2009 IEEE/ACM International Conference on Automated Software Engineering

Towards Augmenting Requirements Models with Preferences

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

Sheila A. McIlraith
Dept. of Computer Science
University of Toronto
Toronto, ON, Canada
sheila@cs.utoronto.ca

John Mylopoulos
Dept. of Computer Science
University of Toronto
Toronto, ON, Canada
jm@cs.utoronto.ca

***Abstract*—The analysis of stakeholder requirements is a critical aspect of software engineering. A common way of specifying stakeholder requirements is in terms of a hierarchy of goals whose AND/OR decomposition captures a family of software solutions that comply with the goals. In this paper, we extend this goal modeling framework to include the specification of optional user requirements and user preferences, aggregated together into weighted formulae to be optimized. We team this with an automated reasoning tool, adapted from state of the art research in artificial intelligence planning with preferences, in order to synthesize solutions that both comply with the goals and optimize stakeholder preferences and optional requirements.**

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

I. Introduction

Goal-oriented requirements engineering techniques specify requirements as stakeholder goals to be refined in terms of AND/OR decompositions in order to derive alternative sets of actions/tasks ([1]). The resulting requirements models demand that these tasks *must* necessarily be performed if the goals are to be attained.

However, in software engineering practice requirements need not always take the form of goals that must be strictly enforced. Rather, requirements can also include optional properties that are "nice to have" but are not mandatory. Such properties can range from low-level behavioral details of the system under design to high-level quality requirements. Further, they are used to distinguish alternative designs. Given two designs that satisfy the mandatory requirements, one design will be preferred to another if it better satisfies the optional requirements. However, optional requirements can often be conflicting and, as such, they need to be further refined through expressing *preferences* over their satisfaction. Moreover, even when such preferences are clearly expressed, the process of transforming them into the appropriate configurations of design decisions can be too complex to be performed without tool support.

In this paper, we explore how preferences can be represented and used to automatically identify designs that best satisfy them. Representation of the domain of possible designs is done through the development of goal decomposition models. Preferences are specified as rankings over desired design properties written in Linear Temporal Logic (LTL). To synthesize designs that adhere to goal requirements while optimizing preferences over optional requirements, a heuristic search preference-based planner is adapted ([2]). By using this tool, analysts can explore the design space and identify combinations of design decisions that best match high-level stakeholder priorities.

The paper is organized as follows. In Section II we present our running example and survey the related literature. In Sections III and IV we present our goal modeling and preference specification languages, respectively. We describe the reasoning tool in Section V and conclude in Section VI.

II. Motivation and Background

To see how requirements variability and preferences emerge during requirements analysis we will consider an example from a health care domain, namely a geriatric assessment unit. In that unit, elderly patients with a variety of health issues are hospitalized for a period of time. We are interested in understanding alternative ways by which nurses can be notified about and respond to different requests that come from the patient's room, either by the patients themselves or by a variety of sensors that have been installed around their bed. The first concern is how the nurse will be notified: this can happen either through a broadcasted notification using the speakers of the unit, or, alternatively, through earphones that the nurse wears while on duty. The broadcasted notification may be disturbing for the workers and the patients, while, on the other hand, the nurses may not feel comfortable wearing a mobile device. Thus, there is a question of preference between disturbance and nurse comfort, which is interpreted into different design decisions. Furthermore, once the notification is sent, the nurse's reaction needs to be determined. Normally, he has to visit the patient's room, but if the patient only wants to ask a question or request permission for something, the visit may be replaced by establishing a voice link between patient and nurse. Again there are several possibilities for enabling this. For example, the nurse may again be carrying a mobile set with microphone and earphones, or there may be a communication device at the nursing station, which is conveniently located in the unit. The nurses think that this would increase unnecessary disturbance from some patients, but they acknowledge it would also increase their productivity and save them from extra walking effort. Again, knowing how much the nurses value productivity versus not being distracted – both of which are high-level quality requirements – can help us select the best solution for them.

In the literature, the need for a view of requirements that explicitly takes such stakeholder attitudes, preferences as well as optionality into account has recently been illustrated by Jureta et al. in [3]. The traditional notion of requirements prioritization originates exactly from the observation that not all requirements have the same importance for all stakeholders. An elementary requirements prioritization approach, for example, is to divide requirements into "must-have" and "nice-to-have", whereby the former are understood as more important, urgent or otherwise of higher priority. In addition to this common qualitative approach, more elaborate quantitative prioritization techniques, such as the Analytic Hierarchy Process ([4]) or multi-criteria analysis methods ([5]) have also been proposed and successfully used in practice.

The modeling and reasoning side of prioritization, however, has not received as much attention in software engineering. The limited number of efforts that attempt to reason about the prioritization of potentially inconsistent requirements focus, for the most part, on identifying combinations of coarse-grained features of the system under design rather than high-level quality goals or behavioral designs that result from such goals (e.g. [6], [7]). In the requirements engineering literature, a constraint language for selecting scenario instances from generic use-cases has been proposed in [8], while in [9], requirements alternatives are computed through reasoning about partial satisfaction of quality goals. However, neither of these approaches combine preference (versus constraint) specification with reasoning about both partial satisfaction of goals and temporal characteristics of solutions. Our goal-based approach, which we describe in the next sections, suggests that this combination is possible.

III. Goal Models

Goal models ([1], [10], [11]) have been found to be effective in concisely capturing large numbers of alternative sets of low-level tasks, operations, and configurations that can fulfill stakeholder goals. The goal modeling language we use adopts the basics of existing goal modeling notations (particularly $i*$ - [10]) and is extended in order to accommodate quantitative analysis of goal satisfaction (adopting [9]), while allowing temporal constraints as well as variables for describing the environment.

In Figure 1 a goal model representing possible alternatives for the nursing example of Section II is shown. The model consists of *hard goals*, *soft goals* and *tasks*, as well as *condition elements* and *effect elements* which comprise *satisfaction predicates* and *domain predicates*. Hard goals (represented using oval-shaped elements) are states of affairs or conditions that one or more actors of interest would like to achieve ([10]). Hard goals are goals for which there is a well-defined criterion for determining goal satisfaction. In contrast, soft goals, depicted as cloud-shaped elements in Figure 1, do not have a clearly-defined criterion for determining goal satisfaction. A soft goal is determined to be satisifed to a *certain degree* by subjective judgement and the existence of relevant evidence. Soft goals are used to represent high-level quality goals to be fulfilled with the support of the system under design. Thus, *Have Nurse Notified* is an example of a hard goal, while *Happy Patient* is a typical soft goal. Tasks, on the other hand, the hexagonal shapes in Figure 1, describe particular activities that the actors perform in order to fulfill their goals, e.g. *Send Audio Notification*. For ease of future reference, we have annotated each task in Figure 1 with a literal t_i.

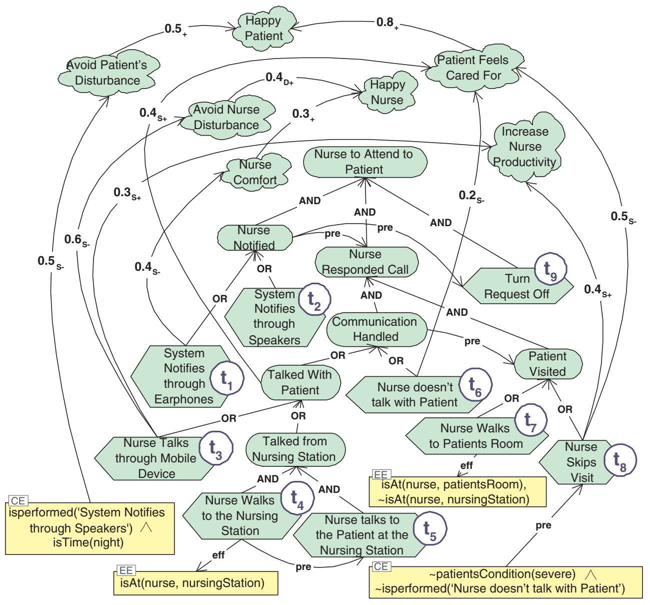
Happy Patient
Avoid Patient's Disturbance
Patient Feels Cared For
Avoid Nurse Disturbance
Happy Nurse
Increase Nurse Productivity
Nurse Comfort
Nurse to Attend to Patient
Nurse Notified
Nurse Responded Call
Turn Request Off
System Notifies through Speakers
Communication Handled
Patient Visited
System Notifies through Earphones
Talked With Patient
Nurse doesn't talk with Patient
Nurse Walks to Patients Room
Nurse Talks through Mobile Device
Talked from Nursing Station
Nurse Skips Visit
Nurse Walks to the Nursing Station
Nurse talks to the Patient at the Nursing Station
t1
t2
t3
t4
t5
t6
t7
t8
t9
0.5+
0.8+
0.4D+
0.4S+
0.3+
0.3S+
0.2S-
0.6S-
0.5S-
0.4S-
0.5S-
0.4S+
AND
OR
pre
eff
CE
isperformed('System Notifies through Speakers') ∧ isTime(night)
EE
isAt(nurse, nursingStation)
EE
isAt(nurse, patientsRoom), ~isAt(nurse, nursingStation)
CE
~patientsCondition(severe) ∧ ~isperformed('Nurse doesn't talk with Patient')

Figure 1. A goal model

Hard goals and tasks are related to each other through AND or OR-decomposition links. When a hard goal is AND-decomposed into other hard goals or tasks, we say that it is satisfied if each one of its children is also satisfied (performed in the case of tasks). When a hard goal is OR-decomposed, then it is satisfied if there exists a sub-goal (resp. sub-task) that is satisfied (resp. performed). Soft goals, on the other hand, are connected to each other and to hard goals and tasks through directed *weighted contribution links*. Such links show how the level of satisfaction and denial of a source can influence our knowledge of the satisfaction or denial of its destination. To model satisfaction levels, each soft goal l is associated with two variables $valS(l)$

and $valD(l)$, each representing the degree to which we believe the goal to be satisfied or denied. The existence and weight of the contribution links dictate how the degree of satisfaction and denial of one soft goal is propagated to other soft goals, through previously defined propagation rules detailed in [9].

Furthermore, we use the satisfaction predicates $isSat(g)$ and $isPerformed(t)$ to denote that a hard goal g or a task t has been satisfied or performed, respectively. Domain predicates, such as $isAt(nurse, nursingStation)$ express ways by which domain concepts relate to each other at a particular time instance and while tasks are being performed. The truth status of domain predicates may change due to performance of tasks. Using satisfaction and domain predicates we can construct *Condition Formulae (CF)*, which are first-order formulae describing what is true in the domain at a particular point in time. For example, $isSat(nurseNotified) \wedge isAt(nurse, nursingStation)$ is true at a given point in time if the goal $nurseNotified$ has been satisfied and the nurse is at the nursing station at that point. An *effect*, on the other hand, is a formula consisting of a single domain predicate, potentially negated. Both CFs and lists of effects are placed in rectangle-shaped elements in Figure 1 – the Condition Elements (CE) and the Effect Elements (EE), respectively – and are connected to tasks using *precedence* and *effect* links, respectively. The former imply that the CF needs to be true before the task can be performed, while the latter shows which domain predicates instantaneously become true or false due to the performance of tasks. Precedence links can be drawn between hard-goals and tasks too, meaning that none of the tasks in the subtree of the target can be performed unless the source is satisfied/performed. Also, CEs may connect to soft goals with contribution links, meaning that if and for as long as the contained CF is true, the satisfaction of the soft goal is affected according to the type and weight of the link.

A *plan* is a sequence of leaf level tasks that collectively satisfy the root goal of the AND/OR tree, while the sequence per se satisfies the precedence and effect links of the goal graph. In our example of Figure 1, given initial conditions in which $patientsCondition(severe)$, $isTime(afternoon)$ holds, the sequence $[t_1, t_3, t_7, t_9]$ is a plan for the root goal, while the sequences $[t_1, t_3, t_9]$ and $[t_3, t_1, t_7, t_9]$ are not. Each plan also determines the degree of satisfaction or denial of soft goals, following satisfaction propagation and aggregation rules - cf. [9].

IV. The Preference Language

Goal models represent a great number of alternative designs (plans) for fulfilling stakeholder goals. However the perceived quality of these alternative designs varies based on criteria imposed by individual stakeholders and the context of a particular instance. In our health care example, different geriatric assessment units, different stakeholders in the same unit or even the same stakeholders in different times and situations, may have different priorities over high-level characteristics of the desired solution, such as for example its implications with respect to nurse productivity or comfort that we saw in Section II. Therefore, a different solution is appropriate in each case. Our preference specification language allows expression of and reasoning about such priorities via the construction of priority rankings over or linear combinations of desired properties of preferred plans. Such desired properties are formulated as *Optional Condition Formulae (OCFs)*.

OCFs describe properties of plans implied by the goal tree. OCFs are not mandatory, in the sense that they are not used to constrain the family of plans implied by the goal model. Instead, OCFs are the building blocks of preference formulae, which in turn are used as criteria for evaluating the desirability of plans. To form OCFs we use satisfaction predicates such as $isSat(nurseNotified)$ and $isPerformed(turnRequestOff)$, domain predicates such as $isTime(night)$, as well as subformulae that either compare pairs of soft-goal satisfaction values, such as $valS(happyPatient) \geq valS(happyNurse)$ or compare soft-goal satisfaction values with an absolute value, such as $valD(nurseComfort) \leq 0.3$. Thus, the last formula means that the denial value of the soft goal $nurseComfort$ should (possibly) be less than 0.3. Furthermore, the symbols $\square$,$\diamond$, $\circ$ and U are used to represent the temporal operators *always, eventually, next* and *until*, respectively. In addition to these standard LTL operators, $final(\cdot)$ holds if the operand holds after the performance of the last task.

Returning to our example, assume that the managers of the geriatric assessment unit state that *"we should definitely avoid anything that would make the patient unhappy"*. In our goal language this means that at all times (i.e. *always*, $\square$), the denial value of the soft goal $happyPatient$, (i.e. $valD(happyPatient)$) must remain below a small value (say 0.1). Thus, we would write the OCF as follows:

$$\square(valD(happyPatient) \leq 0.1) \tag{1}$$

OCFs are also used to impose optional temporal constraints – the counterparts of the mandatory constraints realized in the goal graph through precedence links. For example *"the nurse should turn the request off only after she has responded to the patient's call"* can be formulated as follows:

$$\neg isPerformed(turnRequestOff) \\ U\ isSat(nurseRespondedCall) \tag{2}$$

Given an OCF and a plan for the root goal, the plan will either satisfy or not satisfy the OCF. For example, OCF 2 above is satisfied by plan $[t_1, t_3, t_7, t_9]$ but not by plan $[t_1, t_3, t_9, t_7]$.

OCFs can be combined into preference formulae, which are in turn used as evaluation criteria for plans. In this work, we propose two types preference formulae: (plain) *Preference Formulae (PF)* and *Weighted Preference Formulae (WPF)*.

A PF is of the form $\phi_0[v_0] \succeq \phi_1[v_1] \succeq, \ldots, \succeq \phi_n[v_n]$, where $\succeq$ is a binary preference relation, $n \geq 0$, each ϕ_i is an OCF, and v_i is a value between 0 and 1 characterizing the strength of preference. 0 is most preferred, 1 is least preferred, and for every $i < j$, $v_i < v_j$. When n=0, preference formulae correspond to single OCFs. Using PFs, analysts can define priorities over the satisfaction of properties in the resultant system. In our example, the nurse's statement that they *"don't like the idea of talking to the patient remotely, but if they had to, they would at least choose to do so from the nursing station"* can be formulated with PF:

$$\Box(\neg isSat(talkedWithPatient))[0.0] \\ \succeq \ \Diamond isSat(talkedFromNursingStn)[0.5]$$

This states that the first component OCF, $\Box(\neg isSat(talkedWithPatient))$, is preferred to the second, $\Diamond isSat(talkedFromNursingStn)$. Given a plan, a PF evaluates to a number between 0 and 1 based on which constituent OCFs are satisfied by the plan. In the above example, if the first OCF is satisfied, then the PF evaluates to 0.0. Otherwise, if the second OCF is satisfied, then the PF evaluates to 0.5. If neither OCF is satisfied, the PF evaluates to 1.0 – the worst possible score. Plans $[t_1, t_6, t_7, t_9]$, $[t_1, t_4, t_5, t_7, t_9]$ and $[t_1, t_3, t_7, t_9]$ evaluate the above PF to 0.0, 0.5 and 1.0 respectively.

A PF captures the notion that we are indifferent to the satisfaction of a given ϕ_j, given the satisfaction of ϕ_i, $i < j$, which is the case in the example. We may use *weighted preference formulae* (WPFs), to aggregate preferences over sets of properties. A WPF is a weighted linear combination of OCFs of the form $\Sigma_i(w_i \times \psi_i)$, where $0 \leq w_i \leq 1$, $\Sigma_i(w_i) = 1$, and ψ_i is a PF. Recall that PFs may consist of a single OCF. Returning to our nursing example, assume that the management provides a combination of desires: *"we should definitely avoid anything that would make the patient unhappy, but it would also be nice to increase nurses' productivity somehow."* The statement prioritizes the patient's happiness over the productivity of the nurses, but indicates that both are desirable. This could be represented by the following WPF:

$$\{\Box(valD(happyPatient) \leq 0.1)[0.0]\} \times 0.8 \\ + \{final(valS(incrNurseProd)) \geq 0.1)[0.0]\} \times 0.2$$

This WPF evaluates to 0.0 if both of its constituent single-OCF PFs are satisfied, to 0.2 if only the OCF of the first PF is satisfied, to 0.8 if only the OCF of the second PF is satisfied, and to 1.0 if the OCF of neither PF is satisfied.

$$\boxed{\begin{array}{l} \{\Box(valD(happyPatient) \leq 0.1)[0.0]\} \times 0.72 \\ \ + \ \{final(valS(incrNurseProd)) \geq 0.1)[0.0]\} \times 0.18 \\ \ + \ \{\Box(\neg isSat(talkedWithPatient))[0.0] \succeq \\ \quad \Diamond isSat(talkedFromNursingStn)[0.5]\} \times 0.1 \end{array}}$$

Figure 2. Weighted Preference Formula

WPFs are particularly useful in combining PFs and WPFs of different stakeholders, where the weight associated with each PF can be used to reflect the analyst's sense of the relative importance of individual stakeholders' desires. Figure 2 displays the WPF resulting from giving management a weight of 0.9 (0.72 + 0.18) and nurses a weight of 0.1.

V. Reasoning About Preferences

As we saw, each plan implied by the goal decomposition model satisfies the given user preferences to some degree. Plans that optimize the preferences, minimizing the value of the preference formula, are the most desirable. To support the process of identifying such plans, we extended an optimal preference-based planner called PPLan ([2]). In our extension of PPlan, the planner takes as input the initial conditions, the goal model appropriately translated into a PPlan-readable form, the preference formulae, and a number N, reflecting the number of plans to be returned. The planner returns the N plans of the goal model that best satisfy the given preferences.

In our example of Figure 1, assuming that we are given initial values for the domain predicates $\{\ isTime(afternoon),$ $patientsCondition(moderate)\ \}$ and the preference of Figure 2, the resulting ranking is seen in Figure 3.

Rank	**Plan**	**Score**
1.	$[t_1, t_3, t_7, t_9]$	.1
2.	$[t_2, t_3, t_7, t_9]$	.1
3.-6.	$[\ldots, t_3, \ldots, t_7, \ldots]$	.1
7.	$[t_1, t_4, t_5, t_7, t_9]$	.23
8.-14.	...	.23
15.	$[t_1, t_4, t_5, t_8, t_9]$	.77
16.-22.	...	.77
23.-34.	...	...

Figure 3. Preferred Plans

Thus, plans that include the nurse talking through a mobile device and eventually visiting the patient too end up having better score (0.1) due to the significant importance of patient satisfaction in the preference formula. None of the alternatives in the top half of the list seems to completely satisfy the nurses' desire not to establish any voice connection with the patient. However, if the nurses had been given e.g. the same weight as the management in constructing the WPF, that is 0.5 each, the top plans would at least involve partial satisfaction of the preferences of nurses, namely absence of carrying and talking through a mobile device, which is something that we know (from Figure 2) that they dislike.

Thanks to the presence of CFs in the goal model, the resulting ranking is also sensitive to the original values of the domain predicates, which represent the state of the context. Consider the WPF:

$$\{\Box(valD(avoidPatientDisturb) \leq 0.1)[0.0]\} \times 0.7 \\ + \{\Box(valD(nurseComfort) \leq 0.1)[0.0]\} \times 0.3$$

In initial conditions where $isTime(night)$ does *not* hold, the evaluation of the preference is 0.0 for any plan of the form

length	6	7	8	9	10	20	30	40
Best	≤ 0.1	≤ 0.1	≤ 0.1	≤ 0.1	≤ 0.1	5.6	1.5m	16m
Worse	11	24	10m	1.4h	*	*	*	*

Table I
WORST AND BEST CASE TIMES

$[t_2 \ldots]$. The same plans, however, evaluate the preference to 0.7 if $isTime(night)$ holds initially. In the latter case, plans of the form $[t_1 \ldots]$ are more preferred as they satisfy the WPF with 0.3.

To assess the performance of the reasoning tool, we constructed artificial examples of goal models and preference formulae of arbitrary size and complexity and measured the time that the reasoning procedure takes to produce the first plan. All runs were performed on an AMD Phenom CPU at 2.5GHz with 4MB cache; 1GB of RAM was reserved for each run. The results were found to be sensitive to the maximum plan length the goal model could produce, as well as to whether the preference formula is satisfiable or not - i.e. whether there is a plan that satisfies it with optimal score (0.0) or not. Goal models whose maximum plan length did not exceed 9 tasks would, in the worst case, provide the first result within minutes of computation (or seconds for length of 8 or less) independent of preference satisfiability. From plan lengths of 10 and above, though, we were able to craft "difficult" unsatisfiable preference formulae whose evaluation time would often exceed our 8 hour limit. These worst case results are to be expected given the difficult nature of the problem, and the fact that the PPlan tool performance has not been optimized. In practice, however, the performance for unsatisfiable formulae may vary significantly depending on the amount and structure in the precedence constraints of the goal model. The best case results, in contrast, were more surprising; even with unusually large goal models with plan lengths of 30 tasks, the search could terminate within minutes. As expected, satisfiable formulae and simple precedence structures in goal models that reduce non-determinism, would tend to give such results. Table I summarizes the best and worse times we have observed in our experiments with dummy goal models, with respect to the maximum plan length of the goal model – times are in seconds unless otherwise noted.

VI. CONCLUSIONS

We introduced a modeling and reasoning toolset for exploring and evaluating alternative solutions to requirements problems. A common goal modeling formalism – extended to allow expression of both temporal constraints and partial goal satisfaction – is used for representing alternative plans for fulfilling stakeholder goals. A preference specification language allows users to specify priorities over optional, "nice-to-have" properties of potential solutions. Then, a reasoning tool searches for plans that satisfy the mandatory goals of the goal model while fulfilling the preferences as well as possible. This provides analysts with a means of better understanding the impact of the attitudes of different stakeholders on the design and realization of a system.

In future work, we intend to explore suitable preference elicitation processes, potentially adopting existing work from areas such as AI, Economics and Psychology. We further intend to explore ways to augment our preference specification language and to coincidentally improve the performance of our reasoning tool. We are encouraged by recent advances in preference-based planning that will support this effort.

REFERENCES

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

[2] M. Bienvenu, C. Fritz, and S. McIlraith, "Planning with qualitative temporal preferences," in *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR06)*, 2006, pp. 134–144.

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

[4] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing requirements," *IEEE Software*, vol. 14, no. 5, pp. 67–74, 1997.

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

[6] K. Czarnecki, S. Helsen, and U. Eisenecker, "Staged configuration using feature models," in *Proceedings of the 3rd Software Product Line Conference (SPLC'04)*, 2004, pp. 266–283.

[7] H. Zhang, S. Jarzabek, and B. Yang, "Quality prediction and assessment for product lines." in *Proceedings of the 15th International Conference on Advanced Information Systems Engineering (CAiSE'03)*, 2003, pp. 681–695.

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

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

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

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