

Combined Propagation-Based Reasoning with Goal and Feature Models

Yanji Liu, Yukun Su, Xinshang Yin, Gunter Mussbacher

ECE, McGill University

Montréal, Canada

{yanji.liu | yu.k.su | xinshang.yin}@mail.mcgill.ca, gunter.mussbacher@mcgill.ca

Abstract— The User Requirements Notation (URN) is an international requirements engineering standard published by the International Telecommunication Union. URN supports goal-oriented and scenario-based modeling as well as analysis. Feature modeling, on the other hand, is a well-establishing technique for capturing commonalities and variabilities of Software Product Lines. When combined with URN, it is possible to reason about the impact of feature configurations on stakeholder goals and system qualities, thus helping to identify the most appropriate features for a stakeholder. Combined reasoning of goal and feature models is also fundamental to Concern-Driven Development, where concerns are composed not only based on functionality expressed with feature models, but also based on impact on stakeholder goals. Therefore, an analysis technique for feature and goal models based on a single conceptual model is desirable, because of its potential to streamline model analysis and reduce the complexity of the analysis framework. This paper introduces such a technique, i.e., a single, propagation-based reasoning algorithm that supports combined reasoning of goal and feature models and offers additional usability improvements over existing goal-oriented reasoning mechanisms.

Index Terms—goal modeling, feature modeling, GRL, Goal-oriented Requirement Language, URN, User Requirements Notation, trade-off analysis, evaluation algorithm, concern, concern-driven development, jUCMNav.

I. INTRODUCTION

The User Requirements Notation (URN) is an international requirements engineering standard published by the International Telecommunication Union in the Z.15x series [14]. URN supports requirements engineering activities from elicitation to specification as well as analysis to validation. URN contains two, integrated modeling notations, the Goal-oriented Requirement Language (GRL) for modeling and analysis of goals and the Use Case Map (UCM) notation for modeling and analysis of scenarios. For more than a decade [2], URN has proven and continues to be useful and successful for many requirements engineering activities, in academia and industry. The free, open-source, Eclipse-based jUCMNav tool has contributed significantly to this success, as it allows URN models to be defined, navigated, analyzed, and transformed [15].

The analysis capabilities of URN cover scenario-based analysis as well as goal-based analysis. URN scenario-based analysis enables the definition of a test suite for URN scenario models. The focus of this paper, however, is on support for goal-based analysis with URN. The URN standard describes several non-normative goal model evaluation mechanisms which are implemented in the jUCMNav tool. These mechanisms help with trade-off analysis of several candidate solutions by highlighting their advantages and disadvantages for various stakeholders. All of these mechanisms are propagation-based, i.e., the analysis typically proceeds from the leaf nodes and propagates analysis values to the top-level nodes in the goal model. The propagation is either based on qualitative or quantitative values. The former are more appropriate for early analysis where detailed characteristics of candidate solutions may not yet be known, whereas the latter require a good understanding of the candidate solutions but may lead to more accurate analysis results.

Recently, it was suggested to combine goal-oriented and feature-based modeling for more sophisticated reasoning [17] in the context of Software Product Lines (SPLs) [19] and Concern-Driven Development (CDD) [1]. SPLs are a structured reuse approach and have traditionally used feature models to identify the commonalities and variabilities of a family of similar products. In this context, feature models are typically analyzed to determine whether a selection of features (i.e., a feature configuration) is valid with respect to the constraints captured in the feature model definition. When feature models are combined with goal models, it is possible to also reason about the impact of feature configurations on stakeholder goals and system qualities, thus helping to identify the most appropriate features for a stakeholder in addition to confirming the validity of feature configurations.

The objective of CDD is to provide next generation reuse technologies, through a three-part interface describing units of reuse, i.e., concerns. The variation interface describes required decisions in the solution space and their impact on high level system qualities with the help of feature and goal models. The customization interface allows the chosen variation to be adapted to a specific reuse context, while the usage interface

defines how a customized concern may eventually be used. The first stage of concern composition involves the variation interface. The purpose of this stage is to find the feature configuration of a concern that best addresses the forces at play in the design space. It, therefore, requires an assessment of whether the desired feature configuration of the concern is valid, but also whether those features achieve the design goals, i.e., the features impact the design goals as needed.


As can be seen from these two examples, a coordinated analysis of features and goals is desirable. Our motivation is to harness existing modeling and analysis capabilities of goal models by considering feature models as goal models. This is contrary to adding limited quality/goal information to feature models (typically in the form of feature attributes [8]), and then analyzing this additional information with formal techniques such as propositional logic, constraint satisfaction solving, or description logic [8]. Goal modeling techniques allow quality information and the relationships among qualities to be captured in a richer model [14], which this work intends to exploit through three contributions to the goal and feature modeling communities: (Contribution A) the definition of feature models based on goal model semantics, (Contribution B) simultaneous analysis of feature and goal models through a single, shared evaluation algorithm, and (Contribution C) improved usability of existing propagation-based analysis techniques through the ability to identify invalid feature configurations and automatic selection of mandatory features. Contributions A and B essentially establish feature models as a special case of more general goal models. The shared evaluation algorithm is simpler to maintain compared to two individual analysis mechanisms, thus reducing the complexity of the analysis framework. Contribution C, on the other hand, supports the user when the new evaluation algorithm is applied.








The remainder of this paper gives a background overview of URN goal model analysis techniques in Section II. Section III describes how goal modeling concepts and feature modeling concepts are consolidated into one metamodel based on the semantics of URN goal models. Section IV then discusses the changes to the existing GRL evaluation mechanism, required for the integration of feature model analysis. Section V gives a brief overview of related work, while Section VI concludes the paper and presents future work.

II. URN GOAL MODEL ANALYSIS TECHNIQUES

The Goal-oriented Requirement Language (GRL) is a visual modeling notation for the specification of intentions, business goals, and nonfunctional requirements (NFRs) of multiple stakeholders as well as system qualities. GRL is part of the User Requirements Notation (URN), an international requirements engineering standard published by the International Telecommunication Union [14].

A. Goal Modeling Basics

A GRL *actor* () represents a stakeholder of a system or the system itself. When representing stakeholders, actors are holders of intentions; they are the active entities in the system or its environment who want goals to be achieved, tasks to be performed, softgoals to be satisfied, and resources to be

available. A goal model is a connected graph of *intentional elements* (softgoal, goal, task, resource) that optionally reside within an actor. A goal model shows the high-level business goals and qualities of interest to a stakeholder and the solutions considered for achieving these high-level elements. Goals and qualities are modeled with GRL softgoals and GRL goals. *Softgoals* () differentiate themselves from *goals* () in that there is no clear, objective measure of satisfaction for a softgoal whereas a goal is quantifiable. *Tasks* () represent solutions to goals or softgoals that are considered for a system. In order to be achieved or completed, softgoals, goals, and tasks may require *resources* () to be available. Various kinds of *links* connect the elements in a goal graph. AND, XOR, and IOR *decomposition links* () allow an element to be decomposed into sub-elements. *Contribution links* () indicate desired impacts of one element on another element, either expressed qualitatively with labels (+ or -) or quantitatively as an integer value between -100 and 100. Finally, *dependency links* () model relationships between actors, i.e., one actor depending on another actor for something.

GRL is based on *i** [23] and the NFR Framework [10], but allows intentional elements and links to be more freely combined than *i**. Furthermore, *i**'s notion of agent is replaced with the more general notion of actor/stakeholder/system, and a task does not necessarily have to be an activity performed by an actor, but may also describe properties of a solution.

B. Evaluation of Goal Models

Building on the NFR Framework, GRL supports reasoning about goals and requirements, especially NFRs and quality attributes, through its evaluation mechanism. GRL shows the impact of often conflicting goals and various proposed candidate solutions to achieve the goals. A GRL *strategy* describes a particular candidate solution by assigning initial qualitative or quantitative satisfaction values to a set of intentional elements in the model, typically leaf nodes in the goal model. Satisfaction values can range from -100 to 100. The *evaluation mechanism* propagates the initial satisfaction values of a strategy to those of higher-level stakeholder goals, NFRs, and qualities. Strategies can therefore be compared with each other to help reach the most appropriate trade-offs among often conflicting goals of stakeholders. Note that an evaluation result of a strategy never describes an absolute assessment, but always has to be considered relative to the results of other strategies. The URN standard provides non-normative examples of evaluation algorithms. Three evaluation algorithms applicable to GRL – a quantitative, a qualitative, and a hybrid approach – are described in more detail in [3] and have been implemented in the jUCMNav tool.

All three evaluation algorithms follow the same high-level structure. Each node in the goal model has a number of incoming links and a node is ready to be evaluated if all incoming links are from already evaluated nodes. The two major steps of the evaluation algorithms are:

1. **Step 1:** Add leaf nodes (no incoming links!) and nodes with user-defined initial satisfaction values to the list of elements that are ready to be evaluated.

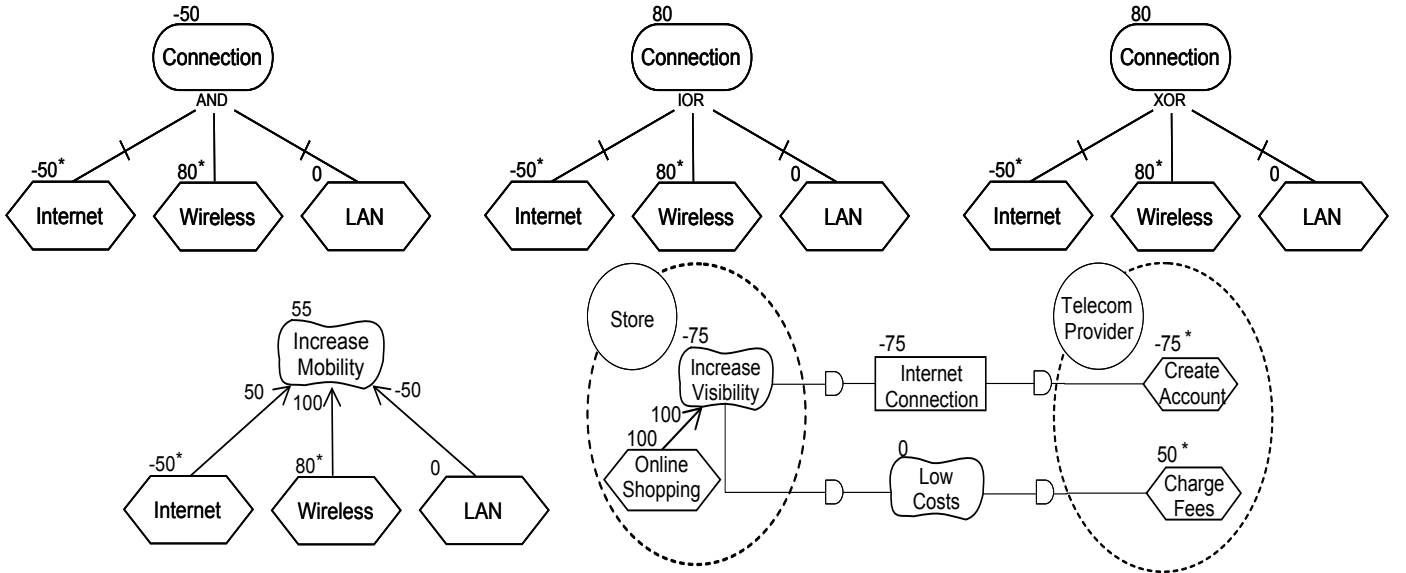


Fig. 1. Quantitative Evaluation of Decomposition (top), Contribution (bottom left) and Dependency (bottom right) Links (adapted from [3])

2. **Step 2:** As long as there is at least a node ready to be evaluated, get the next node N that is ready. Determine the satisfaction value of N and check whether any other nodes connected to N are now ready to be evaluated. If yes, add these other nodes to the ready list.

This evaluation mechanism calculates satisfaction values for all existing nodes, unless circular dependencies exist in the goal model, in which case some nodes may not be evaluated (this is a general constraint of a propagation-based approach).

In Step 2, the satisfaction value of a node is determined based on incoming links and user-defined satisfaction values. The following examples summarize the quantitative approach for the determination of satisfaction values. For the qualitative or hybrid approach, the interested reader is referred to [3].

If a user-defined value exists, the satisfaction value is immediately determined by it. In other words, a user-defined value always overrides any calculated value. If a user-defined value does not exist, then the incoming links are examined. First, decomposition links are considered, then contribution links, and finally dependency links (see Fig. 1; user-defined values are indicated with *).

1. **Decomposition links:** for AND decomposition, the minimum of the satisfaction values of the child elements is taken (e.g., -50 is less than 80 and 0 ; see top left in Fig. 1), while for IOR and XOR the maximum is taken (e.g., 80 is more than 0 and -50 ; see top middle and right in Fig. 1).
2. **Contribution links:** the weighted sum of the child satisfaction values is taken (e.g., $[(-50 \times 50) + (80 \times 100) + (0 \times -50)] / 100 = 55$; see bottom left in Fig. 1), and the value from the decomposition links is added. The minimum result is limited to -100 and the maximum result is limited to 100 , as the result cannot be outside the range of $[-100, 100]$ for GRL satisfaction values.

3. **Dependency links:** dependency links constitute the maximum, allowed satisfaction value, i.e., a parent can never have a higher satisfaction value than a child connected with a dependency link. Hence, the minimum is taken from the result of the contribution links and the satisfaction values of the child elements (e.g., -75 is less than 0 and 100 (the result of the contribution)); see bottom right in Fig. 1).

III. COMBINING FEATURE MODELS AND URN GOAL MODELS

A. Feature Models

Feature models [16] describe the problem space of a family of software systems, i.e., a Software Product Line (SPL) [19]. A *feature model* captures the potential features of members of an SPL in a tree structure, containing features common to all family members and those that vary from one member to the next. A feature model configuration [11] is a selection of desired features defining a particular family member. A node in a feature model represents a *feature* of the SPL. A set of inter-feature relationships allows specifying (i) mandatory and optional parent-child feature relationships, (ii) alternative (XOR) feature groups, and (iii) OR (IOR) feature groups.

A *mandatory* parent-child relationship specifies that the child is included in a feature model configuration if the parent is included. In an *optional* parent-child relationship, the child does not have to be included if the parent is included. Exactly one feature must be selected in an *alternative (XOR) feature group* if its parent feature is selected, while at least one feature must be selected in an *OR (IOR) feature group* if its parent feature is selected.

Often, *includes* and *excludes* cross-tree integrity constraints are also specified, which cannot be captured with the tree structure of the feature model alone. An *includes* constraint ensures that one feature is included if another one is. An *excludes* constraint, on the other hand, specifies that one feature must not be selected if another one is.

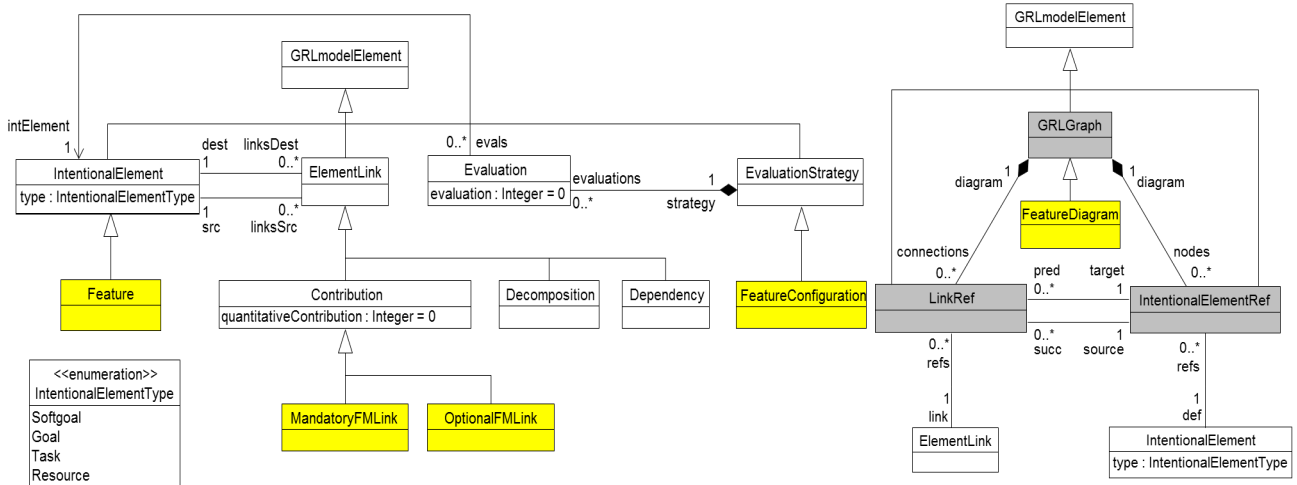


Fig. 2. Extract of GRL Metamodel Extended with Feature Model Concepts Highlighted in Yellow (Decomposition covers XOR and IOR feature groups; external OCL constraints cover cross-tree integrity constraints)

Feature models can be augmented with feature attributes, which are often used to capture qualities and non-functional requirements [8]. However, contrary to goal models, complex relationships among such attributes are usually not expressed.

B. Extending the GRL Metamodel

To support combined modeling and analysis of feature and goal models, the metamodel for GRL is extended with feature modeling concepts by subclassing existing GRL metaclasses (see Fig. 2). This only makes sense if the concepts for feature models are close to the concepts for goal models and the analysis performed on these models is also similar.

The analysis of these two models is indeed similar, because in both cases the required analysis is propagation-driven, i.e., the satisfaction value of a parent element in a goal model depends on and is derived from the satisfaction values of its children. Similarly in feature models, the selection statuses of a parent feature and its child features are dependent on each other (e.g., if the only child of a parent is selected in a valid feature configuration, then the parent must also be selected).

The following feature model concepts need to be defined to fully describe and analyze feature models: (i) feature model, (ii) feature diagram (the concrete representation of a feature model), (iii) feature, (iv) each type of link among features as well as cross-tree integrity constraints, and (v) feature configuration.

C. Feature Model and Diagram

A feature model is an and/or tree with additional link types among features, while a goal model is an and/or graph with additional link types among intentional elements. The overall structure of a feature model is hence a subset of a goal model, if correspondences can be established between (i) features and intentional elements on the one hand and (ii) links in the feature model and links in the goal model on the other hand.

D. Feature

Features are conceptually close to GRL tasks. Both represent either behavioral aspects or properties of a product

family/system. Therefore, a feature (\diamond , e.g., Energy Management in Fig. 3) is a subclass of GRL intentional elements (i.e., the node definitions in a GRL model). The GRL metamodel differentiates various types of intentional elements with the enumeration class *IntentionalElementType*. To clearly separate feature concepts from GRL concepts by consistently using subclassing of existing GRL metaclasses for feature concepts, the *Feature* class is added as a subclass of *IntentionalElement* instead of introducing a new enumeration value for features. Given the similarity of features and tasks, the inherited enumeration value for a feature is still *Task*. Consequently, features are also visualized with hexagons as are tasks, because rectangles (commonly used for features) represent resources, which are a very different concept in GRL. The visualization as hexagons is particularly important when features are used in goal models to describe the impact of features on stakeholder goals. In those situations, a feature takes on the role of a task in regular goal models and hence should be visually identified as such.

E. Links and Cross-Tree Integrity Constraints

Since feature models are an and/or tree and goal models are an and/or graph, XOR feature groups (e.g., the children of Door Identification) and IOR feature groups (e.g., the children of Light Control) can directly be modeled with existing XOR and IOR decomposition links in GRL, respectively. Because these concepts are exactly the same, no new subclasses are introduced for them. In feature models, an XOR feature group or an IOR feature group cannot be combined with any other type of link among features. This restriction does not apply to goal models. Hence, additional constraints ensure the well-formedness of feature models.

Optional links ($\text{---}\circ$, e.g., between Energy Management and Air Conditioning Control) cannot be modeled with decomposition links, because decomposition links require at least one child element in a decomposition group to be selected, whereas it is possible to select none of the children in a group of optional links. Mandatory links ($\text{---}\bullet$, e.g., between Energy Management and Light Control), on the other hand,

could be modeled with AND decomposition links (e.g., see children of EHOME), but only for a group of children with solely mandatory links and no optional links. However, optional and mandatory links can be freely combined into a single group in a feature model, which makes AND decomposition links difficult to use for mandatory links in that case. Dependency links are out of question, because the semantics of dependencies are different than the semantics of optional and mandatory links. Therefore, contribution links are the only choice for optional and mandatory links, and the following paragraphs describe an elegant way of modeling mandatory and optional links with contribution links, assuming that a selected feature has a satisfaction value of 100, while 0 indicates that the feature is not selected. Three cases must be handled: (Case 1) a set of only mandatory links, (Case 2) a set of only optional links, and (Case 3) a set of mixed mandatory and optional links.

1. **Mandatory links only (Case 1):** in this case all children need to be selected for the parent to be selected, too. Therefore, each mandatory link is represented by a contribution link with a contribution value that ensures the parent reaches 100 (and therefore is selected) only if all children are also selected. Given the formula for contribution links stated in Section II, a selected parent hence has to satisfy $[(a \times 100) + (b \times 100) + \dots + (n \times 100)] / 100 = 100$ with a, b, \dots, n being the contribution values of the mandatory links. Consequently, $a + b + \dots + n = 100$, i.e., the sum of all contribution values of all mandatory children equals 100. Therefore, the total of 100 is distributed evenly over all mandatory links barring rounding errors (e.g., see 33, 33, and 34 for the mandatory children of Safety Monitoring).
2. **Optional links only (Case 2):** in this case a single, selected child causes the parent to be selected, too. Therefore, each optional link is represented by a contribution link with a contribution value of 100 (e.g., see the optional children of Air Conditioning Control). As soon as one child is selected the parent is also selected: $(100 \times 100 + 100 \times 0 + \dots + 100 \times 0) / 100 = 100$. If more than one child is selected, the maximum result of contributions links comes into play and the result for the parent is therefore still 100, i.e., selected. If none of the children is selected, it is still possible to select the parent feature, because GRL allows user-defined satisfaction values for any goal model element. Therefore, it is possible to select the parent of only optional children without selecting any of the children, and the satisfaction values will still be propagated correctly upwards from the parent in the feature model.
3. **Mixed mandatory and optional links (Case 3):** in this case the selection of an optional child does not have an impact on the selection status of the parent, because only if all mandatory children are selected will the parent be selected, too. Therefore, the contribution values of optional links is 0, while the mandatory links are handled as in the case of only mandatory links (e.g., see the mandatory and optional children of Energy

Management; there is only one mandatory link in this group and its contribution value is therefore 100 – if there were another mandatory link, the contribution values of both mandatory links will be 50, while the optional link will remain at 0).

Consequently, optional and mandatory links can be mapped to GRL contribution links in a way that preserves the semantics of feature selections through the common GRL propagation mechanism for quantitative satisfaction values. Therefore, optional links (`OptionalFMLink`) and mandatory links (`MandatoryFMLink`) are subclasses of the `Contribution` class.

The *includes* and *excludes* cross-tree integrity constraints can be defined with OCL constraints as demonstrated in [17]. An OCL-based solution for integrity constraints is not just confined to the standard *includes* and *excludes* constraints. It is possible to define more elaborate integrity constraints, such as “exactly two out of a feature group” need to be selected, which otherwise would have to be expressed by more complex tree structures. Note that OCL constraints have been supported by jUCMNav since version 3.1. If an OCL constraint is violated, an error is shown in the Problems view of the jUCMNav tool.

F. A Special Case of Goal Models

Now, as correspondences between features and tasks as well as links in a feature model and GRL decomposition and contribution links have been established, a feature model is indeed a special case of a goal model. In GRL, a goal model is defined by the set of `IntentionalElements` and `ElementLinks` that comprise the goal model as shown on the left side of Fig. 2. Similarly, a feature model is defined by the set of `Features` and feature model-specific `ElementLinks`. A feature model does not contain any goal model-specific elements. On the other hand, a goal model may include features to capture the impact of features on goal model elements such as softgoals, but does not contain mandatory or optional links.

In addition to the already mentioned well-formedness constraints, the whole feature model may only contain one root feature, i.e., a feature that does not have any outgoing links to another feature (see EHOME in Fig. 3), and a feature model is a tree and not a graph.

GRL also has metaclasses for the concrete syntax of URN goal models as shown with the shaded classes `GRLGraph`, `IntentionalElementRef`, and `LinkRef` on the right side of Fig. 2. A `GRLGraph` is a view of the goal model and contains references to `IntentionalElements` and `ElementLinks`. A feature diagram is also a view of the feature model (i.e., the `FeatureDiagram` class is a subclass of the `GRLGraph` class). One further well-formedness constraint exists for feature diagrams. Each feature diagram may only contain one local root feature, i.e., one feature that does not have any outgoing links to another feature on the same feature diagram.

G. Feature Configuration

A `FeatureConfiguration` is a subclass of the `GRLEvaluationStrategy` class. Both allow a set of model elements to be selected for analysis purposes. This is done by assigning initial satisfaction values (i.e., `Evaluations`) to model elements.

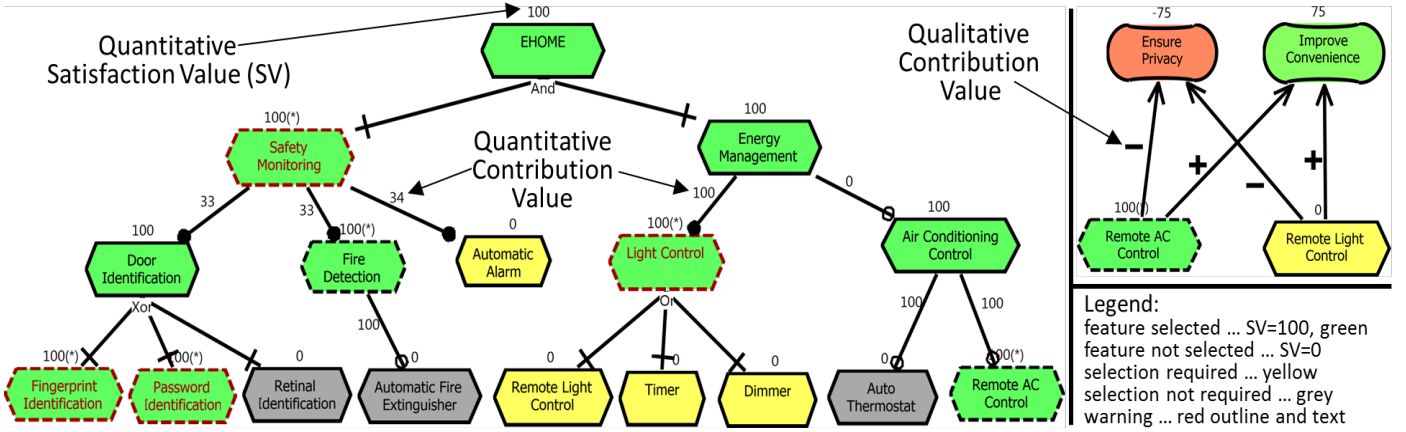


Fig. 3. Feature Modeling and Analysis Based on Adapted URN Propagation Algorithm

In GRL, an initial satisfaction value may range between -100 (element is not satisfied at all) and 100 (fully satisfied), while in the context of feature models well-formedness constraints limit initial satisfaction values to 0 (i.e., not selected by the feature configuration) and 100 (i.e., selected by the feature configuration). Note that all well-formedness constraints stated in this section can be enforced in the jUCMNav tool. The initial satisfaction values of the feature configuration in Fig. 3 contains all elements identified with the (*) marker, i.e., Safety Monitoring, Fire Detection, Fingerprint Identification, Password Identification, Light Control, and Remote AC Control. The satisfaction values of the remaining elements are determined by the evaluation algorithm.

IV. INTEGRATION OF FEATURE MODEL ANALYSIS WITH GOAL MODEL ANALYSIS

A. Basic Feature Model Analysis

When the extended URN model is analyzed, its feature and goal models are evaluated together as one unit. Only three small adaptations are required to the 2-step evaluation algorithm described in Section II to support basic feature model analysis, i.e., the ability to select a set of features and then determine whether this feature configuration is valid. A prerequisite of a valid feature configuration is when the usual propagation of initial satisfaction values from leaf nodes to high-level nodes yields a satisfaction value of 100 for the root node of the feature model. If the root node does not evaluate to 100 , then the feature configuration is certainly invalid.

The selection of features is already handled by the existing analysis framework, except that initial satisfaction values for features need to be constrained to two values: 0 or 100 . Furthermore, it is not useful to directly select the root feature, because the satisfaction value of the root feature should always be calculated to reflect the status of the currently assessed feature configuration.

Since the existing analysis framework already allows the definition of several strategies (and hence feature configurations), it is possible to compare various feature configurations in terms of their validity but also with regards to their impact on goal model elements.

B. First Adaptation

Adaptation #1 ensures that the propagation algorithm takes into account the semantics of mandatory and optional links as described in the previous section. Therefore in Step 2 of the evaluation algorithm, when the satisfaction value of a feature is determined based on incoming links and user-defined satisfaction values, the contribution values of incoming mandatory and optional links are adjusted as defined for Cases 1–3 in Section III.E before the satisfaction value of the feature is determined. Therefore, a modeler never has to specify the contribution values of mandatory and optional links as the adapted propagation algorithm calculates them automatically. With this single change, the adapted propagation algorithm calculates the correct satisfaction values for all features in the feature model as shown, for example, in Fig. 3.

C. Second Adaptation

The results of the propagation algorithm are reported back to the user with the color green to visualize nodes with a satisfaction value of 100 and yellow to visualize those with a satisfaction value of 0 . Nodes with user-defined values are shown with a black dashed outline and the marker (*) next to the satisfaction value. A user-defined value of a node is highlighted in such a way, because it may contradict the satisfaction value derived from the children of the node. The visualization of the results of the propagation algorithm, however, does not indicate whether a contradiction actually exists, the visualization only indicates the possibility because of the user-defined value. This is not sufficient for the analysis of feature models, because without the ability to indicate an actual contradiction (by highlighting features with a red dashed outline and red text as shown in Fig. 3), a modeler may miss problem areas that may arise even if the root feature evaluates to 100 . Furthermore, the visualization does not show gray nodes in the situations shown in Fig. 3. These changes to the visualization of a feature model require adaptations #2 and #3.

Adaptation #2 affects Step 2 of the standard propagation algorithm, which currently does not always calculate the result based on incoming links, e.g., when a user-defined value exists. With adaptation #2, the calculation is always performed, so that

it can be compared against the user-defined value and a warning can be given if needed.

D. Third Adaptation

There are three types of warnings that are indicated with a red dashed outline and red text: (Case A) exactly one feature of an XOR feature group is not selected when the parent of the group is selected (see alternative features Fingerprint Identification and Password Identification in Fig. 3), (Case B) at least one feature of an IOR feature group is not selected when the parent of the group is selected (see the IOR feature group of Light Control), and (Case C) not all mandatory children of a selected parent are selected (see the mandatory children of Safety Monitoring). Cases B and C can be identified the same way, because in those cases the user-defined satisfaction value of a parent feature is not the same as the satisfaction value calculated based on the incoming links. All of these problem areas are caused by user-defined selections. In Case A, the user selected two features in an XOR group. In Case B, the user selected the parent of an IOR feature group without selecting any of its children. In Case C, the user selected the parent of mandatory children without selecting all mandatory children.

In Step 1 of the standard propagation algorithm elements are collected that are already ready to be evaluated at the start of the evaluation. Recall that leaf nodes and user-defined nodes are ready at the start. However, Step 1 does not respect any order among these elements. Therefore, it may happen that a user-defined value from the middle of the feature tree is evaluated before a leaf node. Hence, if an element is evaluated before its child elements, the calculation of the incoming links may not be correct, because the calculation depends on the satisfaction value of a child element, which may not yet have been calculated. This is not an issue in the current evaluation algorithm, because a user-defined value does not need to be compared against the value derived from incoming links.

With adaptation #3, Step 1 keeps separate three groups of elements that are ready for evaluation at the start of the evaluation: (Group 1) global leaf nodes of the combined goal and feature model, (Group 2) leaf nodes of the feature model only, and (Group 3) user-defined nodes. Elements that may belong to more than one of these groups at the same time are only added to one of the groups with Group 1 given priority over Group 2 and Group 2 given priority over Group 3.

During Step 2 of the adapted propagation algorithm, the evaluation starts with group 1 and evaluates all of the elements in this group, and then any other element that is now ready for evaluation because their children have been evaluated. If one of these other elements is from group 2 or 3, it is removed from group 2 or 3, so that it is not evaluated twice. When group 1 is exhausted, the evaluation moves on to the remaining elements in group 2, and eventually to group 3.

Given the well-formedness constraint that a feature model is a tree and the additional constraint that no circular dependencies¹ impact feature model elements, the warnings

can now be reliably displayed, because any user-defined value can be compared to the correct value derived from the incoming links. If the values are different, the element is highlighted with a red dashed outline and red text (i.e., Cases B and C are handled).

There is one exception where a user-defined value is different than the value derived from incoming links, and this is the case for feature Fire Detection in Fig. 3. If the incoming links are all optional and the child features are not selected (i.e., the user-defined value of the parent is 100, but the value of the incoming links is 0), then this is not a problematic situation and does not need to be highlighted.

Case A can now also be handled, because the satisfaction values of features in an XOR group can be compared with each other to identify two (or more) elements that are all selected.

E. Differentiating Feature Selections

Last but not least, the gray color coding in addition to the default yellow color coding helps differentiate two kinds of features that are not selected: a feature that does not need to be selected for the feature configuration to be valid, and a feature that may still have to be selected for the feature configuration to be valid. For example in Fig. 3, Automatic Alarm is shown in yellow, because it still needs to be selected, since it is a mandatory child of the selected Safety Monitoring feature. Similarly, the three child features of the selected Light Control feature are shown in yellow, because at least one of them needs to be selected, so that the feature configuration is valid. On the other hand, Retinal Identification is shown in gray, because it does not need to be selected anymore, since features from the XOR feature group have already been selected (the same rule applies to IOR feature groups, e.g., if one child feature of Light Control is selected, the other two will be shown in gray). Note that the gray color coding does not mean that one is not allowed to select the feature at all. The gray color coding simply means that a valid feature configuration does not need this feature to be selected. The other two features (Automatic Fire Extinguisher and Auto Thermostat) are shown in gray, because each one is an unselected optional child. The last situation where an unselected feature is shown in gray occurs when the parent of a feature is not selected, i.e., any child features of the yellow or gray features in Fig. 3 would be shown in gray.

To summarize, a feature model is valid if the evaluation of its root node is 100, no OCL constraints describing cross-tree integrity constraints of features are violated, and no warning exists in the feature model. The warnings and color coding of features help differentiate five cases:

1. **User-selected feature:** value 100 with (*) marker, green fill color, black dashed outline, black text
2. **Propagation-selected feature:** value 100, green fill color, black solid outline, black text
3. **Unselected feature, but selection may be required:** value 0, yellow fill color, black solid outline, black text
4. **Unselected feature and selection is not required (but selection is still possible):** value 0, gray fill color, black solid outline, black text

¹ This constraint is reasonable as features are typically leaf nodes in the goal model and, if so, cannot be part of a circular dependency (see goal model on right side of Fig. 3).

5. **Warning (applies only to selected features):** value 100 (with (*) marker if applicable), green fill color, red dashed outline, red text

Furthermore, the propagation algorithm may result in a satisfaction value of less than 100 but more than 0 for a feature. Such a feature is not selected, but the value indicates how far away from being properly selected the feature is (the closer the value to 100, the closer it is to being selected). These features are shown with a lighter green fill color that turns more and more into yellow the closer the satisfaction value is to 0.

F. Summary of Adaptations

All of the warnings and color coding use the results of the adapted propagation algorithm to change the visualization of features, but are not technically part of the propagation algorithm. The same applies to the OCL constraint checker, which is provided by a third party and is hence also not part of the propagation algorithm. In summary, the three changes to the propagation algorithm are: (i) considering the semantics of mandatory and optional links, (ii) always calculating the satisfaction value derived from incoming links and comparing it to the user-defined value, and (iii) evaluating the elements in a certain order (first global leaf nodes, then feature model leaf nodes, and then user-defined nodes).

These are rather minor changes that further demonstrate the fundamental similarity of the analysis of feature and goal models. The described, required adaptations to the propagation algorithm are inconsequential for the evaluation of goal models, but make feature model analysis possible and improve its usability.

The advantage of this approach is that (i) the existing analysis framework for evaluation algorithms [3] is readily available for feature models (including color-coded visualization of evaluation results), (ii) features can be used in GRL models to define the impact of features on stakeholder goals and system qualities, thus enabling combined reasoning about feature and goal models for stakeholder trade-offs, and (iii) the full power of goal-oriented modeling and analysis is available to feature models, which is important for more sophisticated reasoning in the context of SPLs and CDD.

The single evaluation algorithm shared between feature and goal models only needs to be adapted three times as stated in

the previous paragraphs, because features and intentional elements, decomposition links and feature groups, feature configuration and strategies, and integrity and OCL constraints are semantically equivalent in terms of their required behavior for the evaluation algorithm.

Furthermore, the whole basic premise of satisfaction values and color-coded feedback as used in goal model analysis is readily applicable to feature model analysis.

The result of a combined feature and goal model evaluation with color-coded feedback is shown in Fig. 3. On the right side, a brief example is shown of a standard GRL goal model, capturing the impact of features on goals (e.g., Remote Light Control and Remote AC Control have a negative impact on Ensure Privacy but a positive impact on Improve Convenience).

While the focus of this paper is on quantitative propagation mechanism, the example goal model in Fig. 3 shows a hybrid approach with qualitative contribution values and quantitative satisfaction values to indicate that qualitative reasoning may also be used instead of quantitative reasoning. In fact, all three types of evaluation mechanisms (quantitative, qualitative, and hybrid) may be used for combined feature/goal model analysis.

The only part of the evaluation mechanism that needs to be adapted to a specific type of evaluation is the propagation of mandatory and optional links. However, the exact specification for the qualitative and hybrid approaches is out of scope for this paper.

G. Automatic Selection of Mandatory Features

Auto-selection of a feature's mandatory children (including AND-decomposed children; see Fig. 4) is not supported by the standard evaluation mechanism. Auto-selection is a top-down approach that starts with the root feature as the first parent node. The approach then marks as auto-selected all mandatory children of the current parent node, if the parent node is selected or has been auto-selected (note that the root feature is deemed selected in this approach, as the end result will always be a selected root feature). The approach continues by considering as parent nodes those children that were marked auto-selected or were already selected. This continues as long as a child feature meets the criteria to be considered as the next parent node.

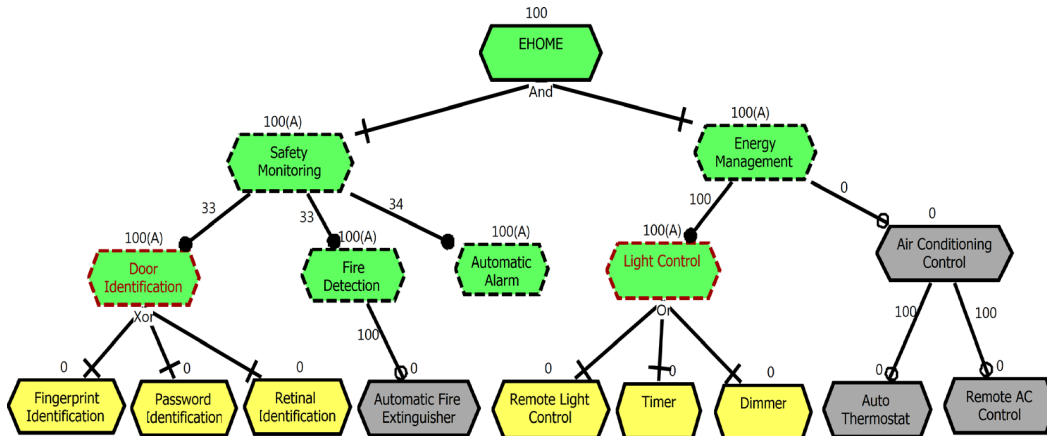


Fig. 4. Automatic Selection of Features

The auto-selection of mandatory children is clearly dependent on which features are already selected in the feature configuration. For example, an optional feature may have a mandatory child. This mandatory child should only be selected automatically, if its optional parent has been selected.

This interplay between user-defined selections and auto-selections is addressed in the combined analysis of feature and goal models by running the evaluation mechanism a first time (which propagates selection values based on the user-defined selections), then running the new auto-selection mechanism (which marks up auto-selected features), and finally running the evaluation mechanism again to update the selection values in the feature model based on the auto-selection markers.

The required change to the standard evaluation algorithm is hence rather straightforward. Besides running the evaluation algorithm twice, the calculation of the satisfaction value in Step 2 needs to take the auto-selection markup into account, when deciding on the satisfaction value. If a markup exists, the satisfaction value is 100 (i.e., selected).

Auto-selection avoids one of the problems discussed in Section IV.D. Case C, the third warning, cannot occur anymore, because all mandatory children that should be selected will be selected automatically by this approach.

Auto-selected features are indicated with an additional (A) marker next to the satisfaction value. The feature model in Fig. 4 shows the result of an evaluation where no user-defined selections were made. The color-coding clearly indicates the next steps of the modeler. All green features are handled, except if warnings are shown (red dashed outline and red text), in which case the modeler needs to make a decision on how to resolve the warning, possibly by considering the yellow features. Furthermore, the modeler may want to select any of the gray features with a selected parent as necessary for the family member of the software product line. In addition, any OCL constraints for cross-tree integrity constraints that were violated also need to be resolved by the modeler.

As a proof of concept, the combined analysis of feature and goal models presented in this paper as well as automatic feature selection have been implemented in jUCMNav 6.0 [15].

V. RELATED WORK

To the best of our knowledge, feature models have not been considered as a specific kind of goal model and then analyzed together with goal models based on goal model semantics, i.e., based on a propagation-based analysis technique. There are several formal approaches [6][7][12][21] to support analysis of feature configurations as well as impact analysis, but they are not propagation-based. Many other approaches for the analysis of feature configurations exist (e.g., using constraint satisfaction solving techniques [13] or business rule systems [18]), but they are not discussed here in detail as they do not explicitly include goal/quality information for impact analysis.

Bagheri et al. [7] use a fuzzy propositional language for a semi-automated, interactive approach to construct feature configurations. This approach takes into consideration the stakeholders' desired quality attributes (called soft constraints)

during feature configurations. The propositional language represents both feature models and their quality attributes. Bagheri et al. [6] also introduce the Stratified Analytic Hierarchy Process (S-AHP) approach, which allows ranking and selecting the main business goals and most relevant features during feature configurations.

Tun et al. [21] present an approach to relate requirements and feature model configurations. Once requirements are selected for a desired product, one or more feature configurations that satisfy the requirements and the quantitative (quality) constraints are generated.

The DiVA project also combines software product lines with analysis of stakeholder goals – in this case, for the purpose of dynamically selecting the most appropriate feature configuration given an application's changing environmental context [12]. DiVA's reasoning framework uses reasoning engines such as an Alloy SAT solver, but does not use a combined, propagation-based approach.

Several other approaches derive/map feature models from/to goal models (Yu et al. [24], António et al. [4], Silva et al. [20], Asadi et al. [5], Uno et al. [22], and Borba and Silva [9]), but again combined, propagation-based analysis according to goal-model semantics is not supported.

Both Alam et al. [1] and Mussbacher et al. [17] argue for combined reasoning of goal and feature models, but do not specify in detail how a combined reasoning algorithm works.

VI. CONCLUSIONS AND FUTURE WORK

This paper demonstrates how feature and goal models can be defined and analyzed with a common conceptual model based on the semantics of the goal model language of the User Requirements Notation (URN). Existing propagation-based analysis techniques for URN goal models only require three small adaptations to be able to offer feature model analysis, i.e., to determine whether a feature configuration for a feature model is valid or not. More advanced feature model functionality that automatically selects mandatory features also requires only minor changes to the existing propagation-based algorithm that amount to a few extra lines of code.

We argue that a single evaluation algorithm that combines these two similar analysis techniques results in a less complex reasoning framework than one with separate analysis algorithms for feature and goal models based on separate language definitions. The single evaluation algorithm treats features and goal model elements the same way, evaluates them simultaneously, and operates on a single model. Separate analysis algorithms, on the other hand, have to duplicate the similar behavior of the analysis techniques in each algorithm. Furthermore, the feature definitions from the feature model have to somehow be shared and kept consistent with the goal model, because the goal model needs to evaluate the impact of features on stakeholder goals and system qualities. Any feature that is added in a goal model would have to be added to the feature model and vice versa. A single model with a single evaluation algorithm, as the one proposed in this paper, avoids these problems.

Combined reasoning over feature and goal models is useful in the context of Software Product Lines, when it is desired to reason about the impact of feature configurations on stakeholder goals and system qualities in addition to reason about the validity of feature configuration. Furthermore, combined reasoning of goal and feature models is fundamental to Concern-Driven Development (CDD), where concerns are composed not only based on functionality expressed with feature models, but also based on impact on stakeholder goals.

In future work, we plan to apply the combined reasoning framework to CDD [1], in particular, to concern hierarchies that result from the incremental composition of reusable concerns at various levels of abstraction. In this context, the reasoning framework aims to provide greater insight on how low level decisions may impact high-level stakeholder goals and how trade-offs of feature configurations at different levels of abstraction may be managed.

REFERENCES

- [1] Alam, O., Kienzle, J., and Mussbacher, G., “Concern-Oriented Software Design”, *Model Driven Engineering Languages and Systems*, Springer, LNCS, vol. 8107, pp. 604–621, 2013. DOI: 10.1007/978-3-642-41533-3_37.
- [2] Amyot, D. and Mussbacher, G., “User Requirements Notation: The First Ten Years, The Next Ten Years”, *Journal of Software (JSW)*, vol. 6(5), pp. 747–768, 2011. DOI: 10.4304/jsw.6.5.747-768.
- [3] Amyot, D., Ghanavati, S., Horkoff, J., Mussbacher, G., Peyton, L., and Yu, E., “Evaluating Goal Models within the Goal-oriented Requirement Language”, *International Journal of Intelligent Systems (IJIS)*, Wiley, vol. 25(8), pp. 841–877, 2010. DOI: 10.1002/int.20433.
- [4] António, S., Araújo, J., and Silva, C., “Adapting the i* Framework for Software Product Lines”, *ER Workshops: Advances in Conceptual Modeling - Challenging Perspectives*, Springer, LNCS, vol. 5833, pp. 286–295, 2009. DOI: 10.1007/978-3-642-04947-7_34.
- [5] Asadi, M., Bagheri, E., Gasevic, D., and Hatala, M., “Goal-Driven Software Product Line Engineering”, *26th Annual ACM Symposium on Applied Computing (SAC’11)*, ACM, pp. 691–698, 2011. DOI: 10.1145/1982185.1982336.
- [6] Bagheri, E., Asadi, M., Gasevic, D., and Soltani, S., “Stratified Analytic Hierarchy Process: Prioritization and Selection of Software Features”, *14th Intl. Software Product Line Conference (SPLC’10)*, South Korea, pp. 300–315, 2010. DOI: 10.1007/978-3-642-15579-6_21.
- [7] Bagheri, E., Noia, T., Ragone, A., and Gasevic, D., “Configuring Software Product Line Feature Models Based on Stakeholders’ Soft and Hard Requirements”, *Software Product Lines: Going Beyond*, Springer, LNCS, vol. 6287, pp. 16–31, 2010. DOI: 10.1007/978-3-642-15579-6_2.
- [8] Benavides, D., Segura, S., and Ruiz-Cortés, A., “Automated analysis of feature models 20 years later: A literature review”, *Information Systems*, Elsevier, vol. 35(6), pp. 615–636, 2010. DOI: 10.1016/j.is.2010.01.001.
- [9] Borba, C., and Silva, C., “A Comparison of Goal-Oriented Approaches to Model SPLs Variability”, *ER Workshops: Advances in Conceptual Modeling - Challenging Perspectives*, Springer, LNCS vol. 5833, pp. 244–253, 2009. DOI: 10.1007/978-3-642-04947-7_34.
- [10] Chung, L., Nixon, B.A., Yu, E., and Mylopoulos, J., “Non-Functional Requirements in Software Engineering”, Kluwer Academic Publishers, 2000.
- [11] Czarnecki, K., Helsen, S., and Eisenecker, U.W.: “Staged configuration through specialization and multilevel configuration of feature models”, *Software Process: Improvement and Practice*, vol. 10(2), pp. 143–169, 2005. DOI: 10.1002/spip.225.
- [12] DiVA project website – DiVA Reasoning Framework, <https://sites.google.com/site/divawebsite/divastudio/diva-reasoning-framework>.
- [13] Felfernig, A., Friedrich, G., Jannach, D., and Zanker, M., “Intelligent Support for Interactive Configuration of Mass-Customized Products”, *Engineering of Intelligent Systems*, Springer, LNCS, vol. 2070, pp. 746–756, 2001. DOI: 10.1007/3-540-45517-5_82.
- [14] International Telecommunication Union, “Recommendation Z.151 (10/12), User Requirements Notation (URN) – Language definition”, 2012. <http://www.itu.int/rec/T-REC-Z.151/en>
- [15] jUCMNav website, version 6.0, University of Ottawa. <http://jucmnav.softwareengineering.ca/jucmnav>
- [16] Kang, K., Cohen, S., Hess, J., Novak, W., and Peterson, S.: “Feature-oriented domain analysis (FODA) feasibility study”, Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, 1990.
- [17] Mussbacher, G., Araújo, J., Moreira, A., and Amyot, D., “AoURN-based Modeling and Analysis of Software Product Lines”, *Software Quality Journal (SQJ)*, Springer, vol. 20(3-4), pp. 645–687, 2011. DOI: 10.1007/s11219-011-9153-8.
- [18] Pillat, R.M., Basso, F.P., Oliveira, T.C., and Werner, C.M.L., “Ensuring consistency of feature-based decisions with a business rule system”, *7th International Workshop on Variability Modelling of Software-intensive Systems (VaMoS’13)*, ACM, pp. 15:1–15:8, 2013. DOI: 10.1145/2430502.2430523.
- [19] Pohl, K., Böckle, G., and van der Linden, F.J., “Software Product Line Engineering: Foundations, Principles and Techniques”, Springer, 2005.
- [20] Silva, C., Alencar, F., Araújo, J., Moreira, A., and Castro, J., “Tailoring an Aspectual Goal-Oriented Approach to Model Features”, *20th Intl. Conference on Software Engineering and Knowledge Engineering (SEKE’08)*, Knowledge Systems Institute Graduate School, pp. 472–477, 2008.
- [21] Than Tun, T., Boucher, Q., Classen, A., Hubaux, A., and Heymans, P., “Relating Requirements and Feature Configurations: A Systematic Approach”, *13th Intl. Software Product Line Conference (SPLC’09)*, Carnegie Mellon University, pp. 201–210, 2009.
- [22] Uno, K., Hayashi, S., and Sacki, M., “Constructing Feature Models Using Goal-Oriented Analysis”, *9th Intl. Conference on Quality Software (QSIC’09)*, IEEE, pp. 412–417, 2009. DOI: 10.1109/QSIC.2009.61.
- [23] Yu, E., “Modeling Strategic Relationships for Process Reengineering”, Ph.D. thesis, Department of Computer Science, University of Toronto, Canada, 1995.
- [24] Yu, Y., Leite, J., Lapouchnian, A., and Mylopoulos, J., “Configuring Features with Stakeholder Goals”, *23rd Annual ACM Symposium on Applied Computing (SAC’08)*, ACM, pp. 645–649, 2008. DOI: 10.1145/1363686.1363840.