8-12-2011

Automating Analysis of Qualitative Preferences in Goal-Oriented Requirements Engineering

Zachary James Oster
Iowa State University, zach.oster@gmail.com

Ganesh Ram Santhanam
Iowa State University, ganeshram.santhanam@gmail.com

Samik Basu
Iowa State University, sbasu@iastate.edu

Automating Analysis of Qualitative Preferences in Goal-Oriented Requirements Engineering *

Zachary J. Oster Ganesh Ram Santhanam Samik Basu

August 12, 2011

Abstract

The goal model at the core of the goal-oriented approach to requirements engineering graphically represents relationships between the goals (functional requirements), tasks (realizations of goals), and softgoals (non-functional properties) involved in designing a system. It may, however, be impossible to find a design that fulfills all top-level goals and satisfies all softgoals. In such cases, it is useful to find designs that provide the required functionality while satisfying the most preferred set of softgoals under the goal model's constraints. Existing methods typically consider quantitative preferences over softgoals, where the quantification produces a ranking among softgoals. We instead present an approach that considers expressive qualitative preferences over softgoals; unlike quantitative preferences, 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.

1 Introduction

Requirements engineering [1] has become a vital aspect of software engineering with the rapidly increasing complexity of modern software systems. Goal-oriented requirements engineering [2–4] is a useful and increasingly popular method for defining the requirements for a system in terms of the goals or objectives of the system. This method has at its foundation a *goal model* consisting of *goals*, or objectives of the system described in terms of parts of the system's functionality that are required or strongly desired, and *tasks*, each of which realizes or operationalizes a given goal. Many goal models also include *softgoals* (often representing non-functional properties), which are not formally defined and cannot be simply "true" or "false"; furthermore, these goal models contain *contribution links* to illustrate how each goal contributes to or works against the satisfaction of certain softgoals [5].

Research Objective. The functionality specified by the goals in the model must be provided in order for the system to be acceptable, but fulfillment of the non-functional properties represented by the softgoals may be optional or subject to tradeoffs. Consider an online banking service, which would be legally required to provide certain security features. While it is desirable to fulfill the softgoal "Provide Strong Identity Verification" by using additional security measures, it may be impossible to fulfill this softgoal along with the softgoal "Minimize Customer Support Workload". To resolve such conflicts, preferences must be established over the softgoals in a goal model to determine the ordering of the softgoals in terms of priority or desirability. For instance, for the online banking system one might establish that "Provide Strong Identity Verification" is preferred to "Minimize Customer Support Workload". The preference of a design (i.e., a set of goals and tasks) that satisfies all of the required functionality depends on the preference of the set of softgoals it fulfills. The objective, therefore, is to *obtain a design D such that there exists no other design that is more preferred than D*.

*This work is supported in part by U.S. National Science Foundation grant CCF0702758.

Although this analysis may be relatively easy to do manually when working with a small goal model for a simple system, it quickly becomes difficult, if not impossible, without automation when considering a goal model for a large and complex system — the sort of industrial-scale system where requirements engineering techniques are necessary. The goal model for such a system may be quite large, containing hundreds of goals and other concept instances [6]. Such large models are likely to contain conflicting goals and/or softgoals, and it may be difficult to detect the conflicts by manual analysis. Furthermore, because of these conflicts, it is probable that not all softgoals in a large goal model can be fulfilled by any one design (that satisfies a specific set of goals). There is, therefore, a need for automated methods for obtaining the design that is correct (satisfies the required goals) and is most preferred in terms of the preferences specified over softgoals.

Driving Problem. Preference reasoning frameworks use either quantitative or qualitative preference valuations. Quantitative preference valuations, if provided by the end-users, are precise and, to some extent, are intuitively understandable. A number of existing goal-model analysis methods, including [7] and [8], incorporate reasoning over quantitative preferences between softgoals to choose the most preferred set of satisfiable goals. However, in many cases, it may be difficult to quantify (in terms of numbers) the desirability of a softgoal. This is because quantification requires specific knowledge about the *degree* by which a softgoal is preferred to another, but such knowledge may not be readily available. On the other hand, qualitative preferences such as those specified in [9] and [10] can be as intuitive as quantitative ones. The real advantage of qualitative preferences is that the desirability can be specified in an intuitive fashion without the need to quantify the exact extent by which a softgoal or a set of softgoals is desired over another.

However, existing goal-model analysis methods that employ qualitative preferences generally represent qualitative preferences as a binary relation between two softgoals or optional goals, when in fact a user's true preferences may involve more complex tradeoffs that depend on whether other softgoals are fulfilled. In this report, we consider such nuanced qualitative preferences and present a novel method for finding the most preferred design, which is the design that fulfills the most preferred (qualitatively speaking) set of softgoals under the constraint that it must conform to the required goals or objectives of the system. This is the first automated analysis method for goal-oriented requirements engineering that employs such an expressive representation of qualitative preferences among softgoals.

Contributions. The contributions of our work can be summarized as follows:

1. Developing a framework where preferences over softgoals are specified using CI-nets, a well-studied preference formalism.
2. Developing a novel method based on model checking to identify a ranking of softgoals consistent with the preferences expressed in CI-nets.
3. Developing algorithms to identify designs that fulfill the most preferred set of softgoals while meeting the objectives of the system as specified by a goal model.
4. Realizing a prototype implementation of the proposed framework and evaluating the practical application of the framework using three case studies presented in the existing literature.

2 Illustrative Example

We will use a modified version of the bookseller goal model introduced by Liaskos et al. in [7] as a running example throughout this report. Consider a bookseller who wants to open an online retail system. Such a system must provide several core functionalities, such as ensuring that the ordered books are available, providing a price quote for each order, and receiving payment for each order. Each of these core functions is provided by combining one or more smaller components; for instance, in order for the system to obtain books to sell, it must contact suppliers, receive price quotes from those suppliers, and then choose to order the books from some suppliers. Some core functions or portions of those functions can be provided in multiple ways, such as the different payment options commonly offered to customers.

In addition, there are also certain non-functional properties that the bookseller considers relevant to the success of the system. These include reduced transaction costs, customer satisfaction, and use of robust

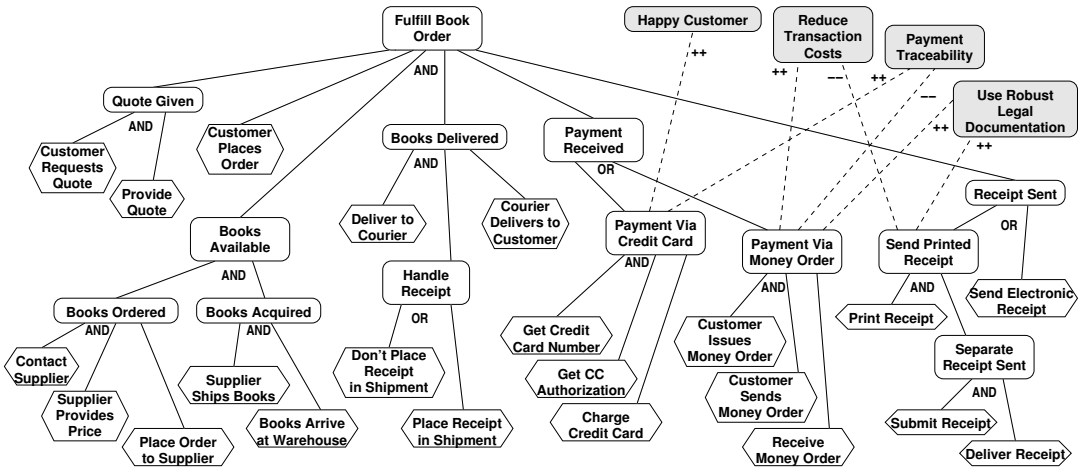

Figure 1: Bookseller goal model taken from [7]

documentation for legal purposes. Unfortunately, it may not be possible to satisfy all of these non-functional properties while still providing all of the core functions required by the system. For example, requiring payment by money order instead of credit card would reduce transaction costs to the business, but it would also reduce customer satisfaction by making payment less convenient. Realizing this, the bookseller has identified several acceptable tradeoffs between the non-functional properties:

1. If robust legal documentation is used, then payment traceability is more important than reducing transaction costs.
2. If transaction costs are reduced at the expense of customer satisfaction, then using robust legal documentation takes precedence over ensuring payment traceability.
3. If robust legal documentation cannot be provided, then it should at least be possible to trace the payments in order to satisfy regulatory auditing requirements, even at the expense of reduced customer satisfaction and increased transaction cost.

The functional requirements and non-functional properties for the online book selling system can be expressed using hard- (functional) and soft- (non-functional) goals in a goal model as in Figure 1. Although this model can be analyzed by itself to determine the set of all acceptable designs for the system, information about the bookseller's preferences over the non-functional properties is also needed to identify the *best* or *most preferred* design, i.e., the set of system components that can be combined in a way that provides the required functionality of the system while fulfilling the most preferred subset of the non-functional properties.

Some goal-model analysis methods require that these preferences be specified explicitly using numeric values, i.e., via quantitative valuations. In contrast, we consider qualitative preferences, which are often more natural for users to provide, and present a method that is capable of identifying the most preferred design. In the following section, we explain the underlying details of both the goal model formalism and the formalism used for expressing qualitative preferences.

3 Background

3.1 Goal Models

Goal models [3, 4] are at the core of the goal-oriented requirements engineering methodology. Recall from Section 1 that a goal model generally incorporates a set of *goals* or more precisely *hardgoals*[1] (G^H), a set of

[1]We will use the terms goal and hardgoal interchangeably.

tasks (G^T), and optionally a set of *softgoals* (G^S). A goal represents a condition, outcome, or state of the world that is to be achieved [4], while a task indicates a certain activity that an actor performs to fulfill a goal [7]. In other words, each task realizes or operationalizes a goal. The relationships between pairs of goals and between goals and tasks combine to form an AND-OR tree, which we will call a *goal tree*. The goal located at the root node of the goal tree (the *root goal*) corresponds to the overall required functionality of the system, while goals at intermediate levels (internal nodes) express portions of the system's functionality or alternatives for providing that functionality. Each goal may be refined into one or more subgoals via AND-decomposition, where a goal is divided into subgoals that must all be satisfied in order to fulfill the original goal, or OR-decomposition, where each subgoal indicates an alternative way to satisfy the original goal. Because tasks specify (in full or in part) how particular goals will be fulfilled, they form the leaf nodes of the goal tree, where each task's parent is the goal that it contributes to realizing. A *design* in our context is a set of tasks that, when taken together, are sufficient to fully satisfy the root goal.

A *softgoal* represents a condition that is not formally defined, is dependent on the fulfillment of one or more unrelated goals, or simply cannot be "true" or "false" in the normal sense [4]; it typically corresponds to a desired non-functional property of the system. Softgoals are not organized in a unifying structure in the goal model, but they are connected to the goal tree by *contribution links* that denote whether a certain goal supports (contributes positively to) or denies (contributes negatively to) a given softgoal. Each contribution link is a dotted-arc extending from a goal in the goal tree to a softgoal. If the goal contributes positively toward the softgoal, the link is labeled with MAKE (`++`); if the goal contributes negatively toward the softgoal, the link is labeled with BREAK (`--`). A softgoal in our model is fulfilled if and only if it has (a) no incoming BREAK links from any satisfied goal and (b) at least one incoming MAKE link from any satisfied goal.

Figure 1 contains a goal model for the online book selling system described in Section 2. Unshaded ovals indicate goals, hexagons indicate tasks, and shaded ovals indicate softgoals. Additionally, each goal node is annotated with `AND` or `OR`, which denotes the way the goal is decomposed into subgoals. For instance, the root goal *Fulfill Book Order*, which represents the required overall functionality of the system, is AND-decomposed into five subgoals and one task. Each of the subgoals must be fulfilled and the task must be performed to realize the root goal. We have not considered a similar AND-OR structure for the softgoals in order to simplify our explanation; however, we do not impose any constraint that prevents such decomposition of softgoals. The contributions of goals to softgoals are denoted by dotted edges. For instance, the goal *Payment via Credit Card* positively contributes to the softgoals *Happy Customer* and *Payment Traceability*, while the goal *Payment via Money Order* positively contributes to the softgoals *Reduce Transaction Costs* and *Use Robust Legal Documentation* (but negatively contributes to the softgoal *Payment Traceability*).

It should be noted that there are other interesting and useful concepts that can be incorporated into goal models, e.g., optional goals, temporal ordering of tasks, and labeled contribution links indicating the degree by which a softgoal is supported or denied. We have not considered them for the purpose of easy and focused explanation of the contributions described in this report. In fact, our techniques for preference specification and analysis are complementary to these concepts and can be directly combined with them.

3.2 Preferences over Softgoals

As noted above, non-functional properties are expressed as softgoals. As the softgoals are not necessary for the functioning of the system, the non-functional requirements are typically specified in terms of relative importance preferences among the softgoals. Ideally, the user would want the entire set of softgoals to be fulfilled by the design. However, this may not be feasible because of the possible existence of softgoals that receive both positive and negative contributions from two or more goals that are part of the same design.

In Figure 1, if a design includes the goals *Payment via Money Order* and *Send Printed Receipt*, then that design cannot fulfill the softgoal *Reduce Transaction Costs*, as the first goal positively contributes to the softgoal while the second goal contributes negatively to the softgoal. Also note that this design will not be able to fulfill the softgoals *Reduce Transaction Costs* and *Payment Traceability* because of the presence of the goal *Payment via Money Order*, which positively contributes to one softgoal and negatively contributes to the other.

In light of such conflicts, a user will typically ascertain preferences over the set of softgoals with the objective of obtaining a functionally correct design that is most preferred. A design can be considered *correct* if the goal at the root of the goal tree is satisfied. The preference of the design, on the other hand, depends directly on the set of softgoals fulfilled by the design. Therefore, the most preferred correct design is one that fulfills a set of softgoals γ such that there exists no other correct design that fulfills another set of softgoals γ', where γ' is preferred to γ.

CI-nets Syntax and Semantics. In order to capture and reason with the user's preferences over the set of softgoals, we use *conditional importance networks* (CI-nets) [11], which allow the user to specify relative importance among softgoals. 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^S. Informally, such a statement is read as: Given two designs, if both fulfill the softgoals in S^+ and do not fulfill the softgoals in S^-, then the design that fulfills all softgoals in S_1 is preferred to the design that fulfills all softgoals in S_2. For instance, the preference described in Section 2, *"if transaction costs are reduced at the expense of customer satisfaction, then using robust legal documentation takes precedence over ensuring payment traceability,"* is expressed in the language of CI-nets as follows:

$$\{\textit{Reduced Transaction Cost}\}, \{\textit{Happy Customer}\} : \\ \{\textit{Use Robust Legal Documentation}\} \succ \{\textit{Payment Traceability}\}$$

Formally, a CI-net C over a set of softgoals G^S is *satisfiable* if and only if there exists a strict partial order (irreflexive and transitive) relation $\succ$ over the powerset of G^S such that:

1. For each CI-net statement $S^+, S^- : S_1 \succ S_2$, if $\gamma \subseteq G^S \backslash (S^+ \cup S^- \cup S_1 \cup S_2)$ then $\gamma \cup S^+ \cup S_1 \succ \gamma \cup S^- \cup S_2$;
2. $\succ$ is monotonic, i.e., $\gamma \subset \gamma' \Rightarrow \gamma \succ \gamma'$

Going back to the preference statement above, as per the rule in item 1, the set of softgoals {*Reduced Transaction Cost, Use Robust Legal Documentation*} is preferred to ($\succ$) the set of softgoals {*Reduced Transaction Cost, Payment Traceability*}. As per the rule in item 2, a set of softgoals is preferred to all of its proper subsets.

CI-nets for softgoal preferences. CI-nets are a natural choice for modeling user preferences over softgoals in goal models for the following reasons:

1. Preferences in CI-nets are monotonic. According to the semantics of CI-nets, a set of softgoals is preferred to all its proper subsets. This property is necessary to model preferences over softgoals because the user would typically like to see as many (ideally all) softgoals fulfilled as possible.
2. The CI-net semantics induces a strict partial order among the subsets of softgoals with respect to the conditional importance statements of the user. Thus, it is possible to order the subsets of softgoals in a way that is consistent with the semantics of a CI-net. Such an ordering can be used to search for designs that fulfill more preferred softgoals ahead of the ones that fulfill less preferred softgoals.
3. When it is not possible to find a design that satisfies all the subgoals, users are often willing to trade certain sets of softgoals for other sets of softgoals. This can be directly represented in CI-nets using conditional importance statements over sets of softgoals.

4 Finding the Most Preferred Design

In this report, we devise a method that automatically identifies the most preferred correct design which has the following properties. First, it is correct, i.e., it satisfies the goal at the root of the goal tree. Second, it fulfills a set of softgoals γ such that there exists no other correct design that fulfills another set of softgoals γ', where γ' is preferred to γ. Our method is summarized as follows:

1. *Decide:* Automatically decide the preference of a set of softgoals over another, where preferences are specified using CI-nets.

2. *Order:* Use the above decision process to automatically identify the preference ordering of sets of softgoals, from most preferred to least preferred.

3. *Iterate:* Use the above ordering process and automatically identify the most preferred correct design(s) by iteratively considering the sets of softgoals in the descending order of their preference.

Section 4.1 presents an effective decision technique, Section 4.2 discusses the ordering technique, and finally Section 4.3 describes the iteration technique for identifying the most preferred design(s).

4.1 Dominance Testing

Given two choices (of sets of softgoals), deciding the preference of one choice over the other is referred to as *dominance testing*. Dominance testing is known to be PSPACE-complete [11, 12]. Recently, in [13], we have demonstrated an effective model checking [14] based approach to dominance testing for certain families of preferences, such as TCP-nets [15]. In this report, we follow a similar approach for dominance testing between choices (of sets of softgoals) where preferences are represented using CI-nets. This approach relies on alternate semantics of CI-nets given in terms of an *improving flipping sequence*, analogous to the worsening flipping sequence defined in [11].

Definition 1 (Improving Flipping Sequence [11]) *A sequence of sets of softgoals $\gamma_1, \gamma_2, \cdots \gamma_{n-1}, \gamma_n$ is an* improving flipping sequence *with respect to a set of CI-net statements if and only if, for $1 \leq i < n$, either*

1. (*Monotonicity Flip*) $\gamma_{i+1} \supset \gamma_i$*; or*

2. (*Importance Flip*) *there exists a conditional importance statement $S^+, S^- : S_1 \succ S_2$ in the CI-net, s.t.*

   (a) $\gamma_{i+1} \supseteq S^+$, $\gamma_i \supseteq S^+$, $\gamma_{i+1} \cap S^- = \gamma_i \cap S^- = \emptyset$*;*

   (b) $\gamma_{i+1} \supseteq S_1$, $\gamma_i \supseteq S_2$, $\gamma_{i+1} \cap S_2 = \gamma_i \cap S_1 = \emptyset$*;*

   (c) if $\gamma = G^S \setminus (S^+ \cup S^- \cup S_1 \cup S_2)$ *then* $\gamma \cap S_1 = \gamma \cap S_2$. □

In the above definition, condition (1) states that fulfilling additional softgoals is always preferred to fulfilling fewer softgoals. Condition (2) states that if the set S^+ of softgoals are fulfilled and the set S^- of softgoals are not fulfilled, then fulfilling the set S_1 of softgoals is more important than fulfilling the set S_2 of softgoals, all others being equal (which is ensured by condition (2c)). Given a CI-net C and two sets γ and γ' of softgoals, we say that γ is preferred to γ', denoted by $C \models \gamma \succ \gamma'$, iff there is an improving flipping sequence with respect to C from γ' to γ (Proposition 1, [11]). In our example CI-net (see Section 2), we can thus say that the set {*Reduced Transaction Cost*, *Use Robust Legal Documentation*} is preferred to the set {*Payment Traceability*}. This is because the set {*Payment Traceability*} has an improving (importance) flip to the set {*Reduced Transaction Cost*, *Payment Traceability*}, which in turn has an improving (monotonicity) flip to {*Reduced Transaction Cost*, *Use Robust Legal Documentation*}.

From the above definition, one can construct a graph where each node corresponds to a set of softgoals and each directed edge from one node to another denotes an "improving flip" capturing the fact that the set of softgoals at the destination node is preferred to the set of softgoals at the source node. This graph is referred to as the *induced preference graph*.

Definition 2 (Induced Preference Graph) *Given a CI-net C over a set of softgoals G^S, the* induced preference graph *$\delta(C) = (N, E)$ is constructed as follows. The nodes N correspond to the powerset of G^S, and each directed edge $(\gamma, \gamma') \in E$ corresponds to an improving (monotonicity or importance) flip from γ to γ' as per the CI-net semantics (Definition 1) such that $\gamma' \succ \gamma$.* □

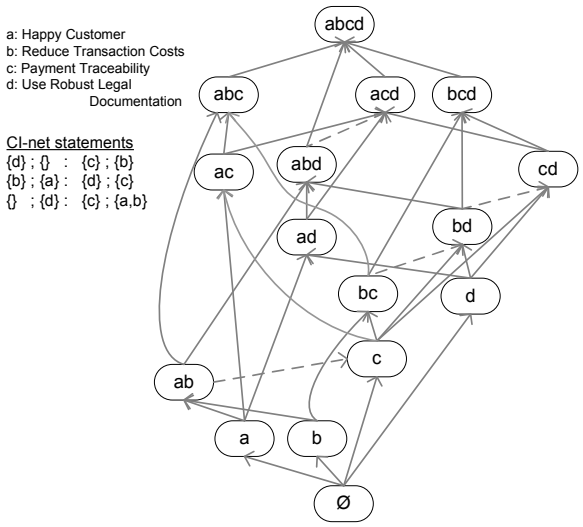

Figure 2: Induced preference graph

Figure 2 presents the CI-net statements corresponding to the preferences over softgoals specified in Section 2, along with the corresponding induced preference graph. The solid edges between sets of softgoals in this graph correspond to monotonicity flips and the dotted edges correspond to importance flips. Consequently, a path in the graph corresponds to an improving flipping sequence.

A set γ' of softgoals *dominates* (i.e., is preferred to) another set γ with respect to CI-net C if and only if the node corresponding to γ' in $\delta(C)$ is reachable from γ. For example, the set {*Reduced Transaction Cost*, *Use Robust Legal Documentation*} is preferred to the set {*Payment Traceability*} due to the existence of the path $c \rightarrow bc \rightarrow bd$ (see Figure 2). Because the CI-net semantics induces a strict partial order relation over the powerset of softgoals, $\delta(C)$ is acyclic.

Dominance Testing via Model Checking. We use a model checker, specifically NuSMV [16], to verify reachability (and therefore dominance) from one node to another in the induced preference graph. There are two primary advantages in using NuSMV for testing dominance. First, NuSMV is equipped with (symbolic or BDD-based) algorithms that allow for efficient state-space exploration of large graphs. Second, NuSMV can verify properties (beyond reachability) in expressive temporal logic (e.g., CTL or LTL), a capability that we will use in Section 4.2 to obtain a preference ordering over sets of softgoals (see item 2 of the summary at the start of this section).

The input model of NuSMV is a Kripke structure $\langle S, S_0, T, L \rangle$, where S is the set of states, $S_0 \subseteq S$ is the set of start states, $T \subseteq S \times S$ is the set of transition relations, and L is a labeling function mapping each state in S to a set of propositions that hold at that state. In our encoding, we represent each softgoal (say, x_i) as a proposition. Given a set of CI-net statements C, the Kripke structure K_C that represents the induced preference graph $\delta(C)$ contains states that are labeled with the truth-values of the set of softgoal propositions x_i's, a set of Boolean variables h_i's, and a Boolean variable g.

Each state in K_C corresponds to a node in $\delta(C)$. For instance, if $x_3 \wedge x_4$ holds (i.e., evaluates to true) in a state in K_C, then that state corresponds to the node annotated with x_3 and x_4 in $\delta(C)$. Note that the condition for the existence of an edge in $\delta(C)$ depends on the contents of the source and destination nodes (improving flip, see Definition 1). Direct encoding of such edges in the form of Kripke structure transitions is not possible in NuSMV, as NuSMV does not allow specifying transition conditions on the basis of the propositions that hold (or do not hold) at the destination states. As a result, we have used auxiliary variables h_i, one for each softgoal proposition x_i. Each h_i is encoded such that if h_i is `false` in the current state, then in the next state the valuation of x_i cannot change; otherwise, the valuation of x_i may or may not change in the next state. The initialization and updates to the h_i's are performed non-deterministically by the model checker. Details of this encoding and further explanation are available at `http://www.cs.iastate.edu/~zjoster/ase11`.

According to the above encoding, the different truth-valuations of each h_i for the same truth valuation of each x_i correspond to states in K_C that allow different ways in which the valuation of that x_i can be changed. Consequently, K_C contains multiple states where an identical set of x_i's hold true; all of these states correspond to one node in the induced preference graph $\delta(C)$. Transitions between these states do not result in a change in the valuations of propositions x_i and, therefore, do not correspond to any edge in $\delta(C)$. The variable g is set to `true` whenever a transition traversed in K_C results in a change in the valuation of at least one of the x_i's (i.e., when a transition in K_C corresponds to an edge in the induced preference graph $\delta(C)$).

Given a CI-net C, we first construct K_C, which is provided as the input model to the NuSMV model checker. For any two sets of softgoals γ_i and γ_j, we use the following CTL formula to check whether γ_j is preferred to γ_i: $X_i \Rightarrow \texttt{EF}(X_j)$, where X_i (resp. X_j) is the set of propositions corresponding to the softgoals in γ_i (resp. γ_j). This property is satisfied by any state in K_C where X_i holds true and where there is a path leading to a state where X_j holds true. If the property is satisfied, we conclude that γ_j is preferred to γ_i.[2] Otherwise, there exists no improving flipping sequence from γ_i to γ_j, i.e., γ_j is *not* preferred to γ_i. In the CI-net used in our example (see Section 2), querying the model checker with the CTL formula $\neg(acd \Rightarrow \texttt{EF}bd)$ yields the counter-example corresponding to the path $bd \rightarrow cd \rightarrow acd$.

We find the most preferred set of softgoals by verifying the CTL property $\texttt{AGEF}(g = \texttt{true})$ in K_C. This property is satisfied at a state s in K_C if and only if all states s' reachable from s can, in turn, reach some state where g evaluates to true. The property is not satisfied at states in K_C that correspond to the *top-most node* (containing the most preferred set of softgoals) of the induced preference graph (see, for instance, Figure 2). This is because the top-most node in $\delta(C)$ does not contain any outgoing edges. Any one of the states in K_C that corresponds to the top-most node in $\delta(C)$ is identified by NuSMV as a counter-example, proving the unsatisfiability of the property $\texttt{AGEF}(g = \texttt{true})$. In our running example, this query returns the state where the state variables a, b, c and d are `true`.

4.2 Preference Ordering over Sets of Softgoals

Once we have modeled the induced preference graph $\delta(C)$ as a Kripke model K_C input to the NuSMV model checker, our next objective is to obtain an ordering of sets of softgoals from most preferred to least preferred. Note that $\delta(C)$ represents a strict partial order between sets of softgoals. The ordering we obtain is a total order consistent with this strict partial order. We achieve this by performing model checking on the model K_C and its modifications against CTL properties.

The steps in our approach are as follows.

1. We verify K_C against the CTL property $\texttt{AGEF}(g = \texttt{true})$. As noted above, this returns the most preferred set of softgoals (say γ_i) from the top of $\delta(C)$. In general, as $\delta(C)$ is a strict partial order, it may have multiple elements at the top. Any state that corresponds to any one of the top elements will be returned as the counter-example (proving the unsatisfiability of the CTL property).

2. Let $\gamma_1, \gamma_2, \ldots, \gamma_n$ be the sequence of sets of softgoals that has been obtained so far (as the total order consistent with the partial order presented in $\delta(C)$). We define the following formula

$$I = \bigvee_{i=1}^{n} \bigwedge_{j} (X_{ij})$$

where X_{ij} is the set of propositions corresponding to γ_i. We then query the model checker with the modified CTL property $\texttt{AGEF}(g = \texttt{true}) \vee I$. The property is satisfied by state s in K_C if and only if (a) all states s' reachable from s can, in turn, reach some state where g is true or (b) s corresponds to nodes $\gamma_1, \gamma_2, \ldots, \gamma_n$ in $\delta(C)$. On the other hand, if the property is not satisfied by s, then s cannot reach

[2] An improving flipping sequence from γ_i to γ_j can be obtained from the model checker by querying the model checker with the negation of the formula $X_i \Rightarrow \texttt{EF}(X_j)$. The counter-example to this formula returned by the model checker is a path in the Kripke structure that proves dominance, which can be used to construct the improving flipping sequence.

a state where g is set to true and s does not correspond to nodes $\gamma_1, \gamma_2, \ldots, \gamma_n$. In other words, if the property is satisfied, there exists no state corresponding to a set of softgoals that is at least as preferred as at least one element in $\gamma_1, \gamma_2, \ldots, \gamma_n$; otherwise, the model checker produces a counterexample, which is a state corresponding to a set of softgoals, γ_{n+1}.

3. If a γ_{n+1} is obtained in the previous step, we iterate Step 2 using the new sequence $\gamma_1, \gamma_2, \ldots, \gamma_{n+1}$. Otherwise, we remove from the Kripke structure K_C all the states corresponding to the softgoals $\gamma_1, \gamma_2, \ldots, \gamma_n$ (obtained by iterating Step 2 so far) by adding $\neg I$ as an invariant to the Kripke structure (the model checker only considers the states where the invariant holds). Thus, the reduced model corresponds to the induced preference graph where the nodes corresponding to $\gamma_1, \gamma_2, \ldots, \gamma_n$ are not considered. We then iterate starting from Step 1 until the invariant results in a model where no states are considered by the model checker.

Note that in Step 2, the states in the model corresponding to $\gamma_1 \ldots \gamma_n$ are *ignored* (although they are present in the model) by the model checker, as our query is modified to consider states except these. This enables us to obtain the top-most nodes one by one in sequence without altering the model. However, when all the top-most nodes are obtained, we *remove* the states corresponding to $\gamma_1 \ldots \gamma_n$ from the model in Step 3 (by adding the $\neg I$ as an invariant to the current model). This modification of the model makes it possible for us to obtain the next set of top-most nodes in the following iteration. We explain the above steps using the example $\delta(C)$ presented in Figure 2.

Iteration 1: Initially, the Kripke structure K_C encoding of $\delta(C)$ is model-checked following Step 1 above. The result (counterexample) obtained is the top-most element $\gamma_{11} = (abcd)$. In Step 2, model checking is performed again with the property $\texttt{AGEF}(g = \texttt{true}) \vee I$, where $I = (a \wedge b \wedge c \wedge d)$. The property is satisfied because all states except the ones corresponding to $(abcd)$ can reach a state where $g = \texttt{true}$. Therefore, as per Step 3, we remove from K_C the states corresponding to the node $(abcd)$ by adding $\neg I = (\neg a \vee \neg b \vee \neg c \vee \neg d)$ as an invariant to K_C. As a result, we have forced the model checker to consider only the states where the invariant holds, i.e., at any state corresponding to the node $(abcd)$, the invariant does not hold. This can be viewed as an updated K_C which encodes a $\delta(C)$, where the nodes $((abc), (acd), (bcd))$ are at the top.
Iteration 2: Step 1 is performed again and the model checker returns as a counterexample one of the states that corresponds to either (abc), (acd), or (bcd). Note that such a state is identified non-deterministically by the model checking algorithm. Suppose that the state corresponding to (abc) is obtained as a counterexample. So far, we have $\gamma_{11} = (abcd)$ (obtained in the previous iteration) followed by $\gamma_{21} = (abc)$ in our total ordering of sets of softgoals. Proceeding to Step 2, we have a new $I = (a \wedge b \wedge c \wedge d) \vee (a \wedge b \wedge c)$. Note that the states corresponding to $(abcd)$ have already been removed (by adding the invariant $\neg a \vee \neg b \vee \neg c \vee \neg d$) before the start of Step 1 in the current iteration. As a result, we do not need to consider the first disjunct in the new I. When model checking is performed again, one of the states corresponding to either (acd) or (bcd) is obtained as a counterexample. Suppose that a state corresponding to $\gamma_{22} = (acd)$ is returned as a counterexample.

We proceed to perform Step 2 again with $I = (a \wedge b \wedge c) \vee (a \wedge c \wedge d)$. The model checker returns a counterexample state corresponding to the node $\gamma_{23} = (bcd)$. Proceeding further, Step 2 is again performed using $I = (a \wedge b \wedge c) \vee (a \wedge c \wedge d) \vee (b \wedge c \wedge d)$. At this point, the model checker fails to find any counterexamples for the property $\texttt{AGEF}(g = \texttt{true} \vee I)$. In Step 3, we remove all the states corresponding to the nodes $(abc), (acd)$ and (bcd) by adding the invariant $\neg I = (\neg a \vee \neg b \vee \neg c) \wedge (\neg a \vee \neg c \vee \neg d) \wedge (\neg b \vee \neg c \vee \neg d)$ to the model, and we start a new iteration from Step 1. So far, we have obtained an ordering of sets of softgoals $\gamma_{11} = (abcd), \gamma_{21} = (abc), \gamma_{22} = (acd), \gamma_{23} = (bcd)$.

The iterative process (starting from Step 1) is continued until $\neg I$ results in a K_C where no states are considered by the model checker. The number of such iterations is equal to the height of the partial order in $\delta(C)$. In the example (Figure 2), it is equal to 9. Each such iteration obtains a sequence of equally preferred (or indistinguishable as per the given preferences) sets of softgoals. For instance, in iteration 2, we obtained $(abc), (acd)$ and (bcd), which are equally preferred. Such elements are obtained by iterating Step 2 multiple times each time with a new value of I. The maximum number of iterations starting at Step 2 is equal to the width of the partial order in $\delta(C)$. In the example (Figure 2), it is equal to 3.

The main advantage of using the above method is that a total ordering of sets of softgoals is obtained without performing all possible pairwise comparisons. Instead, systematic updates to the model corresponding to the induced preference graph and repeated model checking using a CTL property are used to automatically and effectively find the total order of sets of softgoals.

4.3 Finding the Most Preferred Assignment

We now have effective methods for finding the most preferred set of softgoals and for computing a sequence of softgoals that forms a total ordering over the powerset of softgoals and is consistent with the stated CI-net preferences. Before we proceed to find the most preferred satisfiable set of goals (i.e., the most preferred design), we define some additional concepts. Let $\mathcal{P}(S)$ denote the powerset of set S.

1. Let φ be a Boolean formula representing the Boolean combination of the goals in the goal model, where each proposition φ_i in the formula indicates the corresponding goal $g_i \in G^H$ (set of hardgoals in the goal tree). Then a *satisfiable goal assignment* is any set of goals $\hat{G}^H \subseteq G^H$ such that setting the corresponding Boolean propositions to true satisfies φ. Let $Sat(\varphi) = Sat(G^H)$ denote the set of all satisfiable goal assignments in $\mathcal{P}(G^H)$. For instance, $Sat(Books\ Delivered)$ is a set containing two different solution; one solution contains the goal *Don't Place Receipt in Shipment* and the other contains the goal *Place Receipt in Shipment*. This is because either one of these goals must be satisfied in order to satisfy the goal *Handle Receipt*, satisfaction of which is necessary to satisfy the root goal *Books Delivered*.

2. Let $\gamma \subseteq G^S$ be a set of softgoals. A *contributing goal set*, denoted by $Contrib(\gamma)$, is a set of goals (equivalently, a set of truth assignments to goal propositions in φ) that, taken together, contribute positively to (support) every softgoal in the set γ. In other words, for every softgoal $s_j \in \gamma$, both of the following hold:

   (a) At least one goal in the set supports (has a `++` link to) s_j.

   (b) No goal in the set denies (has a `--` link to) s_j.

   For instance, in Figure 1,

   $$Contrib(Use\ Robust\ Legal\ Documentation) = \{Payment\ Via\ Money\ Order, Send\ Printed\ Receipt\}$$

With the help of the above concepts and the method described in Section 4.2 for computing a total preference ordering over the powerset of softgoals using CI-nets, we present our algorithm to find the most preferred functionally satisfactory assignment of goals in the goal tree. The aim is to identify a satisfiable goal assignment for the root of the goal tree such that the assignment also contributes to the most preferred subset of softgoals G^S. In order to achieve this, we iterate through the possible sets of softgoals in $\mathcal{P}(G^S)$ in descending order of preference (using the method described in Section 4.2) with respect to the CI-net specification. The algorithm proceeds as follows:

1. Let φ be the Boolean formula described above which, when satisfied, results in the satisfaction of the root of the goal tree. Let $\gamma_1, \gamma_2, \ldots, \gamma_n$ be the total order sequence of sets of softgoals from most preferred to least preferred obtained using the method in Section 4.2.

2. For each i starting from 1 to n do:

   (a) For each $x \in Sat(\varphi)$ and $y \in Contrib(\gamma_i)$, if $y \subseteq x$, then return x.

3. If the loop terminates, then no satisfiable assignment was identified for any set of softgoals (including the empty set); inform the user and return.

Goal Model	*Goals*	*Tasks*	*Softgoals*	*CI-net Rules*	*Mean Total Run Time (s)*	*Calls to Preference Reasoner*	*Mean Time for Pref. Reasoning (s)*
Bookseller [7]	13	22	4	3	0.52	3	0.47
Trentino Transport [9]	24	40	3	3	0.47	2	0.34
Online Shop [17]	7	16	3	2	0.22	1	0.17

Table 1: Results of running our tool on three case studies

In the above procedure, Step (2) is iterated until either (a) a satisfiable goal assignment that contributes to all the softgoals in some γ_i (considered in descending order of preference) is found; or (b) no satisfiable goal assignment contributes to all the softgoals in any γ_i. Note that in the latter case, indeed there is no satisfiable goal assignment because the procedure considers *all* possible subsets of softgoals one by one, which includes the set $\gamma_n = \emptyset$ (i.e., none of the softgoals are fulfilled; least preferred). In other words, the above procedure finds a satisfiable goal assignment, if one exists. Further, the procedure considers sets of softgoals in descending order of preference, and hence the procedure will find a satisfiable goal assignment (if one exists) that fulfills a more preferred set γ_i of softgoals ahead of other assignments that fulfill the set γ_j $(j > i)$, where γ_i is preferred to γ_j.

This establishes the correctness of our procedure and its optimality in finding the most preferred satisfiable goal assignment — in other words, the most preferred design.

Theorem 1 (Soundness and Completeness) *If our algorithm returns a satisfiable assignment to φ, then there is no satisfiable goal assignment that contributes to a more preferred set of softgoals than the assignment returned by our algorithm. Furthermore, if our algorithm does not return a satisfiable assignment to φ, then no satisfiable assignment to φ exists.* □

The proof of the theorem directly follows from the steps of our method described above.

5 Implementation and Results

5.1 Tool Implementation

We have developed a tool in Java, which is available at `http://www.cs.iastate.edu/~zjoster/ase11`, to implement our approach for finding the most preferred goal assignments for a goal tree. The tool's architecture comprises two main modules: a *Goal Model Analyzer* module that serves as the tool's front-end and a *Preference Reasoner* module that provides the tool's back-end functionality. The Preference Reasoner uses the model checker NuSMV to compute the next-most-preferred softgoal set at each step of the goal model analysis. This module reads a text file containing a CI-net specification and automatically generates the Kripke structure in the input language of NuSMV. It implements the procedure described in Section 4.2.

The Goal Model Analyzer constructs the goal model, including the goal tree (goals and tasks), softgoals, and contribution links, from a text input file. Once finished with this task, the Goal Model Analyzer executes the algorithm given in Section 4.3, obtaining preference data from the Preference Reasoner. One important aspect of the Goal Model Analyzer and its interaction with the Preference Reasoner is that the latter communicates the best (most preferred) set of softgoals to the former and waits until it is instructed to compute the next best set of softgoals. In other words, the Preference Reasoner does not compute the entire total order sequence of sets of softgoals; instead, it computes one and sends it to the Goal Model Analyzer. If the Goal Model Analyzer cannot find a satisfiable assignment that contributes to the set of softgoals, it communicates with the Preference Reasoner to obtain the next set of softgoals in the sequence of the total order. The Goal Model Analyzer eventually returns either the most preferred satisfiable goal assignment(s) or a message stating that no satisfiable goal assignment could be found.

The Preference Reasoner and Goal Model Analyzer are loosely coupled, allowing different approaches for preference reasoning or goal model analysis to be substituted into the existing tool with minimal effort. In addition, the data structures for the goal model are designed for extensibility. This will simplify the process of adding support for additional concepts to the Goal Model Analyzer in the future.

5.2 Preliminary Results

We have tested our implementation of our goal-model analysis framework against modified versions of three goal models from the existing literature on goal-oriented requirements engineering. These goal models describe requirements for an online book selling service [7] (Figure 1), a generalized online shopping system [17], and a public transport system [9]. 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 are available at `http://www.cs.iastate.edu/~zjoster/ase11`.

Table 1 summarizes the results obtained by running our analysis tool on each goal model. The tests were executed on a Gateway laptop with 4 GB of RAM and an Intel Core 2 Duo T5550 dual-core CPU (1.83 GHz), running 64-bit Windows Vista Service Pack 2. The times shown in Table 1 represent the mean of the running times reported for 20 runs of our tool over each model under the same configuration. It is clear that the preference reasoner's CI-net analysis accounts for the bulk of the running time. The time required for preference reasoning appears to depend primarily on the number of calls to the preference reasoner, although differences in the number of softgoals and CI-net preference rules will 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 much larger 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 preliminary results show that our goal model and preference analysis framework is as efficient as other comparable goal model analysis techniques such as [9] and [10], even though it supports more expressive preference specifications than those techniques.

6 Discussion

Since it is not always possible to fulfill all softgoals in a goal model, it is frequently necessary to consider preferences between different softgoals in order to design a system that both provides the required functionality and fulfills the most preferred subset of the softgoals in the goal model. Despite the increasing research interest in goal-oriented requirements engineering during the past several years, there remains a need for automated methods for analyzing the effects of preferences over softgoals in a goal model. In this report, we presented a framework for specifying and reasoning over such preferences using CI-nets. Our framework includes a novel model checking-based method for ranking softgoals according to the specified preferences, as well as an algorithm for identifying designs that fulfill the most preferred set of softgoals while also ensuring that the functional requirements of the system are met. We have implemented this framework in Java, using NuSMV to perform the preference analysis, and we have evaluated the performance of our method using three case studies from the requirements engineering literature. We have also formally proven the correctness of our framework's results.

Other researchers have also taken an interest in considering preferences over softgoals as part of the goal-oriented requirements engineering process. In [7], Liaskos et al. present a framework for specifying both mandatory and optional requirements, along with quantitative preferences over the optional requirements, within the context of a goal model. The goal tree and the specified preferences are translated into the Hierarchical Task Network (HTN) and Planning Domain Description Language (PDDL) planning formalisms, respectively; the HTNPlan-P planning tool is then used to obtain the most preferred design. Unlike our work, [7] makes no distinction between (hard)goals and softgoals; instead, the concept of an "optional goal" is used to represent softgoals as well as hardgoals that are not part of the mandatory functional requirements for the system. Otherwise, the goal model definition used in [7] is similar to the definition used in this report, although their model also includes support for precedence constraints ("pre"-arcs) and optional

subgoals of AND-decomposed goals; however, these additional concepts can be directly incorporated into our framework. Our framework goes beyond the framework in [7] by using expressive qualitative preferences to identify preferred designs, instead of using quantitative preference values that are less powerful and may be less accurate.

The work of Ernst et al. in [10] is similar to ours in that they also use qualitative preference valuations and a technique similar to that in [9] for converting the process of identifying satisfiable goal assignments into an instance of the SAT problem. [10] uses a more expressive labeling scheme for contribution links, allowing HELP (+) and HURT (-) labels for partial support or denial of a goal in addition to MAKE and BREAK labels. However, the qualitative preference model in [10] is much less expressive, as it only models simple preferences between any two goals (e.g., $s_A \succ s_B$ always); in contrast, the CI-net formalism used in our framework can also represent more nuanced preferences among softgoals (e.g., $s_A \succ s_B$ unless s_C is fulfilled, in which case $s_B \succ s_A$). The SAT-solving goal-model analysis framework in [9] and the abstract requirement modeling language Techne [18] similarly model preferences as a binary relation and, therefore, have similar constraints on the expressive power of their preference specifications.

Future work includes adding support for additional goal-model concepts, such as "pre"-arcs and optional goals as used in [7] and the more expressive partial satisfaction semantics used for softgoals in [10]. We would also like to add an easy-to-use GUI to make it easier to apply our tool to larger, more realistic goal models for industrial-scale systems. In addition, we are interested in exploring the applicability of our preference analysis framework to other formalisms, e.g., feature diagrams [19], as well as to other related software engineering problems such as software product line engineering [20]. We believe our framework can significantly improve the representation and utilization of design preferences in these areas and in others.

References

[1] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*. John Wiley and Sons, 2009.

[2] ——, "Goal-oriented requirements engineering: A guided tour," in *RE*. IEEE Computer Society, 2001, pp. 249–263.

[3] 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.

[4] E. S. K. Yu and J. Mylopoulos, "Understanding 'why' in software process modelling, analysis, and design," in *ICSE*, 1994, pp. 159–168.

[5] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*. Kluwer Academic, 2000.

[6] A. van Lamsweerde, "Goal-oriented requirements enginering: A roundtrip from research to practice," in *RE*. IEEE Computer Society, 2004, pp. 4–7.

[7] S. Liaskos, S. A. McIlraith, S. Sohrabi, and J. Mylopoulos, "Integrating preferences into goal models for requirements engineering," in *RE*. IEEE Computer Society, 2010, pp. 135–144.

[8] E. Letier and A. van Lamsweerde, "Reasoning about partial goal satisfaction for requirements and design engineering," in *SIGSOFT FSE*, R. N. Taylor and M. B. Dwyer, Eds. ACM, 2004, pp. 53–62.

[9] R. Sebastiani, P. Giorgini, and J. Mylopoulos, "Simple and minimum-cost satisfiability for goal models," in *CAiSE*, 2004, pp. 20–35.

[10] N. A. Ernst, J. Mylopoulos, A. Borgida, and I. Jureta, "Reasoning with optional and preferred requirements," in *ER*, ser. Lecture Notes in Computer Science, J. Parsons, M. Saeki, P. Shoval, C. C. Woo, and Y. Wand, Eds., vol. 6412. Springer, 2010, pp. 118–131.

[11] S. Bouveret, U. Endriss, and J. Lang, "Conditional importance networks: A graphical language for representing ordinal, monotonic preferences over sets of goods," in *International Joint Conference on Artificial Intelligence*, 2009, pp. 67–72.

[12] J. Goldsmith, J. Lang, M. Truszczynski, and N. Wilson, "The computational complexity of dominance and consistency in CP-nets," *JAIR*, vol. 33, pp. 403–432, 2008.

[13] G. R. Santhanam, S. Basu, and V. Honavar, "Dominance testing via model checking," in *AAAI*. AAAI Press, 2010, pp. 357–362.

[14] E. Clarke, O. Grumberg, and D. Peled, *Model Checking*. MIT Press, January 2000.

[15] R. I. Brafman, C. Domshlak, and S. E. Shimony, "On graphical modeling of preference and importance," *J. Artif. Intell. Res. (JAIR)*, vol. 25, pp. 389–424, 2006.

[16] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, "NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking," in *Proc. Intl. Conf. on Computer-Aided Verification*. Copenhagen, Denmark: Springer, July 2002.

[17] S. Liaskos, M. Litoiu, M. D. Jungblut, and J. Mylopoulos, "Goal-based behavioral customization of information systems," in *CAiSE*, ser. Lecture Notes in Computer Science, H. Mouratidis and C. Rolland, Eds., vol. 6741. Springer, 2011, pp. 77–92.

[18] I. Jureta, A. Borgida, N. A. Ernst, and J. Mylopoulos, "Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling," in *RE*. IEEE Computer Society, 2010, pp. 115–124.

[19] P.-Y. Schobbens, P. Heymans, and J.-C. Trigaux, "Feature diagrams: A survey and a formal semantics," in *RE*. IEEE Computer Society, 2006, pp. 136–145.

[20] J. Coplien, D. Hoffman, and D. M. Weiss, "Commonality and variability in software engineering," *IEEE Software*, vol. 15, no. 6, pp. 37–45, 1998.