Automating Analysis of Qualitative Preferences in Goal-Oriented Requirements Engineering

Zachary J. Oster, Ganesh Ram Santhanam, and Samik Basu
Department of Computer Science, Iowa State University, Ames, Iowa 50011, USA
Email: {zjoster, gsanthan, sbasu}@iastate.edu

Abstract—In goal-oriented requirements engineering, a goal model graphically represents relationships between the required goals (functional requirements), tasks (realizations of goals), and optional goals (non-functional properties) involved in designing a system. It may, however, be impossible to find a design that fulfills all required goals and all optional goals. In such cases, it is useful to find designs that provide the required functionality while satisfying the most preferred set of optional goals under the goal model’s constraints. We present an approach that considers expressive qualitative preferences over optional goals, as these can model interacting and/or mutually exclusive subgoals. Our framework employs a model checking-based method for reasoning with qualitative preferences to identify the most preferred alternative(s). We evaluate our approach using existing goal models from the literature.

I. INTRODUCTION

Goal-oriented requirements engineering [1] is an increasingly popular method for defining the requirements for a system in terms of its goals or objectives. This method has at its foundation a goal model, which must contain (required) goals describing parts of the system’s required functionality and tasks that partially or fully realize a given goal. A goal model may also include optional goals, which represent non-functional requirements and/or “nice-to-have” features, as well as contribution links indicating how required goals contribute to or work against optional goals [2].

The functionality specified by the required goals must be provided for the system to be acceptable, but fulfillment of the optional goals is not required and may be subject to tradeoffs. The preference of a correct design (i.e., a set of tasks that provide the required functionality) depends on the preference of the set of optional goals it fulfills. However, preference analysis quickly becomes difficult without automation when considering goal models for large, complex systems [3]. Because of the conflicts between goals that generally arise in such models, it may be impossible to find a correct design that satisfies all required and optional goals in such a goal model. There is, therefore, a need for automated methods to realize the objective, which is to obtain a correct design $D$ such that there exists no other correct design that is more preferred than $D$.

Qualitative preference valuations, such as those used in [4] and [5], can be specified in an intuitive fashion without specific knowledge about the degree to which one optional goal is preferred to another, as is required by quantitative preferences. A user’s preferences may involve complex tradeoffs between multiple optional goals; these tradeoffs can be represented as a binary relation. In this paper, we present a novel automated goal-model analysis method that finds the most preferred correct design for a system by using a CI-net [6] representation of qualitative preferences that is sufficiently expressive to specify users’ true preferences.

The contributions of our work include:

1) A framework where optional goal preferences are specified using CI-nets [6], a well-studied preference formalism.
2) A novel method based on model checking to rank sets of optional goals using preferences given as CI-nets.
3) Algorithms to identify designs that satisfy the most preferred set of optional goals while meeting the objectives of the system as specified by a goal model.
4) A prototype implementation of the framework, evaluated using case studies from the existing literature.

II. ILLUSTRATIVE EXAMPLE

We use a modified version of the online bookseller goal model in [7] as an example. This model defines several core functions, such as obtaining books to sell, providing price quotes, and receiving payments. Each function is provided by combining components; for instance, to obtain books, the system must contact suppliers, receive price quotes, and then submit book orders. Some functions can be provided in multiple ways, e.g., different payment options.

Non-functional properties, such as transaction costs, customer satisfaction, and use of robust documentation, also affect the success of the system. However, there may be conflicts between non-functional properties: for example, requiring payment by money order instead of credit card reduces transaction costs to the business, but it also reduces customer satisfaction. Realizing this, the bookseller identifies acceptable tradeoffs between non-functional properties:

1) If robust documentation is used, payment traceability is more important than reducing transaction costs.
2) If transaction costs are reduced at the expense of customer satisfaction, then using robust documentation takes precedence over ensuring payment traceability.
3) If robust documentation is not provided, payments should be traceable even at the expense of reduced customer satisfaction and increased transaction cost.

The functional requirements and non-functional properties for the system can be expressed using a goal model as in Figure 1. Although this model can be analyzed by itself to determine acceptable designs for the system, information about the bookseller’s preferences over non-functional properties is also needed to identify the most preferred design.

III. BACKGROUND: GOAL MODELS

A goal model incorporates a set of (required) goals $G^R$, a set of $G^T$, and optionally a set of optional goals $G^O$. A required goal describes a condition, outcome, or state of the world that must be achieved [1], while a task indicates an activity that fulfills or realizes a required goal in full or in part [7]. Relationships between pairs of goals and between goals and tasks combine to form an AND-OR tree, which we call a goal tree. The goal at the tree’s root node (the root goal) represents the total functionality of the system, while subgoals at internal nodes express portions of their parent goal’s functionality (AND-decomposition) or alternatives for providing that functionality (OR-decomposition). Tasks form the leaf nodes of the tree, where each task’s parent is the goal that it realizes. We define a correct design as a set of required goals and tasks that, taken together, are sufficient to fully satisfy the root goal.

Optional goals represent conditions that are desirable but are not vital to the system’s correctness [7], such as non-functional properties. Optional goals need not be organized in a unifying structure (though this is possible), but they are connected to the goal tree by contribution links. Each link is labeled with MAKE $(++)$ if the required goal supports (contributes positively to) the optional goal or BREAK $(--)$ if the required goal denies (contributes negatively to) the optional goal. An optional goal in our model is satisfied if and only if it has (a) no incoming BREAK links from any satisfied required goal and (b) at least one incoming MAKE link from any satisfied required goal.

Figure 1 shows a goal model for our example system. Required goals are denoted by unshaded ovals, which are labeled AND or OR to specify how they are refined into subgoals. Hexagons indicate tasks, shaded ovals denote optional goals, and dotted edges show contribution links.

IV. FINDING THE MOST PREFERRED DESIGN

In our framework, a most preferred correct design is a correct design that satisfies a set of optional goals $\gamma$ such that there exists no other correct design that satisfies another set of optional goals $\gamma'$, where $\gamma'$ is preferred to $\gamma$. We first present a process for deciding which of two sets of optional goals is more preferred according to a given CI-net (Section IV-A). Using this process, we order all subsets of the set of all optional goals $G^O$ from most to least preferred (Section IV-B). We then iteratively consider sets of optional goals $\gamma \subseteq G^O$ in descending order of preference until a correct design that fulfills all optional goals in the current set $\gamma$ is identified (Section IV-C).

A. Preferences over Optional Goals and Dominance Testing

To capture the user’s preferences, we use conditional importance networks (CI-nets) [6], which allow the user to specify relative importance among optional goals. A CI-net $C$ consists of a collection of conditional importance statements of the form: $S^+, S^- : S_1 \succ S_2$, where $S^+, S^-, S_1$ and $S_2$ are pairwise disjoint subsets of $G^O$. Formally, a CI-net $C$ over a set of optional goals $G^O$ is satisfiable if and only if there exists a strict partial order (irreflexive and transitive) relation $\succ$ over the powerset of $G^O$ such that:

1) For each statement $S^+, S^- : S_1 \succ S_2$, if $\gamma \subseteq G^O \setminus (S^+ \cup S^- \cup S_1 \cup S_2)$ then $\gamma \cup S^+ \cup S^- \cup S_1 \succ \gamma \cup S^+ \cup S^- \cup S_2$;

2) $\succ$ is monotonict, i.e., $\gamma \subset \gamma' \Rightarrow \gamma \succ \gamma'$.

Given optional goal sets $\gamma_1$ and $\gamma_2$, deciding whether $\gamma_1$ is preferred over $\gamma_2$ is known as dominance testing. As in [8], our dominance testing approach relies on alternate CI-net semantics given in terms of improving flipping sequences [6]. A sequence of optional goal sets $\gamma_1, \gamma_2, \cdots, \gamma_{n-1}, \gamma_n$ is an improving flipping sequence with respect to a CI-net $C$ if and only if for all $i \in \{1, \ldots, n-1\}$, either (1) $\gamma_{i+1}$ contains at least one more optional goal than $\gamma_i$ (monotonicity flip) or (2) $C$ contains a statement $S^+, S^- : S_1 \succ S_2$ such that (a) $\gamma_{i+1}$ contains the optional goals in both $S^+$ and $S_1$ but not those in $S^-$ and (b) $\gamma_i$ contains the optional goals in both $S^+$ and $S_2$ but not those in $S^-$ (importance flip).

The induced preference graph $\delta(C)$ of a CI-net $C$ is a directed acyclic graph where each node represents a set of optional goals, each edge denotes an “improving flip” where the optional goals at the destination are preferred to those...
Let us explain this procedure using the induced preference graph $\delta(C)$ in Figure 2. In the first iteration, Step 1 initially obtains $\gamma_{11} = \{abcd\}$, the top-most node of $\delta(C)$. In Step 2, model checking proves that $\{abcd\}$ is the most preferred set of optional goals, because all nodes except $\{abcd\}$ can reach another node in $\delta(C)$. In Step 3, we remove the node $\{abcd\}$ from $\delta(C)$, leaving nodes $\{abc\}$, $\{acd\}$, and $\{bcd\}$ at the top.

In Step 1 of the second iteration, the model checker returns $\{abc\}$, $\{acd\}$, or $\{bcd\}$, chosen non-deterministically. Suppose $\{abc\}$ is returned; then the sequence contains $\gamma_{11} = \{abcd\}$ followed by $\gamma_{21} = \{abc\}$. Step 2 consists of three calls to the model checker, which return $\{acd\}$ and then $\{bcd\}$ (or vice versa) followed by no output on the third call; this occurs because $\{abc\}$, $\{acd\}$, and $\{bcd\}$ are equally preferred. In Step 3, we remove nodes $\{abc\}$, $\{acd\}$, and $\{bcd\}$ from $\delta(C)$ and begin a new iteration from Step 1. At this point, the sequence contains $\gamma_{11} = \{abcd\}$, $\gamma_{21} = \{abc\}$, $\gamma_{22} = \{acd\}$, and $\gamma_{23} = \{bcd\}$. This iterative process continues until all nodes originally in $\delta(C)$ are included in the sequence.

C. Finding the Most Preferred Assignment

Recall that a correct design is a set of goals and tasks $G \subseteq G^R \cup G^T$ that fully satisfy the root goal when satisfied themselves. Let $CD(G^R \cup G^T)$ be the set of all correct designs in the goal model. In addition, let $\gamma \subseteq G^O$ be a set of optional goals; then a contributing goal set $\text{Contrib}(\gamma)$ is a set of required goals such that for each optional goal $s_j \in \gamma$, at least one goal in $\text{Contrib}(\gamma)$ supports (has a + link to) $s_j$ and no goal in $\text{Contrib}(\gamma)$ denies (has a − link to) $s_j$. For instance, in Figure 1, $\text{Contrib}(\text{Use Robust Documentation}) = \{\text{Payment Via Money Order}, \text{Send Printed Receipt}\}$.

To identify the most preferred correct design(s) for the root goal, we iterate through all possible sets of optional goals $\gamma_i \subseteq G^O$ in descending order of preference as follows:

1. Let $\gamma_1, \gamma_2, \ldots, \gamma_n$ be the total ordering of optional goal sets from most to least preferred (obtained using the method in Section IV-B).
2. For each $i$ from 1 to $n$, for each $x \in CD(G^R \cup G^T)$, and for each $y \subseteq \text{Contrib}(\gamma_i)$: If $y \subseteq x$, return $x$.
3. If the loop terminates, then no satisfactory assignment was identified for any set of optional goals (even the empty set).

**Theorem 1:** If our algorithm returns a correct design for $\varphi$, then no correct design contributes to a more preferred set of optional goals than the design returned by our algorithm. Furthermore, if our algorithm does not return a correct design for $\varphi$, then no such design exists.

The proof follows directly from the steps described above.

V. IMPLEMENTATION AND RESULTS

We have developed a Java-based tool that implements our goal-model analysis framework; our tool is described in more detail in [10]. Table I summarizes the results obtained by running our tool on modified versions of three goal models from the existing literature on goal-oriented requirements engineering, which describe requirements for an online book
TABLE I
RESULTS OF RUNNING OUR TOOL ON THREE CASE STUDIES

<table>
<thead>
<tr>
<th>Goal Model</th>
<th>Required Goals</th>
<th>Tasks</th>
<th>Optional Goals</th>
<th>CI-net Rules</th>
<th>Mean Total Run Time (s)</th>
<th>Calls to Pref. Reasoner</th>
<th>Mean Time for Pref. Reasoning (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bookseller [7]</td>
<td>13</td>
<td>22</td>
<td>4</td>
<td>3</td>
<td>0.52</td>
<td>3</td>
<td>0.47</td>
</tr>
<tr>
<td>Trentino Transport [4]</td>
<td>24</td>
<td>40</td>
<td>3</td>
<td>3</td>
<td>0.47</td>
<td>2</td>
<td>0.34</td>
</tr>
<tr>
<td>Online Shop [11]</td>
<td>7</td>
<td>16</td>
<td>3</td>
<td>2</td>
<td>0.22</td>
<td>1</td>
<td>0.17</td>
</tr>
</tbody>
</table>

selling service [7] (Figure 1), a generalized online shopping system [11], and a public transport system [4]. We have also used a CI-net specifying new sets of preferences for each goal model that, in our opinion, are reasonable for each goal model’s application domain. All goal models and CI-nets used in our evaluation are available in [10] and at http://www.cs.iastate.edu/~zjost/ase1l.

The times in Table I represent the mean of the running times for 20 runs of our tool over each model under the same configuration (described in [10]). The preference reasoner's CI-net analysis accounts for the bulk of the running time. The time required depends primarily on the number of calls to the preference reasoner, though differences in the number of softgoals and CI-net preference rules also have an effect. However, the goal-model analyzer uses very little additional running time: about 0.05 seconds for the two smaller models and about 0.14 seconds for the transport system model. While we plan to perform additional experiments with larger goal models and more complex preferences to further quantify the effects of goal model and CI-net size on running time, these results show that our framework identifies preferred designs efficiently even though it considers preferences between sets of optional goals (as opposed to [4], [5], and [7], which use preferences between individual optional goals).

VI. DISCUSSION

Other researchers also consider preferences over optional goals within the goal-oriented requirements engineering process. In [7], Liaskos et al. specify mandatory and optional requirements, along with preferences over optional requirements, within goal models that are similar to ours except for some additional concepts (e.g., precedence constraints and optional subgoals of AND-decomposed goals) that can be incorporated directly into our framework. However, [7] uses quantitative preference valuations instead of qualitative ones. Ernst et al. in [5] use qualitative preferences and a technique similar to [4] and our work for converting goal-model analysis into an instance of the SAT problem. In addition to MAKE and BREAK labels, [5] also uses HELP (+) and HURT (–) labels for partial support or denial of optional goals. However, as mentioned before, the qualitative preference models in [4] and [5] can only model simple preferences between two individual optional goals (e.g., $s_A \succ s_B$ always).

Other preference analysis techniques for requirements engineering, such as S-AHP [12] and the cost-value method in [13], are based on the Analytic Hierarchy Process (AHP). While the AHP can quantify preferences and rank the relative importance of individual options, the AHP and its derivatives cannot represent more complex conditional preferences between options, nor can they handle preferences between sets of options. In contrast, CI-nets can express complex preferences qualitatively and use those preferences to partially order all possible sets of options.

Future work includes adding support for additional goal-model concepts, such as “pre”-arcs as used in [7] and the more expressive partial satisfaction semantics used in [5]. We also plan to consider the different preferences of multiple stakeholders within our preference analysis framework and to apply our framework to related software engineering problems such as software product line engineering [14]. We believe our framework can improve the representation and utilization of design preferences in these areas and in others.

REFERENCES