier(Customer, Courier, Quantity)*, *courierDeliversToCustomer(Customer, Courier, Quantity)*, and *handleReceipt(Customer)* as seen in Fig. 3. While HTN will assume that the parameters *Customer* and *Quantity* of the parent task and each of the subtasks will be bound to the same objects, the parameter *Courier* does not match with any of the parameters of the parent task. However, the precondition defines that

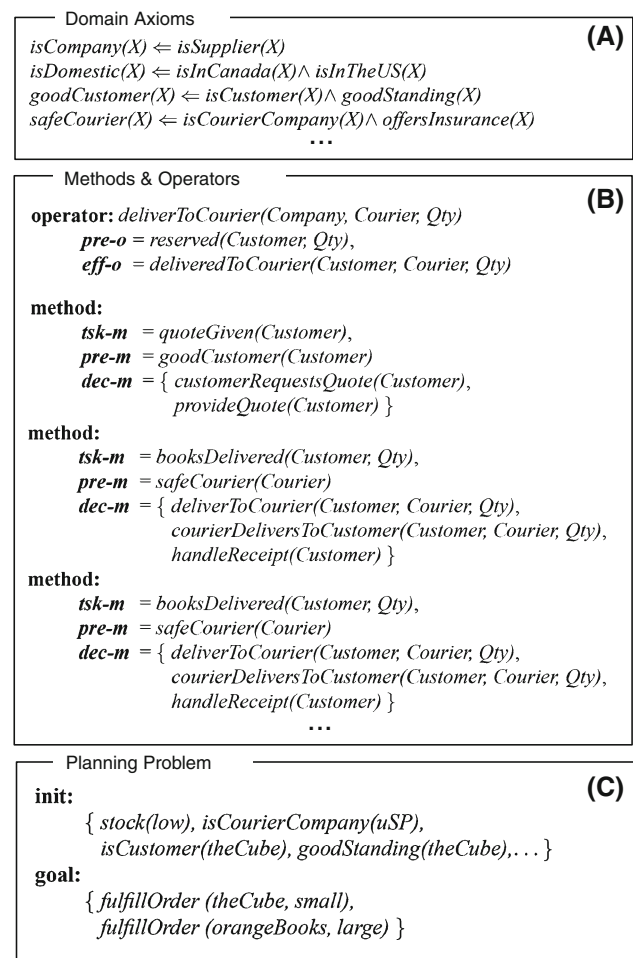


Fig. 3 Adding expressiveness to the HTN output

safeCourier(Courier) must be satisfied. Thus, the planner, when substituting the task with the subtasks/operators, will try different parameter instances of *Courier* that satisfy the precondition based on what instances have been defined in the initial conditions.

6.3 Instance-level preference analysis

Typing and parameterization in the construction of the domain theory allows us to perform instance-level reasoning about goal and preference satisfaction for a particular domain. In our bookseller example, we can specify a scenario in which the bookseller co-operates with two different bookstores, called, say, “The Cube” and “Orange Books”. We can also assume that there might be a selection of suppliers, say, “Johnson” and “Laurier” or courier companies such as “DLH” and “USP” and that the warehouse stock for a particular book is low. These type instances and problem parameters are defined as initial conditions of the planning problem as seen in frame (C) of Fig. 3. The bookstores “The Cube” and “Orange

Books” may both need to have an order fulfilled for a specific book and of a specific quantity. This implies that the goal specification is now a list of two high-level HTN tasks to be resolved as seen again in frame (C) of Fig. 3.

When the planning domain and problem have been constructed this way, different assumptions about qualities of each of the involved domain objects (customers, couriers etc.) may force us to consider different ways for achieving the root goals. PDDL preferences can be formulated in a way that they are sensitive to these qualities. In our bookseller example, we may want to say that “*Orange Books*” should never be served by “*DLH*”—because they have requested so for some reason. The PDDL preference to express this would be:

$$\text{always}(\neg \text{courierDeliveredToCustomer}(\text{orangeBooks}, \text{DLH})) \quad (1)$$

At the same time, it might be the bookseller’s preference that “*large orders are not to be handled by “USP”*”, due to e.g., cost considerations. This would in turn be expressed as follows:

$$\text{always}(\forall X.\text{isCustomer}(X) (\neg \text{deliveredToCourier}(X, \text{uSP}, \text{large}))) \quad (2)$$

In the above example, we are making use of universal quantification to generalize for all instances of a type. In a similar manner, we may want to pose that if there is an order from The Cube, then Johnson Inc (a supplier) must supply and, moreover, no quote must be given to The Cube unless Johnson has provided their price.

$$\text{sometime-before}(\text{quoteProvided}(\text{theCube}), \text{supplierProvidesPrice}(\text{johnsonInc})) \quad (3)$$

The presence of multiple instances of the root level goal with different parameters brings up an issue of priority and synchronization, potentially in the presence of commonly accessed resources. In the bookseller example, we may wish that “*if the stock is low do not ship for Orange Books unless you have shipped for The Cube*” which is represented using the following formula:

$$\text{always}(\forall S.\text{isCourierCompany}(S), \forall Q.\text{isQuantity}(Q) (\text{stock}(\text{low}) \Rightarrow (\text{deliveredToCourier}(\text{orangeBooks}, S, Q) \Rightarrow \text{deliveredToCourier}(\text{theCube}, S, Q)))) \quad (4)$$

At the same time, different general policies regarding use of stock can be expressed, depending again on its level. Thus, to prevent use of the stock when the levels are low, we can construct the following:

$$\text{always}(\forall X.\text{isCustomer}(X), \forall Q.\text{isQuantity}(Q) (\text{stock}(\text{low}) \Rightarrow \neg \text{useStock}(X, Q)))$$

To the above, though, we may want to add an exception for Orange Books and ship to them before contacting any supplier:

$$\text{sometime-before}(\forall S.\text{isSupplier}(S) \text{supplierContacted}(S) \wedge \text{customerPlacedOrder}(\text{orangeBooks}), \forall C.\text{isCourierCompany}(C), \forall Q.\text{isQuantity}(Q) \text{deliveredToCourier}(\text{orangeBooks}, C, Q))$$

When aggregated, offering more weight to the latter preference will effectively allow the planner to override the previous rule in favor of the more specific one. If the size of the order, however, cannot be afforded due to very low stock, then the planner, having failed to satisfy the exception, will opt for satisfying the general rule.

Note that the rationale behind the above preferences can be traced back to the goal model, by looking at the contributions among hard elements and/or quality preferences, which have both been refined here into more expressive formulae. For example, the fact that use of stock allows for quicker turnaround or that preventing use of stock altogether allows for zero storage costs is already presented in Fig. 1 through contribution links from the hard goal to the corresponding preference soft goals. Here, more detail is given regarding preferred use of stock, possibly referring to particular instances. While precise optimization of stock management falls in the scope of specialized quantitative methods (which our framework does not intend to replace), through preference analysis, we are able to understand early in the requirements phase how preferred strategies and tactical rules affect the resulting process.

To perform analysis in the expressive case, we also need to customize both the initial conditions and the planning goal. Thus, let us assume a scenario where both The Cube and Orange Books have placed a small and a large order, respectively, for a book whose stock is low (cf. frame (C) of Fig. 3). Also, let us combine the first four among these preferences (1)–(4) into a preference specification with weights 0.2, 0.1, 0.4, and 0.3, respectively. Given these inputs, we get the result of Table 5—most of the tasks have been pruned for simplicity. The plan has a score 0.9, meaning that preference (2) has not been satisfied. Indeed, a closer look at preferences (1) and (2) reveals that they are conflicting for large orders placed by The Cube, given that there are only two courier companies.

Through the above expressive modeling and reasoning exercise, stakeholders are able to envision concrete

Table 5 Example of plan in instance-level analysis

```

contactSupplier(johnSonBooks),
  supplierProvidesPrice(johnSonBooks),...
    placeOrderToSupplier(johnSonBooks)...
provideQuote(orangeBooks),
  ..., receiveMoneyOrder(orangeBooks),...
    provideQuote(theCube), customerPlacesOrder(theCube),
  ...,
deliverToCourier(theCube,dLH,small),
  deliverToCourier(orangeBooks,uSP,large),...
  ...

```

scenarios and explore preferred ways to go about fulfilling their goals. This level of formalization approaches the one that frameworks such as KAOS [2] or Formal Tropos [16] employ. Our exploration of HTN at this level of expressiveness showed its appropriateness for formalizing and reasoning about requirements, both because it readily allows combined representation of important requirement views such as goal and type hierarchies and because it is backed by powerful reasoners that allow for meaningful interaction with the constructed model. We intent to explore more uses of HTNs and preferences to address requirement analysis problems.

7 In practice

In addition to the bookseller example which we have been discussing so far, we have also experimented with our framework in three other domains: the health care domain, involving nursing processes, the ATM domain, exploring different behavioral designs for an automated teller machine, and the classic meeting scheduler domain, investigating different ways to schedule meetings. The models for these domains were built only for the purpose of this exploration, and the analysis process did not involve external participants. However, all applications, particularly the first and the third, were partially supported by real-world data.

We present here some experiences from those applications, focusing on examples that show how the need to express preferences and priorities emerged and how our specification and reasoning process supported understanding of those priorities and improvement in the models themselves.

7.1 On the presence of goal-level priorities

Our first concern is whether preferences and priorities thereof are concepts relevant to goal analysis and worth being modeled and analyzed. In our applications, interesting design dilemmas emerged which indeed necessitated preference and priority specification. In the meeting

scheduler case, for example, we modeled different ways by which a meeting can be organized—assuming an academic context of which we are more familiar. In this case, one of the examples of dilemmas of temporal nature was the question of when to announce a meeting with respect to when the meeting room has been confirmed: should the meeting initiator wait for room availability confirmation, risking a late meeting announcement? Or should she announce a meeting before room availability has been confirmed, risking retracting the original announcement should the room turn out to be unavailable? Our understanding based on our experience in our workplaces is that different kinds of meetings (e.g., formal thesis defense vs. informal reading group), number and kind of participants as well as general perception of the meeting room demand (of which no real data are normally available) influence the decision and thus the preference. The value of reasoning about this at requirements level is that it exactly triggers this discussion and necessitates further elaboration of influencing factors and design measures. Thus, in an automated meeting scheduler, for example, modules that measure room demand or meeting formality classifications and subsequent announcing types and constraints might need to be introduced to enforce one or the other priority.

In the ATM case, where different behavioral designs for performing transactions on an automated teller machine were explored, the influence of priorities over temporal preferences to design was more direct. When in the process should the user give their password? When should they get their card back (e.g., before or after receiving their receipt?). What if the ATM is adjusted not to provide printed receipt—should a warning be displayed and when? Here, the priority specification was used to specify different scenarios of ATMs based on e.g., their maintainability and traffic. Thus, for an ATM in a busy mall, efficiency is the prime preference implying disabling of most ATM functions other than withdrawal and balance viewing—withdrawal options are also restricted to small predefined amounts. If the ATM is inside the owning bank, such restrictions have less priority. In general, in the ATM example, preferences did not compete within one priority specification. Rather, different ATM scenarios yielded different selections from the space of preferences, which were then combined in the priority specification, in a manner similar to the one presented in Sect. 5.

More trade-offs appeared in the nursing example, where alternatives for supporting nursing activities were investigated, partially based on real-world data collected from a geriatric assessment unit. One question for example was whether a wearable device for establishing a voice link between nurses and patients was useful. On the one hand, this would occasionally save the nurses from walking to a patient's room when the patient does not need his help but

just has a question. On the other hand, nurses may reject the idea of carrying wearable devices or see them as encouraging unnecessary nurse calls. At a higher level, the trade-off appears as one between nurse comfort and patient experience. Subsequent analysis, i.e., exploring different plans for the nurse to attend to patient calls based on the particular prioritization, again offers a good illustration of the implications of the conflict and prompts for design considerations—e.g., alternative ways to establish voice connection.

Our understanding from all applications is that prioritization of preferences of the types we have been discussing is more likely if a *context* is given. Thus, in the Meeting Scheduling example, eliciting the relative importance between *Quick Scheduling* and *Formality* is not possible unless one is also told what kind of a meeting it is, who participates or how urgent and important it is. Or, in the nursing example, the priority of quality preference *Increase Nurse's Productivity* versus preference *Nurse Comfort* depends on several factors including the type of institution and management or the status of employment relations—the priority cannot be given without explicating these contextual aspects.

7.2 On iterative refinement and the use of numbers

While the need to specify preferences and priorities thereof was established in our applications, did the numerical prioritization approach turn out useful and how? How burdensome is the iterative process—does it converge? Also is exhaustive identification of situations where different preferences may be relevant possible?

Let us first observe that the use of numerical weights for expressing priorities is not uncommon in requirements engineering. In the Analytic Hierarchy Process (AHP), for example, pairwise comparisons of high-level expressions of requirements are performed [8]. In the result, numbers constitute a measure of relevant benefit associated with the satisfaction of each requirement. The general AHP literature shows that priorities over large varieties of concepts and of various levels of abstraction can effectively be elicited through pairwise comparisons [19, 20]. Elsewhere, it is suggested that stakeholders can even assess absolute importance of features in e.g., a 1–10 scale [21, 22]. Similar quantifications of benefit are widely used in decision theory through utility measures represented in e.g., utility matrices (cf. [23]).

Nevertheless, such quantifications have a specific purpose and use in requirements prioritization, which does not necessarily assume accuracy of the numeric result. For example, in Karlsson and Ryan's application of AHP [8], a descriptive approach (a plot) allows stakeholders to visually reason about the costs versus the relative benefits of

individual requirements rather than attempting an exact interpretation of the numbers. Similarly, in our work, we use the numbers in order to explore the operational result (the plans) that preferences might imply, without having to reason about the actual values. The result is to be used in an exploratory manner to trigger revisions of both the preference values and the underlying models (cf. Sect. 5), achieving thereby a better understanding of the requirements problem and the nature of the conflicts among stakeholders and their preferences.

In our applications, evidence of such improvement emerged in several forms. In the nursing example, the conflict between the preferences of the patients and management and the preferences of the nurses became clear early. Assigning different priorities to the two, we could observe different ways by which nurses could go about their work—some ways more preferred and some ways less preferred by them. The result triggered thinking about solutions that might satisfy all parties. Even if a third solution is absent, we at least become aware of the preferred solution given the dominance of a certain group of stakeholders and of the impact of that choice to the less dominant group. In the ATM example, larger numbers of desired constraints and qualities were brought together with various weights in one priority specification with the expectation that it is maximized. Should it not be maximized, e.g., due to conflicting precedence preferences, the loss in score value offered an indication as to which component was failing. Changing (e.g., swapping) importance weights and observing the resulting plan offered an indication of what conflict caused the failure. In all cases, the absolute value of the weights was not as important as which was greater than the other. Thus, the incorporation of numerical weights in our framework did not seem to obstruct our application in any way, given the particular way we used and interpreted them.

The answer to the question of burden and convergence, i.e., whether the iterative process—when chosen and applied—is costly and whether it leads to a stable conclusion seems to strongly depend on the needs of the application at hand. As we saw, our representation and reasoning framework can be used in different ways. One way is to decide that one cycle (possibly followed by a corrective follow-up one) suffices for concluding to a prioritization profile and the tool is used to calculate the corresponding preferred plan. To ensure greater accuracy of the numeric prioritization result, this option may suggest pairwise comparisons and subsequent approximation of the eigenvector of the resulting comparison matrix, as proposed in AHP. It also implies confidence that the model has been correctly elicited and constructed, the domain is well understood, and AHP yields an accurate prioritization profile (e.g., with a good consistency ratio). Following this

approach, our representation and reasoning framework is used to demonstrate how an established prioritization result translates into a solution to the requirements problem.

On the other hand, if users of the framework are looking to improve/enrich their model and better understand an otherwise unfamiliar domain, they can resort to the iterations we described in Sect. 5.3. In that approach, numbers may be assessed through straight ad hoc assessment of priority values (i.e., not pairwise but as in [21]) as accuracy is expected to emerge through iterations—if such accuracy is required at all. Yet, for small numbers of involved preferences, pairwise comparisons are few enough to allow AHP-style assessment without adding too much burden (e.g., prioritizing $n = 4$ preferences requires $n(n - 1)/2 = 6$ pairwise comparisons per cycle). It is up to the users, based on their application circumstances, to decide when improvement in domain knowledge and representation has reached a satisfactory level through this process—in our case, we ended the iterations when the result did not suggest any new updates. The associated burden directly translates into gain in understanding and representation accuracy. We feel that avoiding a strongly iterative approach and using our framework solely as an explanatory companion to AHP or other prioritization technique misses many opportunities that our toolset offers (e.g., thanks to its computational efficiency) and this is why we emphasize iterations in this paper.

Finally, a similar comment applies to the number of different situations/scenarios that need to be analyzed—e.g., in our bookseller example different customers and orders. Our framework neither depends on nor requires exhaustive identification of scenarios in which priorities might be different. Thus, analysis can be detached from any scenarios as the user may wish to not distinguish between cases, e.g., a bookseller that wants to treat all customers the same way. We however feel that relativising priority and preference to situations and contexts is a very useful concept. In that case, we expect that the analysts will use their intuition in order to focus on the most likely, interesting, or otherwise relevant situations.

7.3 Model quality and improvement

Another question to consider is the correctness of the preference specification itself as well as of the goal model and how this is facilitated in our framework. For example how possible is for the analyst to specify an internally conflicting preference or establish e.g., cycles of precedence links in the goal model?

As far as temporal preferences are concerned, we did not have examples of inconsistent temporal preferences in our applications. We believe this is due to the fact that the framework encourages the use of either higher-level goals

or tasks within the templates, preventing users from constructing arbitrary formulae grounded on tasks. This limits the inconsistency possibilities to cases such as precedences between parent goals and their subgoals or between copies of the same goal, which can generally be detected and avoided easily. On the other hand, the types of temporal constraints that were constructed never deviated from the ones displayed on Table 1, which is actually why we were later motivated to construct those templates. While, other than our own experience, we do not have empirical evidence on the comprehensibility of the particular templates, we are convinced that it is a generally sound approach considering that the idea of using LTL patterns has been very influential in the literature (see [24]).

More likely reasons for corrections were cases of problematic arrangement of makes, breaks, and precedence links in the goal model, leading e.g., to unsatisfiable quality preferences or, less often, to goal models with no solution due to cyclic precedence dependencies. Most of the effort was actually dedicated to the fine tuning of the makes and breaks links into meaningful configurations. This process is an important part of any i^* type of modeling. In our work it is exactly the reasoning exercise that helped us to identify and address unintuitive modeling decisions—which we think would otherwise remain undetected. The model of Fig. 1 is an example of a model that has improved several times thanks to automated analysis both in terms of makes and breaks links and (mostly in this model) in terms of precedences. In the nursing domain, the first reaction after the identification of a conflict was questioning the contribution links that caused the conflict in terms of their accuracy and proper justification, before assuming that such a conflict might indeed exist.

Moreover, in connecting the mandatory decomposition with the preferences, we often found ourselves tempted to include both partial contributions (help and hurt in i^*) and a less crisp interpretation of the AND-decompositions of quality preferences. Both these options can be pursued by requiring that quality goals yield weighted combinations rather than crisp formulae over lower-level items (cf. Frame E of Fig. 2). Thus, back in Fig. 1, instead of including *Happy Customer* in the priority specification, we can include each of its subgoals with a different weight. Nevertheless, while this is a sound technique, we found that it may result in longer priority specifications, without being rewarding in terms of intuitiveness.

A final comment concerns the size and complexity of the model. Arguably, the more these aspects grow the less comprehensible the goal model tends to be. Our experience however did not indicate that this can become an issue for the models of up to about 50 elements that we tried—larger models that we tried for measuring performance were artificial. In the ATM example (28 elements), a developer

made successful use of a relatively complex precedence structure of the goal model in order to build a high-variability ATM prototype that satisfies the behavioral variability in the goal model—we report on this work elsewhere [25, 26]. We believe that comprehension of larger models can be facilitated either through exploration through the tool or through fragmenting the goal model into submodels that depict different subtrees. The precedence structure can also be separated from the decomposition structure in a separate view. We found however that this may obstruct comprehensibility instead of supporting it, especially if precedence links connect goal elements of different levels.

8 Performance evaluation

While HTNPlan-P is known to perform remarkably well for fairly large and complex hierarchical task networks [9], we performed additional performance evaluation steps in order to understand the planner’s usability on practical goal and preference modeling tasks.

Firstly, we performed measurements of the reasoning times of the four domains we discussed above: the bookseller (34 mandatory elements in total including 21 tasks plus 13 quality preferences), the nursing domain (24 mandatory elements 11 of which tasks plus 7 quality preferences), an extended meeting scheduling example (53 mandatory elements, 32 tasks, 13 quality preferences) as well as the ATM example (28 mandatory elements, 21 tasks, 4 quality preferences). The results can be seen in Table 6. 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”) it shows the time until the optimal plan is found. 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. To further explore up to what size of models can allow for such performance, we experimented on artificial models of various sizes. Such models were constructed by randomly combining the models from the four domains we developed above—i.e., placing one (or parts thereof) as a subtree of another in different combinations. In this way, the artificial models retain some structural similarities with the real ones (e.g., branching factors, precedence concentration and structure, distribution of ANDs and ORs, etc.).

We constructed sets of such models based on their number of leaf-level tasks. In particular, we created 7 sets, each containing 8 models with 50, 60, 70, 80, 90, 110, and 130 tasks. For each set, 4 of the models were tested with priority specifications containing existential and absence preferences and the remaining 4 were tested with ones featuring precedence and response preferences. We deliberately crafted the priority specifications for each goal model so that there exists a plan in the model that satisfies all the preferences mentioned in each priority. In each test, the planner continues to search and once a plan that better satisfies the priority is found it is given as output. The planner terminates if it finds that there is no plan with better score than the one that was last found—if any.

For each of the 7 sets of models, we measured the average percentage of preferences of the priorities that the planner would satisfy within 10 s, 1, 1.5, 5, 15 min, and 1 h. The result can be seen in Table 7. The first column is the number of leaf-level tasks, the second column is the overall size of the mandatory decompositions, and the rest of the columns are the percentages of preferences satisfied. Thus, for models of 50 tasks, the planner was able to find plans that satisfy, on average, 97.5% of the preferences of the given priority specification within 10 seconds. Within 1.5 min, it found the optimal plan in all cases of this set. As the number of tasks increases, the planner needs more time to

Table 6 Performance results for practical goal models

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

Table 7 Percentage of constraints satisfied over time

# Tasks	# Elements	≤10 s (%)	≤1 min (%)	≤1.5 min (%)	≤5 min (%)	≤15 min (%)	≤1 h (%)
50	81–89	97.50	98.75	100.0	100.0	100.0	100.0
60	98–106	91.25	96.25	96.25	97.5	98.75	98.75
70	112–119	85.00	93.75	93.75	96.25	96.25	96.25
80	127–134	87.50	91.25	91.25	93.75	96.25	96.25
90	144–146	81.25	87.50	90.00	90.00	91.25	91.25
110	180–182	81.25	86.25	86.25	87.50	87.50	87.50
130	216–218	78.75	80.00	81.25	82.50	83.75	83.75

find plans that satisfy more preferences. All cases in which the planner was not able to find the optimal plan within 1 h were because the planner ran out of memory. Note also that there was no significant difference between existential and absence versus precedence and response preferences.

The results suggest that the possibility of online analysis of preferences (e.g., in an interactive tool) is open even for models with several tens of leaf-level tasks. The results further constitute dramatic improvement from our earlier experimentation with non-hierarchical preference-based planners [27]. It must be stressed, however, that classical planning remains a hard problem [28] and, as such, the risk of exponential blow-up is present, particularly when the modeler attempts the addition of constraints that connect e.g., low-level nodes across distant sides of the goal tree, effectively creating interactions between subgoals, which are known to worsen the performance of planners. A similar effect may appear when the modeler drastically reduces the number of constraints, which results to a very large search space. Clearly, the sizes and structures of models that are meaningful for preference analysis need to be identified through developing and working with more and realistically large real-world examples. Such empirical evaluation is a priority in our future research agenda.

9 Related work

The need to include stakeholder attitudes and preferences as well as optionality in requirements modeling has been demonstrated by Jureta et al. [29] 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 among stakeholders. Approaches for eliciting these degrees of importance range from simple empirical division of requirements into “must-have” and “nice-to-have” [30, 31] to more elaborate quantitative prioritization techniques, such as the Analytic Hierarchy Process (AHP) we discussed earlier [8], multi-criteria analysis methods [32], or more advanced techniques based on machine learning [33].

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., [34]) 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.

Researchers have also proposed different ways to model and bind variability in business processes. Lapouchnian et al. use goal models for analyzing alternative business

process configurations [36]. Lu et al. propose the construction of flexible business process templates that lay the basic constraints that must be met [37]. Elsewhere [38, 39] variability constructs are added to existing business process notations. The role of goals in business process design and analysis has also been investigated. Soffer and Wand, for example, offer a treatment of soft goals in business processes [40]. In particular, the state of the domain is explicitly modeled, and soft goals are then defined as preference rankings over such states. The work has been extended to allow validation of multi-process settings [41]. Our work differs in a number of ways including the fact that our primary reasoning proposal does not rely on explicitly modeling state (though this is certainly possible as well as we saw) and that our framework is strongly focussed toward efficient automated reasoning (with or without state modeling) about behavioral designs that support a given preference specification. Furthermore, Yi and Johannesson suggest analyzing high-level soft goals into concrete and verifiable business rules [42]. Very similarly in [43] patterns are introduced to ensure that the results of soft-goal analysis are reduced into means and criteria that are sensible from a business process point of view. Techne [18], which we discuss below, makes a similar case for coming up with (potentially ranked) approximations of soft goals. Although our approach for specifying priorities does not preclude deeper analysis of soft goals at the level of concrete criteria and business rules, it is also suitable for preference specification when goals are not yet analyzed to testable criteria. This allows for useful analysis to take place earlier in the requirements process.

Elsewhere, deontic logic has been proposed for characterizing actions in terms of their performance necessity under given conditions [44]. The idea has been applied to also analyze strategic dependency structures [45]. The use of deontic logic to cope with behavioral variability for software product lines is also being explored [46]. While deontic logic offers a semantically rich way to stratify actions into obligatory, recommended, permissible, etc., traditional deontic characterizations may not be comparable to notions of preference—e.g., “obligatory” is not necessarily “preferred”. Our approach is also different in that it does not require exhaustive definition of such characterizations and rules thereof and, again, focuses on offering a very efficient way for automated reasoning.

It also needs to be emphasized that the addition of a temporal dimension to goal models in our proposal is not meant to substitute business process modeling or other kinds of behavioral modeling. It is simply an effort to represent an additional aspect in goal modeling: that of constraining temporal ordering of goal satisfaction.

Nothing prevents the result from being refined into specialized frameworks from business process modeling as, for example, suggested in [36]. Furthermore, the plans that emerge from ordering leaf-level tasks should not necessarily be understood as processes in a business context but can model other forms of behavior: Our ATM example, for instance, describes lower-level interactions between the user and the system, offering to the goal model a stronger task analysis [47] flavor—also attempted elsewhere (e.g., [6, 48]).

In the requirements engineering literature, a constraint language for selecting scenario instances from generic use cases has been proposed by Sutcliffe et al. [49]. In the goal modeling community, there has been work for analyzing the temporal dimension of goal satisfaction [16, 17, 50] often through the use of a planner—e.g., Bryl et al. use PDDL for exploring minimum cost refinement and delegation [51]. Meanwhile, works such as that by Sebastiani et al. [52] propose computation of (static) requirements alternatives through reasoning about partial satisfaction of quality goals. Many such evaluation techniques have been proposed as surveyed by Horkoff and Yu [53]. For example, preference matrices have been used for measuring various qualities of a requirements specification [54]. However, these approaches do not focus on how preference and priority (versus hard constraints) can be modeled, and they do not put emphasis to both high-level quality goals of stakeholders and temporal characteristics of goal fulfillment, 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* [18], founded on the CORE ontology [29]. *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 temporal constraints, nor is it offering specific algorithms for finding solutions to requirement 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 preference goals as well as optional subgoals in the mandatory decomposition triggers a question about the relationship between goal models and feature models [15, 55] in terms of their expressiveness and semantics. There have been several efforts to relate goal

models to feature models [56]. António et al. [57] introduce a number of heuristics for doing so in the form of hypotheses, while Uno et al. [58] propose a feature model derivation process from goal models, through goal model merging and subsequent feature extraction and variability analysis. Yu et al. [59] go on to offer a concrete algorithm for feature model generation from goal models. Moreover, both Silva et al. and Mussbacher et al. suggest a goal-to-feature-model connection via aspect orientation ([60] and [61], respectively). In the former work, a correspondence is established between feature models and an aspect-oriented intentional model, appropriately extended to cater to product lines. The latter effort attempts direct relationship between GRL tasks and features.

There is a fundamental difference between preferences and optional requirements. Generally speaking, a set of consistent preferences P is not a solution in our preference semantics unless the set is maximal, i.e., there is no other preference p such that $p \wedge P$ is consistent. In our work, this maximality criterion is applied through the scoring function. On the other hand, if you have a set of consistent optional requirements O , any subset of those constitutes a solution. In this respect, the notion of optional requirement alone seems insufficient for our purposes, in that if you have one, you can always choose to ignore it. Moreover, our preference specification framework does not necessarily relate to product lines and product variability. It is instead meant to be a tool for exploring the space of alternative behaviors that users can adopt in order to fulfill their goals. It may or may not be the choice of the developers to implement all this behavioral variability in a product line. Even if one is interested in implementing goal variability of the type we discuss in this paper, we hypothesize that feature models may not be the first choice for effectively describing variability in behavior, that is orderings of system functions. Some discussion on this subject based on empirical observations can also be found in [26] and [25]. A closer investigation on the relationship between goal variability and system variability also in the context of the work we survey above is something that we continue to study.

10 Conclusions

We present a goal-based framework for modeling and reasoning about preferred and prioritized requirements. Our main contributions include an approach for modeling preferences in traditional goal models, as well as techniques for reasoning about mandatory and prioritized preferred requirements. We begin by introducing in a traditional goal modeling notation the distinction between mandatory and preferred goals. The former define a

requirements problem, while the latter express desired qualities or temporal ordering constraints. A priority specification defines priorities among preference goals, through the assignment of numerical weights, analogously to popular quantitative priority elicitation techniques. A state-of-the-art planner is then exploited to identify alternative solutions of mandatory goals that best satisfy preferred goals and priorities among them. The formalization can further be extended with instance-level details to allow for more advanced reasoning. In terms of performance, we show that the planning tool performs very well in several realistic requirements models that we experimented with.

Our experiments indicate that the task of reasoning about preferences and alternatives allows better understanding of the connection between the stakeholder attitudes and alternative designs. This makes it particularly useful for exploring alternative designs during early requirements stages, supporting priority elicitation activities by directly showing the implication of certain prioritizations, improving domain understanding and model accuracy, or, potentially, supporting the customization of software systems by connecting the high-level design descriptions obtained through the tool into configurations of variation points in the software itself [14, 25].

Our proposal can be seen as an extension of goal modeling techniques or alternatively as an extension of requirements prioritization techniques. The practicality of our proposal rests on the popularity—and some uptake in industrial practice—of both goals and prioritizations in RE. However, more formal empirical evaluations of our proposal will be needed to confirm this claim.

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 efficiency and meaningfulness of the reasoning process. Secondly, our empirical explorations must include acquisition of a better understanding of how the combination of high-level qualities and temporal constraints in preferences and priorities influences the priority elicitation process. Finally, by looking closely at current variability representation techniques [15, 55, 35], we plan to propose enhancements to our language that would make it an even stronger variability representation tool.

References

- van Lamsweerde A (2001) Goal-oriented requirements engineering: a guided tour. In: Proceedings of the fifth IEEE international symposium on requirements engineering, RE '01. IEEE Computer Society, Washington, DC
- Dardenne A, van Lamsweerde A, Fickas S (1993) Goal-directed requirements acquisition. *Sci Comput Program* 20(1–2):3–50
- Giorgini P, Mylopoulos J, Nicchiarelli E, Sebastiani R (2002) Reasoning with goal models. In: Proceedings of the 21st international conference on conceptual modeling (ER'02). London, UK, pp 167–181
- Yu ESK (1997) Towards modelling and reasoning support for early-phase requirements engineering. In: Proceedings of the 3rd IEEE international symposium on requirements engineering (RE'97). Washington, DC
- Liaskos S, Lapouchnian A, Yu Y, Yu E, Mylopoulos J (2006) On goal-based variability acquisition and analysis. In: Proceedings of the 14th IEEE international requirements engineering conference (RE'06). IEEE Computer Society, Minneapolis, pp 79–88
- Hui B, Liaskos S, Mylopoulos J (2003) Requirements analysis for customizable software: a goals-skills-preferences framework. In: Proceedings of the 11th IEEE international requirements engineering conference (RE'03). Monterey Bay, pp 117–126
- Mylopoulos J, Chung L, Liao S, Wang H, Yu E (2001) Exploring alternatives during requirements analysis. *IEEE Softw* 18(1): 92–96
- Karlsson J, Ryan K (1997) A cost-value approach for prioritizing requirements. *IEEE Softw* 14(5):67–74
- Sohrabi S, Baier JA, McIlraith S (2009) HTN planning with preferences. In: Proceedings of the 21st international joint conference on artificial intelligence (IJCAI'09). Pasadena, pp 1790–1797
- Gerevini A, Long D (2005) Plan constraints and preferences in PDDL3. Technical report. Department of Electronics for Automation, University of Brescia
- Nau D, Cao Y, Lotem A, noz Avila HM (1999) SHOP: simple hierarchical ordered planner. In: Proceedings of the 16th international joint conference on artificial intelligence (IJCAI-99), pp 968–973
- Liaskos S, McIlraith SA, Sohrabi S, Mylopoulos J (2010) Integrating preferences into goal models for requirements engineering. In: Proceedings of the 10th international requirements engineering conference (RE'10). Sydney, Australia, pp 135–144
- Yu ESK, Mylopoulos J (1994) Understanding “why” in software process modelling, analysis, and design. In: Proceedings of the 16th international conference on software engineering (ICSE'94), pp 159–168
- Liaskos S, Lapouchnian A, Wang Y, Yu Y, Easterbrook S (2005) Configuring common personal software: a requirements-driven approach. In: Proceedings of the 13th IEEE international requirements engineering conference (RE'05). Paris, France, pp 9–18
- Czarniecki K, Eisenecker UW (2000) Generative programming—methods, tools, and applications. Addison-Wesley, Reading
- Fuxman A, Liu L, Mylopoulos J, Pistore M, Roveri M, Traverso P (2004) Specifying and analyzing early requirements in Tropos. *Requir Eng* 9(2):132–150
- Wang X, Lesperance Y (2001) Agent-oriented requirements engineering using ConGolog and i*. In: AOIS-2001 Bi-conference workshop at agents 2001 and CAiSE'01
- Jureta JJ, Borgida A, Ernst NA, Mylopoulos J (2010) 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, pp 115–124
- Saaty RW (1987) The analytic hierarchy process—what it is and how it is used. *Math Model* 9(3-5):161–176
- Vaidya OS, Kumar S (2006) Analytic hierarchy process: an overview of applications. *Eur J Oper Res* 169(1):1–29
- Wieggers K (1999) First things first: prioritizing requirements. *Softw Develop* 7(9)

22. Azar J, Smith RK, Cordes D (2007) Value-oriented requirements prioritization in a small development organization. *IEEE Softw* 24:32–37
23. Hansson SO (2005) Decision theory: a brief introduction. Technical report, Department of Philosophy and the History of Technology, Royal Institute of Technology (KTH)
24. Dwyer MB, Avrunin GS, Corbett JC (1999) Patterns in property specifications for finite-state verification. In: Proceedings of the 21st international conference on software engineering (ICSE '99). IEEE Computer Society Press, Los Alamitos, pp 411–420
25. Liaskos S, Litoiu M, Jungblut MD, Mylopoulos J (2011) Goal-based behavioral customization of information systems. In: Proceedings of the 23rd international conference on advanced information systems engineering (CAiSE'11). London, UK
26. Liaskos S, Jungblut MD, Mylopoulos J (2010) From goal models to three-layer web-based systems: an exploratory study. In: Proceedings of the 1st RE'10 workshop on the web and requirements engineering (WeRE'10). Sydney, Australia
27. Liaskos S, McIlraith SA, Mylopoulos J (2009) Towards augmenting requirements models with preferences. In: Proceedings of the 24th IEEE/ACM international conference on automated software engineering (ASE'09), pp 565–569
28. Erol K, Hendler J, Nau D (1996) Complexity results for HTN planning. *Ann Math Artif Intel* 18:69–93
29. Jureta I, Mylopoulos J, Faulkner S (2008) Revisiting the core ontology and problem in requirements engineering. In: Proceedings of the 16th IEEE international requirements engineering conference (RE'08), pp 71–80
30. Beck K (1999) Extreme programming explained. Addison Wesley, Reading
31. Clegg D, Barker R (1994) Case method fast-track: a RAD approach. Addison Wesley, Reading
32. In HP, Olson D, Rodgers T (2002) Multi-criteria preference analysis for systematic requirements negotiation. In: Proceedings of the 26th annual international computer software and applications conference (COMPSAC'02), pp 887–892
33. Avesani P, Bazzanella C, Perini A, Susi A, Facing scalability issues in requirements prioritization with machine learning techniques. In: Proceedings of the 13th IEEE international requirements engineering conference (RE'05), pp 297–305
34. Zhang H, Jarzabek S, Yang B (2003) Quality prediction and assessment for product lines. In: Proceedings of the 15th international conference on advanced information systems engineering (CAiSE'03), pp 681–695
35. Halmans G, Pohl K (2003) Communicating the variability of a software-product family to customers. *Softw Syst Model* 2(1):15–36
36. Lapouchnian A, Yu Y, Mylopoulos J (2007) Requirements-driven design and configuration management of business processes. In: Proceedings of the 5th international conference on business process management (BPM 2007), pp 246–261
37. Lu R, Sadiq S, Governatori G (2009) On managing business processes variants. *Data Knowl Eng* 68(7):642–664
38. Gottschalk F, van der Aalst WM, Jansen-Vullers MH, La Rosa M (2008) Configurable workflow models. *Int J Cooperat Inf Syst*
39. Sadiq SW, Orlowska ME, Sadiq W (2005) Specification and validation of process constraints for flexible workflows. *Inf Syst* 30(5):349–378
40. Soffer P, Wand Y (2005) On the notion of soft-goals in business process modeling. *Bus Process Manage J* 11:663–679
41. Soffer P, Wand Y (2007) Goal-driven multi-process analysis. *J Assoc Inf Syst* 8(3):175–203
42. Yi Ch, Johannesson P (1999) Beyond goal representation: checking goal-satisfaction by temporal reasoning with business processes. In: Advanced information systems engineering, lecture notes in computer science, vol 1626, pp 462–466
43. Andersson B, Johannesson P, Zdravkovic J (2009) Aligning goals and services through goal and business modelling. *Inf Syst e-Bus Manage* 7:143–169
44. Krogstie J, Sindre G (1996) Utilizing deontic operators in information systems specification. *Requir Eng* 1:210–237
45. Padmanabhan V, Governatori G, Sadiq S, Colomb R, Rotolo A (2006) Process modelling: the deontic way. In: Proceedings of the 3rd Asia-Pacific conference on conceptual modelling (APCCM '06). Darlinghurst, Australia, pp 75–84
46. Fantechi A, Asirelli P, ter Beek M, Gnesi S (2009) Deontic logics for modeling behavioural variability. In: Proceedings of the 3rd international workshop on variability modelling of software-intensive systems (VaMoS'09). Sevilla, Spain
47. Diaper D, Stanton NA (2004) The handbook of task analysis for human-computer interaction. Lawrence Erlbaum Associates, London
48. Wang Y, McIlraith SA, Yu Y, Mylopoulos J (2007) An automated approach to monitoring and diagnosing requirements. In: Proceedings of the 22nd IEEE/ACM international conference on automated software engineering (ASE '07), pp 293–302
49. Sutcliffe AG, Maiden NAM, Minocha S, Manuel D (1998) Supporting scenario-based requirements engineering. *IEEE Trans Softw Eng* 24(12):1072–1088
50. Gans G, Jarke M, Lakemeyer G, Vits T (2002) Snet: A modeling and simulation environment for agent networks based on i* and ConGolog. In: Proceedings of the 14th international conference on advanced information systems engineering (CAiSE'02). Toronto, Canada
51. Bryl V, Giorgini P, Mylopoulos J (2006) Designing cooperative IS: Exploring and evaluating alternatives. In: Proceedings of the 14th international conference on cooperative information systems (CoopIS'06). Springer, Montpellier, pp 533–550
52. Sebastiani R, Giorgini P, Mylopoulos J (2004) Simple and minimum-cost satisfiability for goal models. In: Proceedings of the 16th conference on advanced information systems engineering (CAiSE'04), pp 20–35
53. Horkoff J, Yu E (2011) Analyzing goal models—different approaches and how to choose among them. In: Proceedings of the 26th symposium on applied computing (SAC'11). Taiwan
54. Kaiya H, Horai H, Saeki M (2002) AGORA: attributed goal-oriented requirements analysis method. In: Proceedings of the 10th anniversary IEEE joint international requirements engineering conference (RE'02), pp 13–22
55. Schobbens PY, Heymans P, Trigaux JC (2006) Feature diagrams: A survey and a formal semantics. In: Proceedings of the 14th IEEE international requirements engineering conference (RE'06), IEEE Computer Society, Los Alamitos, pp 139–148
56. Borba C, Silva C (2009) A comparison of goal-oriented approaches to model software product lines variability. In: Proceedings of the ER 2009 workshops on advances in conceptual modeling—challenging perspectives, pp 244–253
57. António S, Araújo Ja, Silva C (2009) Adapting the i* framework for software product lines. In: Proceedings of the ER 2009 workshop on requirements, intentions and goals in conceptual modeling (RIGIM'09), pp 286–295
58. Uno K, Hayashi S, Saeki M (2009) Constructing feature models using goal-oriented analysis. In: Proceedings of the 9th international conference on quality software (QSIC '09), pp 412–417
59. Yu Y, do Prado Leite JCS, Lapouchnian A, Mylopoulos J (2008) Configuring features with stakeholder goals. In: Proceedings of the 2008 ACM symposium on applied computing (SAC '08), pp 645–649
60. Silva L, Batista T, Soares S, Santos L (2010) On the role of features and goals models in the aspect-oriented development of software product line. *Information Sciences and Technologies Bulletin of the ACM Slovakia, Special Section on Early Aspects 2*

61. Mussbacher G, Amyot D, Araújo Ja, Moreira A (2008) Modeling software product lines with AoURN. In: Proceedings of the 2008 AOSD workshop on early aspects (EA'08), EA '08, vol 2, pp 1–2 (8)
62. van Lamsweerde A (2000) Requirements engineering in the year 00: a research perspective. In: Proceedings of the 22nd international conference on software engineering (ICSE'00). ACM, New York, pp 5–19