Softw Syst Model (2016) 15:257–279
DOI 10.1007/s10270-014-0402-8

REGULAR PAPER

Goal-oriented modeling and verification of feature-oriented product lines

Mohsen Asadi · Gerd Gröner · Bardia Mohabbati · Dragan Gašević

Received: 27 August 2012 / Revised: 16 September 2013 / Accepted: 3 February 2014 / Published online: 22 February 2014

Abstract Goal models represent requirements and intentions of a software system. They play an important role in the development life cycle of software product lines (SPLs). In the domain engineering phase, goal models guide the development of variability in SPLs by providing the rationale for the variability, while they are used for the configuration of SPLs in the application engineering phase. However, variability in SPLs, which is represented by feature models, usually has design and implementation-induced constraints. When those constraints are not aligned with variability in goal models, the configuration with goal models becomes error prone. To remedy this problem, we propose a description logic (DL)-based approach to represent both models and their relations in a common DL knowledge base. Moreover, we apply reasoning to detect inconsistencies in the variability of goal and feature models. A formal proof is provided to demonstrate the correctness of the reasoning approach. An empirical evaluation shows computational tractability of the inconsistency detection.

Keywords Software engineering · Feature oriented software families · Goal-oriented requirements engineering · Description Logic · Feature Models · Verification

Communicated by Dr. Benoit Baudry.

M. Asadi (✉) · B. Mohabbati · D. Gašević
Simon Fraser University, Burnaby, Canada
e-mail: masadi@sfu.ca

G. Gröner
paluno-The Ruhr Institute for Software Technology, University of Duisburg-Essen, Duisburg, Germany

D. Gašević
Athabasca University, Athabasca, Canada

1 Introduction

One of the most prominent paradigms for reuse in software engineering is software product lines engineering (SPLE). An SPL (also known as a product family) is a set of software systems that share most of their *features*. SPLE consists of the *domain engineering* and *application engineering* life cycles [1]. In the *domain engineering* life cycle, an SPL is developed as a whole. Variability and commonality among SPL members are represented by feature models. In the *application engineering* life cycle, a final application can be derived through the feature model configuration, i.e., the process of selecting and removing features from the feature model based on stakeholders' requirements [2].

Features represent both technical elements (internal features) and non-technical elements (external features) of a software system [3]. That is, not only does a feature model encompass the visible characteristics of product lines, but it also includes many design and implementation features along with the variability relations between those features. Moreover, there are complex mapping relations between features and stakeholders' objectives, such that a goal can be realized by several features and a feature can be used for the realization of several goals. Considering the technical aspects of feature models, the large size of feature models, and the complex mapping relations between features and stakeholders intentions, it is not easy for a stakeholder to understand the functional and non-functional aspects of features and to select features based on their objectives. Additionally, the features in the feature model are of different interest for stakeholders involved in the project [4,5]. For instance, final stakeholders are interested to the user visible features while designer and programmer require to see detail and technical features.

These challenges urge techniques which can handle the complexity of feature models and facilitate the selection of features during the configuration process. Clarke and Proenca [4] and Acher et al. [5] emphasized the importance of modularity in managing complexity by using views to show a feature model up to certain levels of detail to stakeholders. To this end, we aim at providing a more stakeholder-oriented view of feature models where goal-oriented requirements engineering (GORE) [6] is employed to generate a stakeholder view of feature models.

GORE makes extensive use of goal models, i.e., models capturing user intentions for a system-to-be, and facilitates the exploration of design alternatives, described in high-level non-technical terms. Typically, goal models are also used by the SPLE community in both life cycles of SPLE. In domain engineering, they are applied for a top down development of SPLs [7–9]. In application engineering, they ensure the selection of features that are based on the objectives of a target application stakeholder [10].

In essence, goal models and feature models provide different variability perspectives. Goal models represent intentional variability, which is different in objectives of stakeholders and the way stakeholders may use a system-to-be to reach their objectives [7]. On the other hand, feature models are commonly used to illustrate variability between various systems, which is called product line variability [11]. When applying goal models in the context of SPLE, we need to capture and represent the stakeholders' objectives of several products. Hence, not only should the goal models be able to represent the objectives of the stakeholders of various products, but they also should be able to distinguish between objectives of different products. In other words, the goal model should illustrate product line variability in the intentional space, which refers to differences in intentional spaces of product line members. Standard goal model languages such as i* and GRL can represent intentional variability, but lack mechanisms for representing differences between intentional spaces of various systems (i.e., product line variability in the intentional space). Hence, we introduced the notion of *family goal model* by extending standard goal modeling techniques (see Sect. 3.1).

The family goal model and the feature model are connected by mappings, which provide bidirectional relationships and traceability links between high-level business objectives of stakeholders, described by goal models and implementation units encapsulated within features in feature models. In this paper, we refer to the combination of the family goal model, the feature model, and the mapping model as a *family requirements model*. Due to the different perspectives of family goal models and feature models, the variability semantics might be different in both models. Also, these models might be developed by different stakeholders. Moreover, due to technical constraints, relations in feature models may be changed by software designers during the development of a product line. All these factors may lead to inconsistency between family goal model relations and feature model relations, which limit the product configuration within the application engineering life cycle since several feature selections could lead to non-satisfaction of stakeholders' intentions.

To remedy these problems, we present a validation approach that can detect inconsistencies in variability between goal and feature models already in the *domain engineering* life cycle, i.e., based on mappings between goals and features, independent of a particular feature selection. We establish a family requirements model, and we provide an approach for the validation of the family requirements model. The approach does not only ensure the alignment of feature selections to user intentions but also prevent the high cost of changing design and implementation models in last phases of domain engineering to align them with variability in intentions of stakeholders.

In particular, this paper makes the following contributions: (1) a goal model profile extension (i.e., family goal model) to represent intentions of product lines and the difference of intentions for various products; (2) a representation and formal definition of family goal models, feature models, and their mappings in description logic (DL); (3) a logic-based reasoning through standard reasoning mechanisms for the discovering of inconsistencies in a family requirements model; (4) support for maintainability and traceability between goal and feature models; (5) formal and empirical evaluation of the proposed approach.

The paper is structured as follows: Sect. 2 introduces the goal and feature model notations and provides formal definitions for these models. Section 3 explains the role of goal models in the software product line domain and introduces the notion of family goal models. The family requirements model and a set of inconsistency patterns, which may happen in the family requirements model, are explained in Sect. 4. After providing a representation of the family requirements model in description logic in Sect. 5, the validation procedure for identifying inconsistencies in family requirements model is explained in Sect. 6. After describing the evaluation of the inconsistency checking algorithms in Sect. 7 and discussing related work in Sect. 8, concluding remarks and directions of the future work are highlighted in Sect. 9.

2 Foundations

Goal and feature models constitute the cornerstones in goal-oriented SPLE. This section introduces both model types and clarifies their concepts using a running example from the online shopping domain [12].

2.1 Goal models

Among existing goal model representations such as i*/Tropos [13,14], NFR [15], KAOS [16], and Goal Requirements Language (GRL) [17], we adopt GRL, a part of the User Requirements Notation (URN) [17], due to its wide adoption in industry and research and being an international standard. Similar to related work [18,19], in our framework, a goal model consists of decompositions and contribution links.[1]

Intentional elements (or *intentions*) used in this paper are *(hard) goals*, *soft goals*, and *tasks*. *Goals* represent conditions or states of affairs that a stakeholder might like to achieve. *Soft goals* are similar to hard goals, but without a clear-cut criteria for whether the condition is achieved, and it is up to subjective judgment. Typically, they model non-functional requirements of the target system.[2] *Tasks* specify conceptual solutions in the target system.[3]

A goal can be decomposed into several subgoals, represented by *decomposition relationships*. GRL supports *AND*, *IOR*, and *XOR* decompositions. The satisfaction of a target intentional element in an *AND*-decomposition requires that all source intentional elements need to be satisfied. *IOR* is used to specify that satisfaction of at least one source satisfies the target intentional element. Finally, *XOR* specifies that exactly one of the source elements is necessary to satisfy the target.

GRL also supports contribution links that model the impact of satisfaction of a source intentional element on the satisfaction of a target intentional element. Among the *contributions*, Make and Break are respectively positive and negative, and sufficient for the satisfaction of a target intentional element, while Help and Hurt are also respectively positive and negative, but insufficient for the satisfaction of a target element. The extent of the contribution of SomePositive and SomeNegative is unknown. Finally, for the Unknown contribution, both the extent and degree (positive or negative) are unknown.

A concrete goal model is depicted in Fig. 1, where intentional elements and relations are represented. For instance, the goal Payment Collected can be achieved either by Receive Payment By Card, by Receive Payment By Non-Card, or by Payment Postponed. Also, satisfaction of Payment Postponed requires satisfaction of both intentional elements Trustworthiness of the Customer Determined and Determine Payment Date. Satisfactions of Courier Deliver and Deliver Item lead to satisfaction and dissatisfaction of soft goal Minimize Delivery Cost, respectively.

Definition 1 (*Goal model*) A goal model is a triple $GM = \langle \mathcal{G}, \mathcal{C}, \mathcal{D} \rangle$. $\mathcal{G}$ is a set of goals (also called intentions or intentional elements). Intentions are (hard) goals ($\mathcal{G}_g$), tasks ($\mathcal{G}_t$) and soft goals ($\mathcal{G}_s$). $\mathcal{C}$ and $\mathcal{D}$ describe intentional relations on $\mathcal{G}$, $\mathcal{C}$ denotes positive and negative contributions ($\mathcal{G} \times \{\overset{\bullet}{+}, \overset{\bullet}{+}, +, \pm, \overset{\bullet}{-}, -, \overset{\bullet}{-}\} \times \mathcal{G}$). $\mathcal{D}$ are decomposition relations of intentional elements $\mathcal{G} \times \{IOR, XOR, AND\} \times \mathcal{P}(\mathcal{G})$.

2.2 Feature models

In SPLs, *feature models* represent variability between members of a product line (i.e., product line variability) (cf. Definition 2).

Definition 2 (*Feature model*) A feature model $FM = \langle \mathcal{F}, \mathcal{F}_M, \mathcal{F}_O, \mathcal{F}_{IOR}, \mathcal{F}_{XOR}, \mathcal{F}_{incl}, \mathcal{F}_{excl} \rangle$ is a tree structure that consists of features $\mathcal{F}$ and feature relations in terms of parent–child and integrity constraints. $\mathcal{F}_M \subseteq \mathcal{F} \times \mathcal{P}(\mathcal{F})$ and $\mathcal{F}_O \subseteq \mathcal{F} \times \mathcal{P}(\mathcal{F})$ are sets of parent and the set of all their mandatory and optional child features, respectively. $\mathcal{F}_{IOR}$ and $\mathcal{F}_{XOR} \subseteq \mathcal{F} \times \mathcal{P}(\mathcal{F})$ are sets of pairs of child features and their common parent feature. $\mathcal{F}_{incl}$ and $\mathcal{F}_{excl} \subseteq \mathcal{F} \times \mathcal{F}$ are sets of includes and excludes relationships (integrity constraints).

Figure 2 shows a part of the on-line shopping feature model. *Mandatory* features have to be selected when their parent is selected, e.g., Order Management and its mandatory child features Pay Management, Order Preparation, and Shipment. In contrast, the selection of a parent feature (e.g., Pay Management) does not require the selection of its optional child feature (e.g., Payment Postpone). *Group* relationships are classified into *OR* (*IOR*) and *alternative* (*XOR*). An *includes* integrity constraint is used when a selection of one feature requires the selection of another one, e.g., in Fig. 2, the selection of E-Bill requires the selection of Email, as depicted by the annotation in the upper part of the figure. An *exclude* constraints imply that selection of a feature excludes the selection of the other feature.

3 Goal models in the SPLE life cycle

Goal and feature models are used for different purposes. In order to adopt goal models in the development life cycles of software product lines, we continue with a foundational analysis of the different modeling constructs in both types of models.

[1] In this paper, we are interested in validation of intentions in family goal models, typically achieved within one actor. Therefore, we do not consider different actors and dependencies among them.

[2] Some non-functional requirements, like security, can have a clear-cut criteria and also can be achieved with different operationalizations. The GRL standard does not specify how to model these situations. Nevertheless, the standard allows the specification of decompositions on soft goals.

[3] Tasks are considered to be requirements if they are assigned to the system-to-be and to be assumptions if they are assigned to the environment.

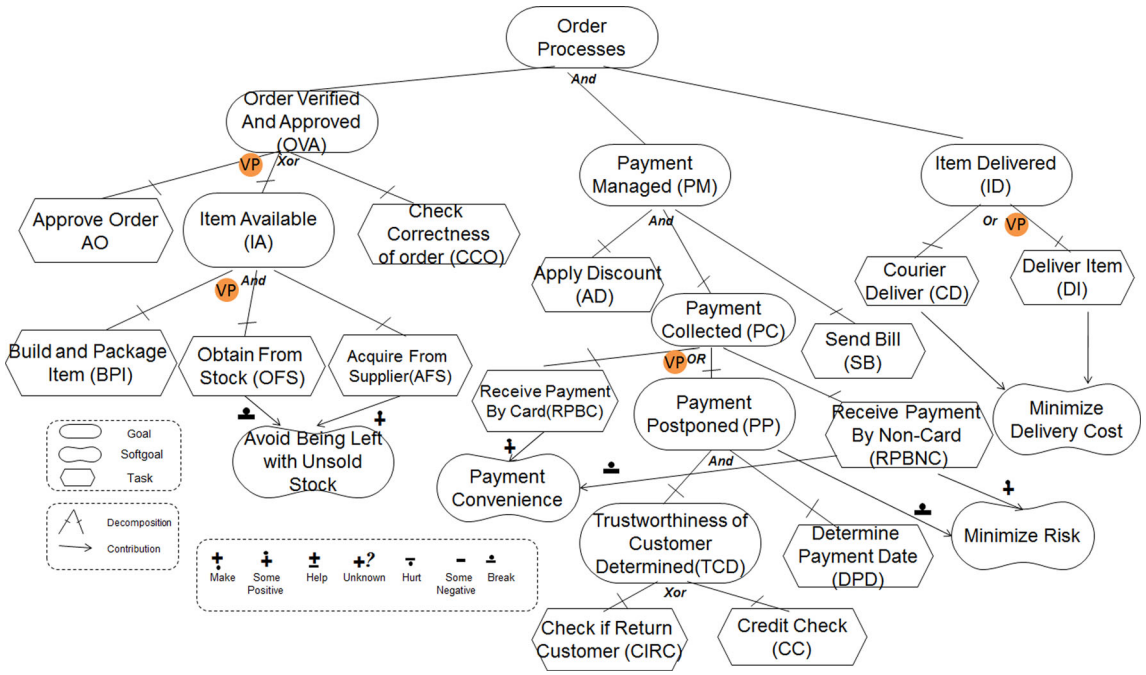

Fig. 1 Goal Model of the e-store case study—*VP* and *circle* notations will be introduced in Sect. 3.1

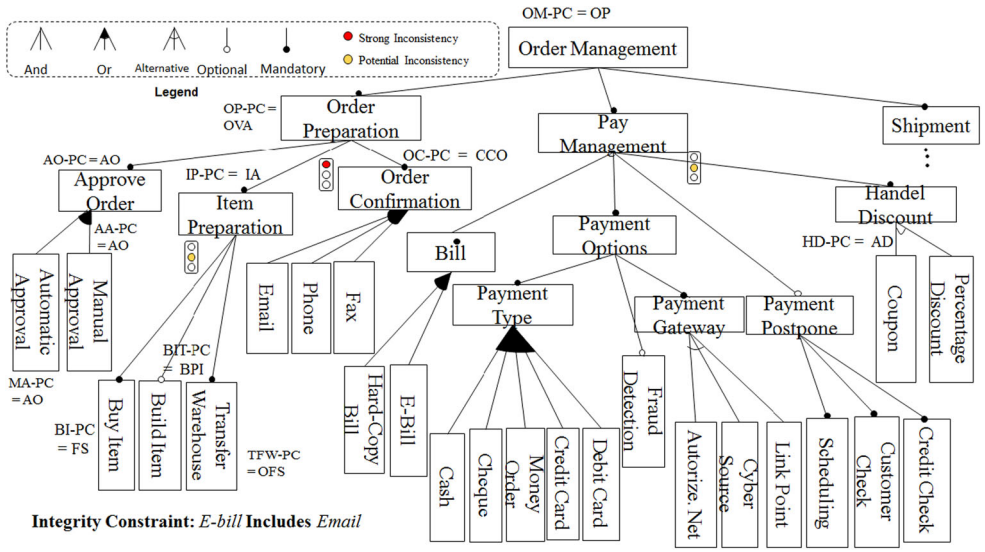

Fig. 2 Feature model of the online shopping case study

3.1 Family goal models and feature models

However, model elements and the variability among these elements are differently captured in both models. To outline this in this section, we present distinctions between goal models and feature models and compare variability types in both models.

We refer to goal models when employed in product lines as *family goal models*, which represent the intentional space of a domain for which the product line is developed. Several works proposed a set of transformation rules to produce feature models from family goal models [13,20]. However, such approaches do not take into account the details and differences between these two models.

A family goal model is an artifact representing stakeholders' objectives and strategies. It describes the intentional space of stakeholders of a domain.[4] Feature models have a different purpose. They describe the configuration space of

[4] The stakeholders include final users, managers, designers, clients, etc. However, in the rest of the paper, we only concentrate the final users and representation of their intentions in goal models.

a product line (i.e., a set of products). Accordingly, they represent characteristics of software products that belong to a product line. Features that are assigned to goals can be seen as the realization of requirements (solutions) in the software products. One or more features may be developed for realizing one or more tasks in goal models. Different terms are defined for goals: achieve, maintain, avoid, and cease, while features can only be selected or deselected [8].

The relationships have also different meanings in both models. Intentional relations in family goal models show decompositions of stakeholders' goals into subgoals and further to low-level subgoals and tasks [21]. An intentional decomposition (except for soft goals) in a goal model is an entailment, i.e., an AND-decomposition of source intentional elements is one possible combination (among others) of source intentional elements that implies the target intentional element. On the other hand, an AND-decomposition in a feature model is a *PartOf* relation that implies a parent feature contains its child features.

With respect to the variability in the context of software product lines, we also distinguish between *product line variability* and *behavioral variability* [11]. Product line variability refers to differences between products in a product line, which may exist among their requirements, design models and implementation models [22]. On the other hand, behavioral variability represents the various behavior that a single system may be used by its user [23]. For example, workflow patterns in process modeling languages such as BPMN, BPEL, and activity diagrams provide mechanisms for representing variability in behavior of a single system.

In family goal models, OR and XOR relations represent variability in stakeholders' goals (i.e., intentional variability), and the ways that stakeholders' goals can be achieved (i.e., behavioral variability). When developing a reference design and implementation models from the family goal model, these variability relations can lead to either *intentional/behavioral variability* or *product line variability* in the reference models. In the family goal model, OR-decompositions (and similarly XOR-decompositions) represent *intentional/behavioral variability*, if all products having a target intentional element involved in the OR relation (XOR relations), contain all source intentional elements of the OR-decomposition (XOR-decomposition) in their goal models. For example, as shown the Fig. 1, a final user in an on-line shop may variably choose to perform Check if Return Customer or Check Credit tasks in order to satisfy the hard goal Trustworthiness of the Customer Determined, even though both tasks are invariably available for them in all products.

On the other hand, in the family goal model, OR-decompositions (and similarly XOR-decompositions) represent product line variability if source intentional elements involved in OR-decompositions (XOR-decompositions) vary between different products that contain the target intentional element. For example, in Fig. 1, some products provide Receive Payment by Card or Payment Postponed to achieve Payment Collected, while other products offer Payment Postponed and Receive Payment By Non-Card to achieve Payment Collected goal.

Feature models aim at modeling differences between products of a product line [3,24,25]. Therefore, feature models only represent product line variability, and variability related to a single system is represented in feature models as commonalities. For example, since XOR relation between Check if Return Customer and Check Credit in the family goal model is a behavioral variability, the features mapped to those tasks are considered as mandatory features. On the other hand, since the OR intentional relation between Deliver Item and Courier Deliver is a product line variability, there is a variability relation between their corresponding features (see Fig. 5). The variability relations in feature models are resolved when a product is derived from a product line, i.e., when a new configuration is created.

We should note that a family goal model represents stakeholder's requirements and their variability. On the other hand, feature models not only show variabilities in requirements but also encapsulate product line variabilities in designing and implementation models [26]. Therefore, a feature model may contain several variability relations that do not exist in the goal model. For example, there is alternative relation between Autorize. Net, Cyber Source, and Link Point features for implementing Payment Gateway feature, which shows variability in the level of a product line implementation. However, this variability relation does not exist in the family goal model, because Receive Payment By Card task is not concerned with different ways of implementation.

Having investigated the notion of variability in the family goal models and feature models, we found out that the semantics of variability in these two models are different. Variability relations in goal models can be either product line variability or intentional/behavioral variability while feature models only represent the product line variability.

The existing goal model notations do not discriminate between product line variability and intentional/behavioral variability. Thus, we extend the standard goal model notation, introduced in Sect. 2.1, in order to distinguish the types of variability in the family goal model.

For OR-decompositions (XOR-decompositions), the product line variability and intentional/behavioral variability are distinguished using the *VP* notation. OR-decompositions, showing product line variability, are labeled as VP. For example, the VP annotation on the Item Delivered goal in Fig. 1 shows that different products having the goal Item Delivered vary in ways they provide fulfillment of this goal for their final users (i.e., Delivered By Courier or Delivered By Company). Therefore, during the development time, at

least one of the source intentional elements might be selected for the target product in goal model.

In standard goal modeling [13,16,17], AND-decomposing a target intentional element (goals or tasks) into source intentional elements implies that the satisfaction of the target intentional element is dependent on the satisfaction of all of its sources. However, in family goal models, there may be some source of intentional elements in AND-decompositions whose fulfillment is necessary for the target intentional element of a particular product, but their non-fulfillment does not make the target intentional element unsatisfiable in other products. To enable family goal models to present and describe these situations, we add the notion of optional goals, resembling optional features in feature model. Hence, intentional elements in an AND-decomposition can be optional if their non-fulfillment does not lead to non-fulfillment of the target intentional element. For example, Fig. 1 shows that non-fulfillment of task Apply Discount does not necessarily lead to the non-satisfaction of the goal Payment Managed.

In family goal models, we use soft goal as means for resolving product line variability in the intentional space. Therefore, contribution links can propagate the desired satisfaction level of soft goals into goals and tasks and help in the selection of a proper variant of product lines-based intentional variability. For example, the Minimize cost delivery soft goal can be used as criteria to resolve product line variability in the Item Delivered goal. We formally define family goal models as follows:

Definition 3 (*Family goal model*) A family goal model $FGM = \langle \mathcal{G}, \mathcal{C}, \mathcal{D}^F \rangle$ extends a goal model $GM = \langle \mathcal{G}, \mathcal{C}, \mathcal{D} \rangle$ as follows: The decomposition relation $\mathcal{D}$ is extended by decompositions that cover product line variability $\mathcal{D}^F \subseteq \mathcal{G} \times \{IOR, XOR, AND, AND\text{-}O, IOR\text{-}VP, XOR\text{-}VP\} \times \mathcal{P}(\mathcal{G})$.

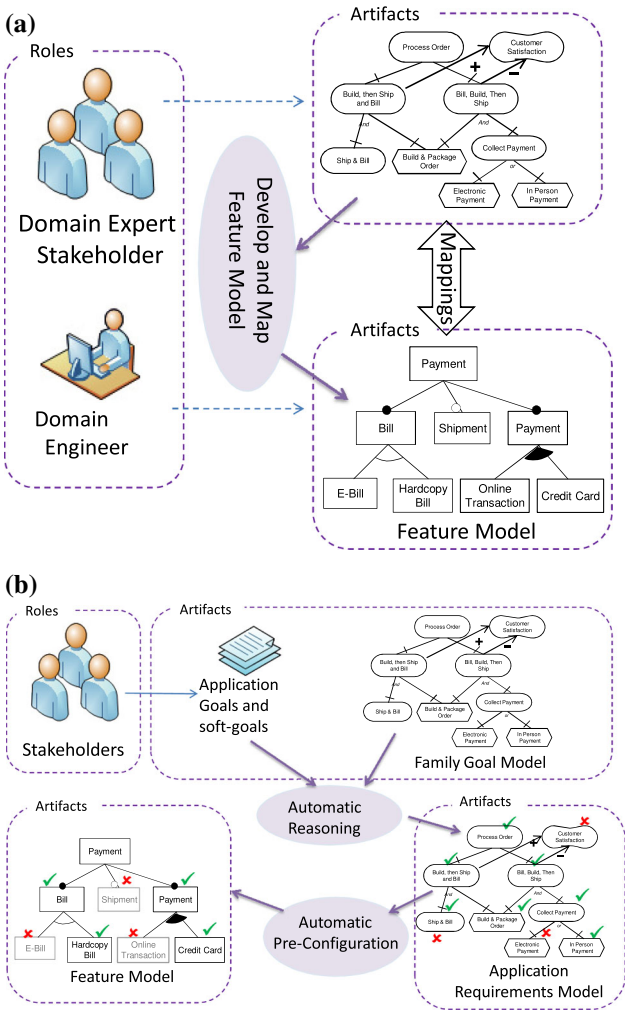

Fig. 3 Contribution to the SPLE life cycle. **a** Domain engineering. **b** Application engineering

3.2 Development life cycles

In our approach, we use goal models in both life cycles of SPLE. Figure 3 shows a family goal model and a feature model in combination with mappings between them in the domain engineering life cycle, as well as the application goal model and configuration in the application engineering life cycle.

In the *domain engineering* life cycle (see Fig. 3a), one of the most important issues are the elicitation, representation and management of different stakeholders' functional and non-functional requirements. This is reflected by the goal model. Hence, in domain requirements engineering phase, first the family goal model is created by domain engineers. High-level goals (e.g., Order Processed) and soft goals (e.g., Minimize Risk) of the stakeholders are discovered, and then, the high-level goals are decomposed into lower-level goals and finally tasks. Goal decomposition is done by following the framework proposed by Liaskos et al. [27] where intentional variability concerns are recognized for each goal. Then, the goals are refined according to the variability concerns. After refining goals, requirements engineers analyze the impacts of each subgoal on the soft goals and model the impacts using contribution links. After developing the family goal model, using the proposed extensions (i.e., VP and optional), the family goal model is analyzed with respect to product line variability, and the goals are annotated with the proper annotations.

The feature model describes elements of a system, their functionality and how different elements depend on each other. Finally, relationships between stakeholders' goals and SPL features, which realize these goals, are represented by mappings between goals and features, describing a realization of goals by system features (cf. [10]).

The feature model is generated from the family goal model by converting product line variability into variability rela-

tions in the feature model. In the family goal model, goals are finally refined to tasks, which show the requirements for designing a product line. These tasks are the sources for designing features in the feature model. Hence, features are developed based on the tasks in the family goal model and mapped to the corresponding tasks. Mappings represent the realization of intentional elements by features. One feature can operationalize several intentional elements, and one intentional element can be realized by several features. We propose a mapping model, which is based on template-based approach proposed by Czarnecki and Antkiewicz [28]. Although mapping between features and tasks may require domain engineers effort, this mapping needs to be created only once. Our approach only requires that atomic tasks in the family goal are mapped to the features, which provide realization for those features. Also, mapping features in the feature model to development artifacts is a common practice in the software product line engineering [29], which is required for further reusability in application engineering life cycle. After designing features for realizing tasks in the family goal model, the feature model is generated by developing variability relations between features based on the variability relations in the family goal model.

In the *application engineering* life cycle (see Fig. 3b), application engineer communicates and understands the stakeholders' needs and requirements by identifying their objectives. The family goal model is used as a reference model for communicating with the customers and capturing their goals. Afterward, a particular application goal model is obtained based on an individual stakeholder's goals and business objectives and by executing the backward reasoning algorithm [14]. Accordingly, based on the mappings between goals and features, a preconfiguration process is executed and features which are not based on the current stakeholders' objectives are filtered out from the feature model, by preserving the feature model constraints. The details of the preconfiguration process are out of scope of this paper. The detailed process of preconfiguration is outside of the scope of this paper. Interested readers can find more details about the preconfiguration process in our previous work [10].

4 Goal-oriented requirements engineering for SPLs

The comparison of goal and feature models and their variability resulted in an extension of goal models, the so-called *family goal models*, which present different notions of variability. In the following, we specify the *realization* of goals from a family goal model by features of a feature model in terms of mappings. Based on these mappings, we specify *realization inconsistencies* that might happen due to contradicting relationships of goals and their mapped features.

4.1 Relating intentional elements to features

In the domain engineering life cycle, goal models capture intentional variability [7,27] and describe the intentions behind existing features in the software product line. Hence, using the goal model, we can ensure that existing features and variability relations in feature models are aligned with intentional variability in the goal models. We can also trace back differences in products to differences in the intentions of the stakeholders.

In our model, for representing an explicit mapping (i.e., a mapping indicated by domain engineers), a mapping relation for each mapped task is developed. For example, the Receive Payment By Card task is mapped to the Debit Card Payment, Credit Card Payment and Payment Gateway features; hence, a mapping relation Φ_{RPBC} (*Receive Payment By Card*, {*Debit Card Payment*, *Credit Card Payment, Payment Gateway*}) is created. If a feature is mapped to more than one goal or/and task, then the corresponding feature appears in the mapping relations of all those goals or/and tasks. After explicit mapping between tasks in a family goal model and features in a feature model, we can drive implicit mappings between intermediate tasks and goals and features through existing relations in goal models and feature models. For example, we can infer that the goal Payment Managed in the family goal model is implicitly mapped to the feature Payment Management (see Fig. 1).

Definition 4 specifies mappings between a family goal model and a feature model, as well as the resulting *family requirements model*, which is the combination of both models including the mappings between them.

Definition 4 (*Mapping and family requirements model*) Let $FGM = \langle \mathcal{G}, \mathcal{C}, \mathcal{D}^F \rangle$ be a family goal model where $\mathcal{G} = (\mathcal{G}_g \cup \mathcal{G}_t \cup \mathcal{G}_s)$ and $FM = \langle \mathcal{F}, \mathcal{F}_M, \mathcal{F}_O, \mathcal{F}_{IOR}, \mathcal{F}_{XOR}, \mathcal{F}_{incl}, \mathcal{F}_{excl} \rangle$ a feature model, $\Phi_i(G_i, \mathcal{F}_i)$ is a mapping relation between a intentional element $G_i \in (\mathcal{G}_g \cup \mathcal{G}_t)$ and a set of features $\mathcal{F}_i \subset \mathcal{F}$, and Φ denotes the set of all mappings between *FGM* and *FM*. A family requirements model Π is defined as a triple of *FGM*, *FM* and Φ : $\Pi = \langle FGM, FM, \Phi \rangle$.

4.2 Realization inconsistency

In a family requirements model $\Pi = \langle FGM, FM, \Phi \rangle$, mappings Φ describe the realization of goals by features, while goals depend on intentional relations *IR* and features depend on feature relations *FR*. Relations in a family goal model capture the relations among objectives of stakeholders, while feature relations originate from intentional relations in the family goal model. This means that variability and commonality in feature models should be aligned with intentional relations defined in family goal models. Therefore, there is an

inconsistency in a family requirements model if intentional relations do not coincide with the feature relations. Hence, we aim at developing a logical-based technique that detects whether the intentional/behavioral relationships in a family goal model are correctly implemented with the commonality and variability relationships in feature models.

An inconsistency can only happen if source and target elements of both relations (i.e., *IR* and *FR*) are mapped to each other. We make the following assumptions: (1) only hard goals and tasks are mapped to features, since soft goals usually describe non-functional requirements; (2) these mappings are collaboratively developed by domain experts and domain engineers; (3) only contributions Make ($\dot{+}$) and Break ($\dot{-}$) are considered in the inconsistency detection of a family requirements model since only these contributions are sufficient for goal fulfillment (cf. Sect. 2.1); (4) unmapped elements (i.e., elements which are not mapped explicitly or their implicit mapping cannot be derived from existing relations in the family goal model and the feature model) do not contribute to inconsistencies; and (5) a mapping of a feature means implicitly also a mapping of the parent feature to the parent goal of the mapped goal (cf. mapping principles in [10]).

Assume *FR* is a feature relation, with target feature F and source features $F_1, \ldots, F_n$, i.e., F depends on $F_1, \ldots, F_n$. Likewise, *IR* is an intentional relation with target element $G \in \mathcal{G}$ and source intentional elements $G_1, \ldots, G_m \in \mathcal{G}$. The fulfillment of G depends on the fulfillment of $G_1, \ldots, G_m$. We distinguish between potential and strong inconsistency.

Definition 5 (*Potential inconsistency*) A permissible satisfaction of the intentional element G, which depends on the satisfaction of $G_1, \ldots, G_m$, might lead to an incorrect configuration of feature F, while F depends on $F_1, \ldots, F_n$.

Definition 6 (*Strong inconsistency*) All permissible satisfactions of G, which depend on the satisfaction of $G_1, \ldots, G_m$ with respect to *IR*, lead to an incorrect configuration of feature F (F depends on $F_1, \ldots, F_n$).

Figure 2 depicts an example of a strong inconsistency between a feature relation and an intentional relation. The feature Oder Preparation has three mandatory children (Approve Order, Item preparation and Order Confirmation). The corresponding goal Order Verified and Approved (OVA) has exclusive subgoals, which are mapped to the children of Order Preparation. Thus, no goal fulfillment of OVA will lead to a valid feature configuration.

Another example illustrates a potential inconsistency. Feature Buy Item is mapped to task Acquire From Supplier (AFS), feature Build Item is mapped to task Build and Package Item (BPI), and finally, feature Transfer from Warehouse is mapped to the task Obtain from Stock (OFS). By mapping goals to features, the implicit mapping between parent goals (e.g., Item Available) and parent features (e.g., Item Preparation) is established. Let us assume that an application engineer wants goal Item Available be satisfied in a target product. This can be achieved by selecting Build and Package Item (BPI) to be fulfilled, but not Acquire From Supplier (AFS) and Obtain from Stock (OFS). However, features Transfer from Warehouse and Obtain from Stock are mandatory features, and removing them from the feature model violates the feature model relations. Thus, this can lead to a potential inconsistency.

Table 1 shows combinations of intentional relations (*IR*) and feature relations (*FR*). The comparison between contributions ($\dot{+}$ and $\dot{-}$) to feature groups means that the goals are mapped to the siblings of the feature group.

Among intentional relations, the relations IOR-VP, XOR-VP and AND-Optional depict product line variability (difference in intentional elements of different products) and the other intentional relations illustrate intentional/behavioral variability (variability in the intentional elements of the stakeholders of a product) in the intentional space. Accordingly,

Table 1 Correspondence between intentional relations and feature relations

Features relations	Intentional relations							
	AND	Optional	IOR	IOR-VP	XOR	XOR-VP	$\dot{+}$	$\dot{-}$
Parent-mandatory child	✓	±	✓	±	✓	↯	✓	↯
Parent-optional child	✓	✓	✓	✓	✓	✓	✓	✓
OR feature group	✓	✓	✓	✓	✓	✓	✓	±
Alternative feature group	↯	✓	↯	✓	↯	✓	↯	✓
Include relation	✓	±	✓	±	✓	↯	✓	↯
Exclude relation	↯	✓	↯	✓	↯	✓	↯	✓

Legend no inconsistency (✓), strong (↯) and potential (±) inconsistency. $\dot{+}$ and $\dot{-}$ are sufficient contribution links. IOR and XOR show intentional variability, which is converted to behavioral variability and should remain for run-time. IOR-VP and XOR-VP show intentional variability, which is transformed to product line variability and should be resolved during the configuration of products

the former relations should be aligned with variability relation in feature models (i.e., Optional, Alternative and OR), and the latter should be aligned with commonality (i.e., Mandatory relation) in the feature model.

5 Knowledge base of the family requirements model

In order to recognize inconsistencies in family requirements models, we need a formal representation formalism that captures intentional relations, feature relations and mappings between goals and features. Furthermore, we need automatic means to automatically compare these relationships and pinpoint the source of an inconsistency. Following this line of argumentation, we propose a formal knowledge representation that offers reasoning (or inference) services to facilitate model validation. We use description logic (DL) as it is expressive enough to represent all kind of elements (i.e., goals and features) and relationships and mappings between them. Besides this, DL offers quite efficient, sound and complete reasoning services.

DL modeling has been used to align between different models and to use reasoning for model verification. For instance, in [30], we proposed an approach for inconsistency detection in design artifact of services (represented in business process model) with respect to requirements that are represented in goal models. In this work, our DL model has to cover three different kinds of variability that are given by the family goal model and by the feature model.

5.1 Foundations of description logic

Description logic (DL) [31] is a decidable fragment of first-order logic (FOL).[5] A DL knowledge base consists of terminological axioms (TBox) and assertions (ABox). The TBox specifies concepts, denoting sets of individuals and roles defining binary relations between individuals. The main syntactic constructs are depicted in Table 2, supplemented by the corresponding FOL expressions.

The universal concept $\top$ is the superconcept of all concepts, i.e., $C \sqsubseteq \top$ holds for each concept C, and $\bot$ is an unsatisfiable concept. Concept inclusion axioms $C \sqsubseteq D$ mean that each individual of concept C is also an individual of D. A concept equivalence (or definition) $C \equiv D$ is an abbreviation for two concept inclusion axioms $C \sqsubseteq D$ and $D \sqsubseteq C$. A concept union is a complex concept expression

Table 2 Constructs and notations in DL and FOL syntax

Construct name	DL syntax	FOL syntax
Atomic concept, atomic role	C, R	$C(x), R(x, y)$
Concept inclusion axiom	$C \sqsubseteq D$	$\forall x.C(x) \rightarrow D(x)$
Concept union	$C_1 \sqcup \cdots \sqcup C_n$	$C_1(x) \vee \cdots \vee C_n(x)$
Concept intersection	$C_1 \sqcap \cdots \sqcap C_n$	$C_1(x) \wedge \cdots \wedge C_n(x)$
Concept negation	$\neg C$	$\neg C(x)$
Existential quantification	$\exists P.C$	$\exists y.(P(x, y) \wedge C(y))$

and refers to a disjunction in FOL. Likewise, a concept intersection refers to a conjunction in FOL. A concept negation $\neg C$ is the set of all individuals that are not individuals of the concept C.

Due to the well-defined model-theoretic semantics of DL, there are sound reasoning algorithms that offer practically efficient reasoning services. In the remainder, we use subsumption checking as one of the basic reasoning services. Subsumption checking refers to the question whether a concept C is subsumed by D, i.e., $C \sqsubseteq D$ holds in the knowledge base, whereby C and D can be complex concepts.

5.2 Representation of models and mappings

The key part of our modeling formalism is to represent the different relations of both models, combined with mappings between features and goals.

5.2.1 Intentional relations

A family goal model $FGM = \langle \mathcal{G}, \mathcal{C}, \mathcal{D}^F \rangle$ contains the *intentional relations (IR)* on goals $G \in \mathcal{G}$. The corresponding DL knowledge base Σ_{FGM} is built according to Algorithm 1. For each goal G, we represent its relations by the concept Rel_G. Relationships between goals are expressed in DL by the role *require*.

Lines 4–6 capture an IOR-decomposition, in which a goal G is satisfied only if at least one of its subgoals G_i is satisfied. This is represented in DL by a concept union over G_i. We use the role $relates$ to express relationships between goals. The representation of exclusive decompositions is straightforward (lines 7–9).[6] Conjunctive decompositions, as well as AND decompositions, which represent behavioral variability, are described by concept intersections (lines 10–12), optional goals are neglected, as the fulfillment of an optional goal

[5] In this paper, description logic (DL) is used to formalize the constraints of interests in models of interests and enable validation services. We could have used some other formalism, but we opted for DL as it is precise and expressive enough to serve our purpose—formalize constraints that need to hold between our models of interest. Comparison of different reasoning formalisms for the particular task under study is beyond the scope the paper and deserves a new paper on its own.

[6] Please note that $\otimes$ is not a standard operator in DL. For a more concise representation, we use $\bigotimes_{G' \in \{G_1,\ldots,G_n\}} \exists requires.G'$ as an abbreviation for $\bigsqcup_{G' \in \{G_1,\ldots,G_n\}} \exists requires.G' \sqcap \neg(\bigsqcup_{G'',G''' \in \{G_1,\ldots,G_n\}} (\exists requires.G'' \sqcap \exists requires.G'''))$.

Algorithm 1 Representation of the Intentional Relations Σ_{FGM}

1: Input: Family Goal Model $FGM = \langle \mathcal{G}, \mathcal{C}, \mathcal{D}^F \rangle$
2: Output: Knowledge base Σ_{FGM}
3: **for all** $(G, \gamma, \{G_1, \ldots, G_n\}) \in \mathcal{D}$ **do**
4: **if** $\gamma = IOR - VP$ **then**
5: $Rel_G := Rel_G \sqcap (\bigsqcup_{i=1,\ldots,n} \exists requires.G_i)$
6: **end if**
7: **if** $\gamma = XOR - VP$ **then**
8: $Rel_G := Rel_G \sqcap (\bigotimes_{G' \in \{G_1,\ldots,G_n\}} \exists requires.G')$
9: **end if**
10: **if** $\gamma \in \{AND, IOR, XOR\}$ **then**
11: $Rel_G := Rel_G \sqcap (\bigsqcap_{(i=1,\ldots,n) \wedge (\neg(G \circ G_i))} \exists requires.G_i)$
12: **end if**
13: **end for**
14: **for all** $(G', \dotplus, G) \in \mathcal{C}$ **do**
15: $Rel_G := Rel_G \sqcap \exists requires.G'$
16: **end for**
17: **for all** $(G', \dot{-}, G) \in \mathcal{C}$ **do**
18: $Rel_G := Rel_G \sqcap \neg \exists requires.G'$
19: **end for**

Algorithm 2 Representation of the Feature Model Knowledge Base Σ_{FM}

1: Input: Feature Model $\langle \mathcal{F}, \mathcal{F}_M, \mathcal{F}_O, \mathcal{F}_{IOR}, \mathcal{F}_{XOR}, \mathcal{F}_{incl}, \mathcal{F}_{excl} \rangle$
2: Output: Knowledge base Σ_{FM}
3: **for all** $F \in \mathcal{F}$ **do**
4: $Rel_F \equiv \top$
5: **end for**
6: **for all** $(F, \{F_1, \ldots, F_n\}) \in \mathcal{F}_M$ **do**
7: $Rel_F := Rel_F \sqcap \bigsqcap_{i=1,\ldots,n} \exists requires.F_i$
8: **end for**
9: **for all** $(F, IOR, \{F_1, \ldots, F_n\}) \in \mathcal{F}_{IOR}$ **do**
10: $Rel_F := Rel_F \sqcap \bigsqcup_{i=1,\ldots,n} \exists requires.F_i$
11: **end for**
12: **for all** $(F, XOR, \{F_1, \ldots, F_n\}) \in \mathcal{F}_{XOR}$ **do**
13: $Rel_F := Rel_F \sqcap (\bigotimes_{F' \in \{F_1,\ldots,F_n\}} \exists requires.F')$
14: **end for**
15: **for all** $(F, F') \in \mathcal{F}_{incl}$ **do**
16: $Rel_F := Rel_F \sqcap \exists requires.F'$
17: **end for**
18: **for all** $(F, F') \in \mathcal{F}_{excl}$ **do**
19: $Rel_F := Rel_F \sqcap \neg \exists requires.F'$
20: **end for**

depend on an individual requirement selection and cannot be determined in the domain engineering life cycle.

Sufficient positive contributions (lines 14–16) specify that the fulfillment of G requires the fulfillment of G'. Thus, we add the expression $\exists requires.G'$ to the definition of concept Rel_G. Sufficient negative contributions (lines 17–19) use concept negation in order to represent the exclusiveness of goals G and G'.

Axioms 1 and 2 exemplify the DL representation for an excerpt of the goal model of Fig. 1. An AND-decomposition of the goal OP (Order Processed) into subgoals OVA (Order Verified and Approved), PM (Payment Managed) and ID (Item Delivered) is described in Axiom 1. An IOR-VP-decomposition of the goal ID (Item Delivered) is given in Axiom 2.

$$Rel_{OP} \equiv \exists\, requires.OVA \sqcap \exists\, requires.PM \sqcap \exists requires.ID \quad (1)$$

$$Rel_{ID} \equiv \exists\, requires.CD \sqcup \exists\, requires.DI \quad (2)$$

5.2.2 Feature model relations

Similar to goal models, the feature model relations FR of a feature model FM are represented in a DL knowledge base Σ_{FM} (Algorithm 2). The DL representation is based on the general modeling principles of Wang et al. [32]. However, due to the different validation purpose, we adapt some modeling principles according to our particular need. We use only one role requires to describe the relations of a feature that requires other features, while Wang et al. use different roles. It is easier and more intuitive to compare concept expressions that use the same role. We use concept definitions (equivalence axioms) in order to allow for a subsumption checking between the different concepts that represent intentional and feature relations (cf. Sect. 6).

Initially, for each feature F, Rel_F is equal to the universal concept (line 4 in Algorithm 2), to capture the case that a feature does not depend on any other feature. All mandatory child features of a feature F are represented by a concept intersection (lines 6–8). An inclusive OR-decomposition of a feature F into features $F_1, \ldots, F_n$ is represented by a concept union over the mapping concepts of each feature F_i (lines 9–11). Likewise, XOR-decomposition is described by concept unions, but with a further restriction that the selection of only one feature is allowed (line 12–14). The includes integrity constraint specifies that the selection of a feature F also requires the selection of a feature F'. In DL, we define Rel_F dependent of the feature F' (lines 15–17). The excludes integrity constraint is defined similarly (lines 18–20).

Axiom 3 defines features Order Preparation, Pay Management and Shipment as mandatory children of Order Management. An exclusive grouping of the features Coupon and Percentage Discount is depicted by Axiom 4. The second part of the axiom excludes the selection of multiple child features. Axiom 4 describes an integrity constraint, i.e., feature E-bill includes feature E-mail.

$$Rel_{OrderManagement} \equiv \exists\, requires.OrderPreparation \sqcap \exists\, requires.PayManagement \sqcap \exists\, requires.Shipment \quad (3)$$

$$Rel_{HandelDiscount} \equiv \exists\, requires.Coupon \otimes \exists\, requires.PercentageDiscount$$

$$Rel_{E-bill} \equiv \exists\, requires.E-mail \quad (4)$$

5.2.3 Mapping representation

Besides intentional relations *IR* and feature relations *FR*, we have to represent the realization of goals by the corresponding features in terms of mappings in the knowledge base Σ_Φ. A mapping is described as a concept equivalence in the knowledge base. If there is a mapping $\phi_i(G_i, \mathcal{F}_i)$ $(\phi_i \in \Phi)$ from a goal G_i to the set of features $\mathcal{F}_i$, we represent the mapping by an axiom $G \equiv \mathcal{P}(\mathcal{F}_i)$ where $\mathcal{P}$ shows propositional formula over features in $\mathcal{F}_i$. Axiom 5 shows the mapping relation between the task Receive Payment By Card and the features Debit Card Payment, Credit Card Payment and Payment Gateway.

$$\begin{aligned} &Rel_{ReceivePaymentByCard} \\ &\equiv (Rel_{DebitCardPayment} \sqcup Rel_{CreditCardPayment}) \\ &\sqcap Rel_{PaymentGateway} \end{aligned} \tag{5}$$

6 Verification of family requirements models

The verification aims at detecting inconsistencies between intentional relations in the family goal model and variability/commonality relations in the feature model. Hence, the verification compares the intentional and feature relations of mapped elements. For each mapping, we check whether there is a strong or potential inconsistency, or even no inconsistency, according to the correspondences of Table 1.

6.1 Verification procedure

The knowledge base contains DL concepts Rel_G and Rel_F that describe intentional relations *IR* of G and feature relations *FR* of F. From a logical point of view, concepts Rel_G and Rel_F represent formulas, and we compare them in order to analyze the influence of Rel_G on Rel_F. (1) A potential inconsistency is identified if the satisfaction of *IR* does not necessarily imply the satisfaction of *FR* (expressed by Rel_F). Thus, the concept Rel_G is not subsumed by Rel_F, i.e., Rel_G does not imply Rel_F. (2) A strong inconsistency is recognized by contradicting relations of goal G and feature F. Thus, the intersection of Rel_G and Rel_F is unsatisfiable, i.e., the intersection is subsumed by the empty concept $\bot$ in DL.

Accordingly, we check either whether $Rel_G \sqcap Rel_F \sqsubseteq \bot$ (strong inconsistency) or whether the implication $Rel_G \Rightarrow Rel_F$ holds, i.e., $\neg Rel_G \vee Rel_F$ is a tautology (no potential inconsistency). In DL, this is represented by a concept union: $\neg Rel_G \sqcup Rel_F$. For this purpose, we extend the knowledge base by verification concepts $Valid_{G \wedge F}$ (strong inconsistency) and $Valid_{G \Rightarrow F}$ (potential inconsistency) for each mapped elements (G, F) $(\phi(G, \mathcal{F})$ with $F \in \mathcal{F})$ (cf. Definition 7).

Definition 7 (*Knowledge base Σ for the Verification*) The knowledge base $\Sigma := \Sigma_{FM} \cup \Sigma_{FGM} \cup \Sigma_\Phi$ is extended as follows: For each mapped elements (G, F) with $(\phi(G, \mathcal{F})$ $F \in \mathcal{F})$ in Σ, we insert the following axioms:

(1) $Valid_{G \Rightarrow F} \equiv \neg Rel_G \sqcup Rel_F$
(2) $Valid_{G \wedge F} \equiv Rel_G \sqcap Rel_F$

Given the final knowledge base, we get the verification result *en passant*. We classify the verification concepts $Valid_{G \Rightarrow F}$ and $Valid_{G \wedge F}$ of the knowledge base Σ, leading to the following observations:

1. A verification concept $Valid_{G \Rightarrow F}$ indicates a potential inconsistency or no inconsistency. If $Valid_{G \Rightarrow F}$ is classified equal to the universal concept $\top$, we can guarantee that the fulfillment of intentional relations *IR* of G ensure the fulfillment of feature relations *FR* of F.
2. Otherwise, $Valid_{G \Rightarrow F} \not\equiv \top$ holds, and we know that there is at least a potential inconsistency, but which kind of inconsistency is unknown. We identify a strong inconsistency if the other verification concepts $Valid_{G \wedge F}$ is classified equal to the empty concept $\bot$; otherwise, it is a potential inconsistency.

As described in Sect. 3, if there is a mapping between a goals G and a feature F, the fulfillment of G determines whether F will be removed or not.

6.2 Correctness of the verification

For a family requirements model $\Pi = \langle FGM, FM, \Phi \rangle$, the verification recognizes inconsistencies between goal G and feature F, based on their relations *IR* and *FR*.

6.2.1 Relationship coverage in the knowledge base

Relationships of both models are represented in a common DL knowledge base Σ. Intentional relations *IR* of a goal G are represented by a single concept Rel_G. Likewise, feature relations *FR* of a feature F are covered by a concept Rel_F. Mappings between goals and features are represented by equivalence axioms in the knowledge base. For each mapping, the corresponding concepts Rel_G and Rel_F are compared in order to determine the influence of intentional relations *IR* on feature relations *FR*.

Based on this representation, we compare whether Rel_G is subsumed by Rel_F or the intersection of them is a satisfiable concept. With respect to Table 1, the correspondences between intentional and feature relations are reduced to subsumption checking and satisfiability in DL.

6.2.2 Verification principles

As a last step, we have to show that the detection of strong and potential inconsistency is correctly achieved in the verification, i.e., the classification of the verification concepts $Valid_{G \Rightarrow F}$ and $Valid_{G \wedge F}$ holds if there is an inconsistency between G and F. Lemma 1 summarizes these statements.

Lemma 1 *Let $(\phi(G, \mathcal{F})$ with $F \in \mathcal{F})$ be a mapping in a family requirements model Π the following holds: (i) the concept $Valid_{G \Rightarrow F}$ is classified equal to the universal concept $\top$, iff all dependent features of feature F appear in a feature configuration, whenever feature F, which is mapped to G, appears; and (ii) the concept $Valid_{G \wedge F}$ is classified equal to the empty concept $\bot$, iff there is no configuration possible where F appears when G is fulfilled.*

We sketch only the proof for the first statement, but the proof of the second statement is based on the same argumentation.

Proof '$\Rightarrow$' For a mapping $(\phi(G, \mathcal{F})$ with $F \in \mathcal{F})$, the concept $Valid_{G \Rightarrow F}$ is equal to the universal concept $\top$. We demonstrate that all dependent features of feature F will appear in a configuration that contains F. Let $F_1, \ldots, F_n$ be the dependent features of feature F. From the classification of $Valid_{G \Rightarrow F}$, we know that the subsumption $Rel_G \sqsubseteq Rel_F$ holds, while both concept definitions contain the relationships of goal G and feature F, and the same roles are used in both concept definitions. The subsumption $Rel_G \sqsubseteq Rel_F$ can only hold, if each dependent features $F_i (i = 1, \ldots, n)$ of F is mapped to a goal G_j (i.e., $F_i \equiv G_j$), and the goal is a dependent goal of G, i.e., G cannot be fulfilled if G_j is not fulfilled. Therefore, F_i will be in each configuration that contains F.
'$\Leftarrow$' We demonstrate the other direction by contradiction. Assume F_i is a dependent feature of F that has to appear in each configuration if F appears, but $Valid_{G \Rightarrow F}$ is not equal to the universal concept $\top$. Either (1) the structures in the concept definitions of Rel_G and Rel_F are different, i.e., different types of relationships, or (2) feature F_i is not mapped to goal G_j, where G_j is a dependent goal of G that appears in Rel_G, and F_i is a dependent feature of F that appears in Rel_F. In both cases, we cannot guarantee that F_i occurs in a configuration if F occurs, because no dependent goal of G guarantees the selection of F_i. This is a contradiction to our assumption. □

6.3 Verification exemplified

We show how DL reasoning detects a potential and strong inconsistency between family goal intentional relations and variability and commonality relations in feature models, based on the examples in Fig. 2.

Feature Item Preparation(IP) has two mandatory child features Buy Item (BI) and Transfer From Warehouse (TFW) (Axiom 6). Feature IP is mapped to goal Item Available(IA), and BI and TFW are mapped to goals Acquire From Supplier (AFS) and Obtain From Stock (OFS), respectively. Goals AFS, OFS and BPI are subgoals within an OR-VP decomposition of goal IA (Axiom 7). Mappings make the concepts BI and AFS, as well as TFW and OFS equivalent. (BPI does not contribute to the inconsistency.)

$$Rel_{IP} \equiv \exists requires.BI \sqcap \exists requires.TFW \quad (6)$$

$$Rel_{IA} \equiv \exists requires.AFS \sqcup \exists requires.OFS \sqcup \exists requires.BPI \quad (7)$$

In this case, the verification concept $Valid_{IA \Rightarrow IP}$ is not equal to the universal concept $\top$, since even if the mapped concepts are equal, Rel_{IA} is not subsumed by Rel_{IP}.

Strong inconsistencies between relations in family goal models and feature model are recognized if the verification concept $Valid_{G \wedge F}$ is equal to the empty concept $\bot$. Consider the strong inconsistency from Fig. 2. Feature Order Preparation(OP) is mapped to the goal Order Verified and Approved(OVA). Mandatory child features Approve Order(AO), Item Preparation(IP) and Order Confirmation(OC) are mapped to goals Approve Order(AO), Item Available(IA) and Check Correctness of Order(CCO), respectively. These three goals are in an XOR-VP decomposition, i.e., the exclude each other, while the corresponding features can only appear together. The feature relations are described by Axiom 8 as a concept intersection in DL, while the intentional relations are represented as a concept union that excludes the appearance of more than one goal (Axiom 9).

$$Rel_{OP} \equiv \exists requires.AO \sqcap \exists requires.IP \sqcap \exists requires.OC \quad (8)$$

$$Rel_{OVA} \equiv \exists requires.AO \otimes \exists requires.IA \otimes \exists requires.CCO \quad (9)$$

As the XOR-expression ($\otimes$) contains the expression $\neg(\exists requires.AO \sqcap \exists requires.IA \sqcap \exists requires.CCO)$, the intersection of Rel_{OP} and Rel_{OVA}, which is the verification concept $Valid_{OVA \wedge OP}$, is equal to the unsatisfiable concept $\bot$, indicating a strong inconsistency.

7 Evaluation

In this section, we highlight two examples of inconsistencies, which happen in online shopping case study, and analyze the performance of the inconsistency detection algorithm.

7.1 Case study analysis

We investigate some parts of the online shopping case study available in the SPLOT repository.[7] The family goal model and feature model were developed based on the existing models of the SPLOT repository. We only concentrate on two common scenarios where inconsistency can happen in a family requirements model. These inconsistencies show where variability relations in online feature models are not aligned with intentional relations in the family goal model.

As shown in Fig. 4a, there is a goal Order Confirmation and Bill Sent, which is AND-decomposed into Sent Bill and Sent Confirmation. Features corresponding to these tasks and their variability relation along with mapping relations are also shown in the figure. These mapped models have some potential inconsistencies. These inconsistencies are caused by the allowable selections of features that lead to the unsatisfaction of the Order Confirmed and Bill Sent goal. For example, Fig. 4b shows two instances of the feature model configurations that do not lead to the satisfaction of Order Confirmed and Bill Sent. These inconsistencies are since (1) the variability relations in the feature model are not aligned with variability relations in the family goal model, and (2) the mapping relations should be logical OR instead of logical AND between features that are mapped to tasks in the family goal model.

Figure 4c shows the revised version of the feature model and mapping relations that ensure consistency between stakeholders intentions (i.e., goals), feature model and mapping relations.

In the second example from this case study, we will show how our description logics-based reasoning approach helps detect inconsistencies. Therefore, throughout the discussion about this example, we will also show description logic axioms that are used for inconsistency detection. In particular, this example, shown in Fig. 5, represents goal Item Shipped (G-IS), which is OR-decomposed to Courier Deliver(T-CD) and Deliver Item (T-DI) tasks. The *VP* notation over the OR-decomposition indicates a product line variability. After applying Algorithm 1, Axiom 10 is generated.

$$Rel_{G-IS} := Rel_{G-IS} \sqcap (\exists\, requires.T - CD \sqcup \exists\, requires.T - DI) \tag{10}$$

The feature model that corresponds to the Item Shipped goal is shown in Fig. 5. Feature Shipment (Sh) is OR-decomposed to the Shipping Gateway (SG) and Store Delivery (SD) features, where the former is further OR-decomposed into the FedEX (FE), UPS, Canada Post (CP), and USPS features. Additionally, the optional feature Shipping Cost Calculation (SCC) is developed which is required by the Shipping Gateway (SG) feature and excluded by the Store Delivery (SD) feature. The transformation of the feature model to description logic using Algorithm 2 is shown by Axioms 11–14.

$$Rel_{Shipment} := Rel_{Sh} \sqcap (\exists\, requires.SG \sqcup \exists\, requires.SD) \tag{11}$$

$$Rel_{SG} := Rel_{SG} \sqcap (\exists\, requires.FE \sqcup \exists\, requires.UPS \sqcup \exists\, requires.CP \sqcup \exists requires.USPS) \tag{12}$$

$$Rel_{SG} := Rel_{SG} \sqcap \exists\, requires.SCC \tag{13}$$

$$Rel_{SD} := Rel_{SD} \sqcap \neg\exists\, requires.SCC \tag{14}$$

The mapping model in Fig. 5 shows that the Courier Deliver task is mapped to the Shipping Gateway and Shipping Cost Calculation features and the Deliver Item task is mapped to the Store Delivery and Shipping Cost Calculation features. The transformation of the mapping relations into description logic generates Axioms 15–18:

$$Valid_{T-CD \Rightarrow \langle SG,SCC \rangle} \equiv \neg Rel_{T-CD} \sqcup (Rel_{SG} \sqcap Rel_{SCC}) \tag{15}$$

$$Valid_{T-CD \sqcap \langle SG,SCC \rangle} \equiv Rel_{T-CD} \sqcap (Rel_{SG} \sqcap Rel_{SCC}) \tag{16}$$

$$Valid_{T-DI \Rightarrow \langle SD,SCC \rangle} \equiv \neg Rel_{T-DI} \sqcup (Rel_{SD} \sqcap Rel_{SCC}) \tag{17}$$

$$Valid_{T-DI \sqcap \langle SD,SCC \rangle} \equiv Rel_{T-DI} \sqcap (Rel_{SD} \sqcap Rel_{SCC}) \tag{18}$$

After performing reasoning over the generated axioms, the results show that both $Valid_{T-CD \Rightarrow \langle SG,SCC \rangle}$ and $Valid_{T-CD \sqcap \langle SG,SCC \rangle}$ equal to universal concept $\top$, which shows that there is no inconsistency for the mapping and relations in the family goal model and the feature model. However, results of the reasoning reveal that $Valid_{T-DI \sqcap \langle SD,SCC \rangle}$ and $Valid_{T-DI \Rightarrow \langle SD,SCC \rangle}$ are equal to the bottom concept $\bot$ (i.e., $Valid_{T-DI \sqcap \langle SD,SCC \rangle} \equiv Valid_{T-DI \Rightarrow \langle SD,SCC \rangle} \equiv \bot$). This shows that there is a strong inconsistency between task Deliver Item and features Store Delivery and Shipping Cost Calculation. The inconsistency is because the Store Delivery feature excludes the Shipping Cost Calculation feature, while they both are mapped (using an AND relation) to Deliver Item. By changing the mapping (i.e., removing the mapping between Shipping Cost Calculation (SCC) feature and Deliver Item (T-DI) task), the inconsistency in the family requirement model can be resolved.

7.2 Performance evaluation

The main objective of our study in this section is to analyze (1) the performance of the verification algorithm and (2) the factors of influence on the performance.

[7] Software Product Lines Online Tools—http://www.splot-research.org/.

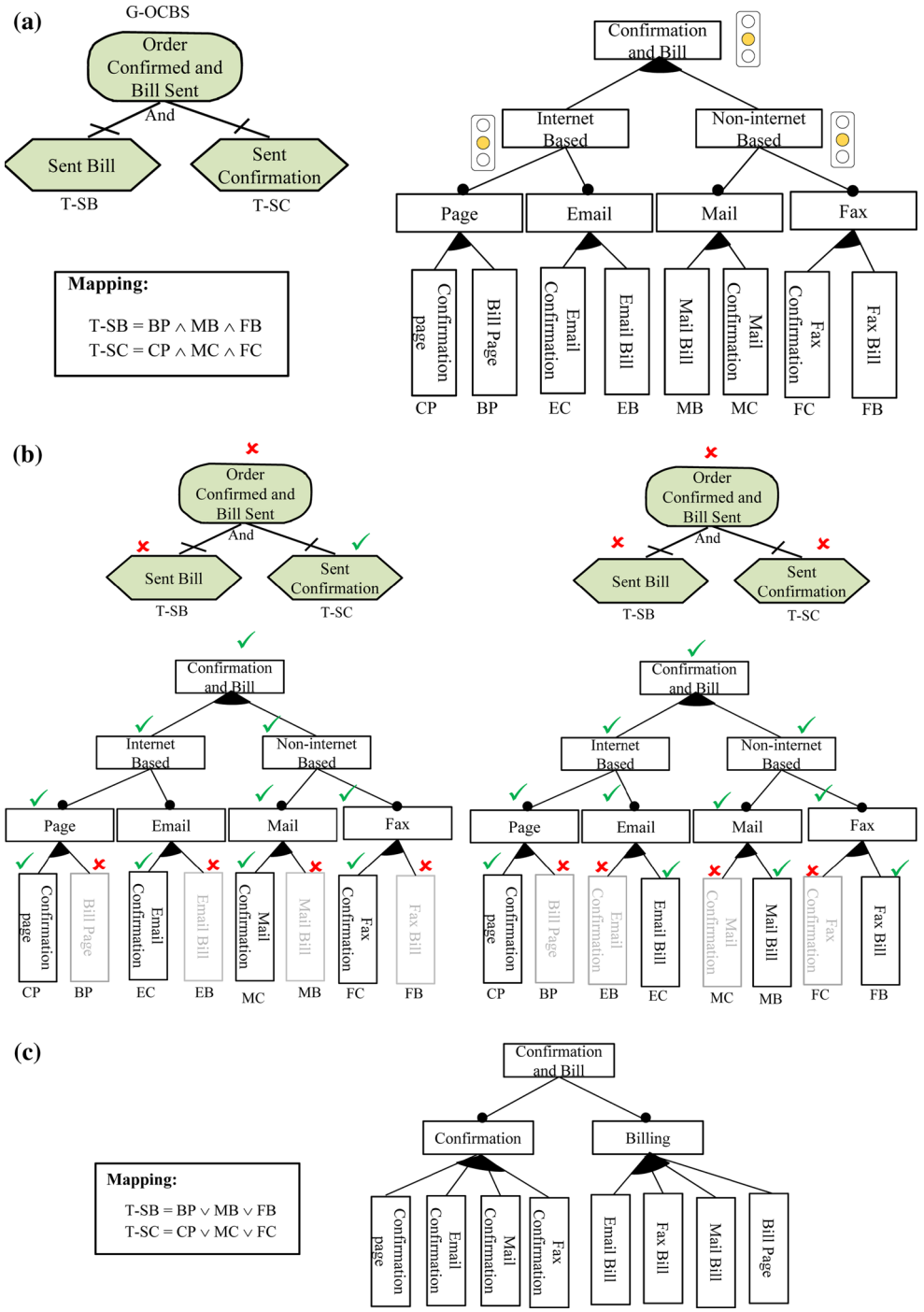

Fig. 4 A part of goal model and its corresponding feature models. **a** Order Confirmed and Bill Sent Goal and corresponding features and mapping relations. **b** Sample Configurations of the feature models which lead to unsatisfaction of Order Confirmed and Bill Sent Goal. **c** Consistent feature model and mapping relations

7.2.1 Scope of the evaluation

In order to analyze the performance of the verification algorithms and the factors of influence on this verification approach, we summarize our investigations by the following four research questions:

- RQ1: How does the execution time of the inconsistency detection algorithms scale-up as the size of family requirements model increases?
- RQ2: Does the increase or decrease in product line variability in the family goal model have significant impact on the running time of inconsistency detection algorithms?

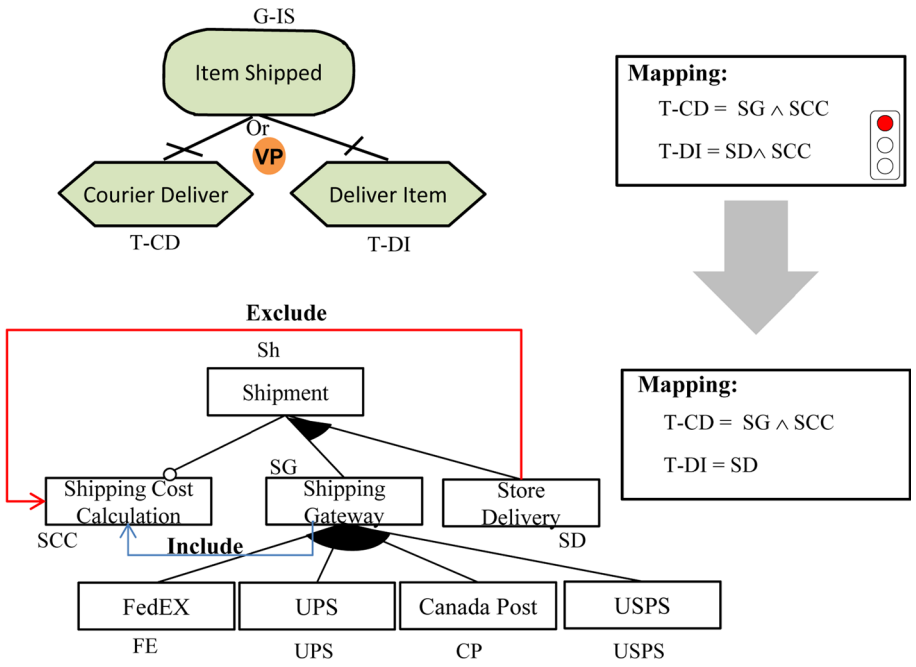

Fig. 5 Item Shipped goal and its corresponding feature model

Integrity Constraint:
Shipping Gateway **Includes** *Shipping Cost Calculation*
In Store Delivery **Exclude** *Shipping Cost Calculation*

– RQ3: Does the percentage of the inconsistency types have major impact on the running time of algorithms?
– RQ4: Does the different distributions of intentional relations in the family goal model have significant impact on the running time of inconsistency detection algorithms?

Besides this, the study also serves as an empirical confirmation of the correctness of our verification approach.

7.2.2 Experimental setting

In order to investigate the above research questions, we applied the simulation modeling technique by following guidelines similar to those proposed in [33]. We selected the simulation technique as it is commonly employed in the context of software product line verification and configuration [3,34,35].

Hence, we developed a generator, by utilizing the FAMA framework, to randomly developed family goal models, feature models and mapping based on the given parameters. By setting the parameters, the generator produces the random models, which satisfy the requested characteristics.

To control the number and kinds of inconsistencies in the family requirements models, we first randomly produced family goal models and then generated feature models from goal models by transforming product line variability in the goal models into variability relations in feature models and behavioral variability into mandatory relations. This way, we can generate random inconsistencies in the family requirements model, by considering inconsistency patterns illustrated in Table 1 and check whether the proposed algorithms can find these inconsistencies.

The initial feature models, which are derived from the goal models, are expanded with additional features. We based the expansion strategy for feature models on the FORM method case study [26] as between one and five implementation features were added to feature model for each capability feature. Thus, we add between 0 and 5 features to each atomic feature in the feature models to reflect the technical features that are added to realize the conceptual feature. The number of potential and strong inconsistencies varies in the family requirements model.

We investigated questions RQ1–RQ4 in two different settings. In the first setting, the distribution of intentional relations is fixed and the other potential factors of influence to answer questions RQ1–RQ3 vary. In the second setting, we analyze the influence of different distributions (question RQ4). Furthermore, in both settings, we check whether the algorithms work correctly.

Experimental setting 1 In this setting, the distribution of intentional relations in the family goal model is fixed. The setting compares different sizes of goal models (RQ1), different distributions of product line variability (RQ2) and different number of inconsistencies (RQ3). The distributions of intentional relations are considered as follows: 50 % AND-decomposition, 25 % OR-decomposition and 25 % XOR-

Table 3 Specification of eight models in the first experiment with 100 goals and tasks

Model size	PV (%)	Pot. Inc. (%)	Str. Inc. (%)
100	25	25	25
100	25	25	50
100	25	50	25
100	25	50	50
100	50	25	50
100	50	50	25
100	50	50	50

The other sixteen models have similar specification except number of goals and tasks

Table 4 Specification of models in the second experiment

AND (%)	OR (%)	XOR (%)	Avg. Time (ms)
25	37	38	5,510
50	25	25	5,680
75	13	12	5,120
37	25	38	5,710
25	50	25	5,130
12	75	13	4,950
37	38	25	5,620
25	25	50	5,840
13	12	75	5,970

After deciding on one of the intentional relation distributions, the other relations have the equal distributions in the models

decomposition. For AND-decompositions, 50 % are considered optional and the other 50 % are mandatory intentional elements. This distribution is selected in order to avoid any influence of different distributions of intentional relations on the running time for this experiment. We investigate such different distributions of intentional relations in the experimental setting 2. There are contribution links from 20 % of the goals and tasks (as source intentional elements) to the soft goals (as destination intentional elements).

With the fixed distribution of intentional relations, we generated goal models with 100, 200 and 300 goals and tasks and 10 soft goals. The number of goals and tasks was selected based on investigations of the size of practical goal models in the literature [36,37]. From existing intentional relations in the generated goal models, either 25 or 50 % of the relations are set as product line variability by annotating them with *VP*. For changing the behavioral variability and product line variability percentages, we do not change the structure of family goal model. We should note that in our approach, generated OR and XOR relations in family goal model are fixed (5 % OR-decomposition, and 25 % XOR-decomposition). The change in the product line variability is only done by annotating the generated family goal models; hence, we prevent any change in the structure of family goal models. With respect to inconsistencies, we considered 25 and 50 % of the total number of possible inconsistencies. Hence, we generated 24 family requirements models covering the wide range of goal models sizes, product line variability and inconsistency distributions. Table 3 illustrates eight generated models with 100 number of goals and tasks. The other 16 models have similar characteristics except different number of goals and tasks (i.e., 200 and 300). We use 10 different models for each kind of generated family requirements model to reduce the impact of the randomness of generated models on the running time.

Experimental setting 2 This setting concentrates on question RQ4, which aims at investigating the effect of the distribution of intentional relations on the running time of the inconsistency detection. During the experiments, we generated goal models with 300 goals and tasks, 10 soft goals. We consider 50 % distribution of product line variability in all the models. For every intentional relation AND, OR and XOR, three distributions 25, 50 and 75 % are devised leading to nine different kinds of models, as outlined in the columns 1–3 in Table 4. This enables us to cover wide ranges of structural variability in family goal models. The average number of inconsistencies in all models was 50 % of the total number of possible potential and strong inconsistencies.

System information and implementation details The knowledge base creation is implemented with OWL-API. For reasoning, we used the Pellet reasoner (Pellet reasoner site: http://clarkparsia.com/pellet/). Our test system is a Notebook with an Intel Core 2 Duo T7300 CPU (2.0 GHz, 800 MHZ FSB, 4 MB L2 cache and 2GB DDR2 RAM). We used 256MB RAM for the Java VM of the Eclipse environment.

Given generated family requirements model, our tool creates an knowledge base as described in Sect. 5. The DL expressivity is $\mathcal{ALC}$. After reasoning on the family requirements model, the tool produces a list of potential and strong inconsistent mappings.

7.2.3 Experimental results and analysis

The results of the analyses of performance and influence factors are illustrated in Figs. 6 and 7. The percentage of potential inconsistencies (third column in Table 3) and strong inconsistencies (fourth column in Table 3) refer to the corresponding percentage of the total number of possible potential and strong inconsistencies that can occur in each individual model of each type. Because the models are randomly generated, the number of possible inconsistencies (potential and strong inconsistencies) is different in each model.

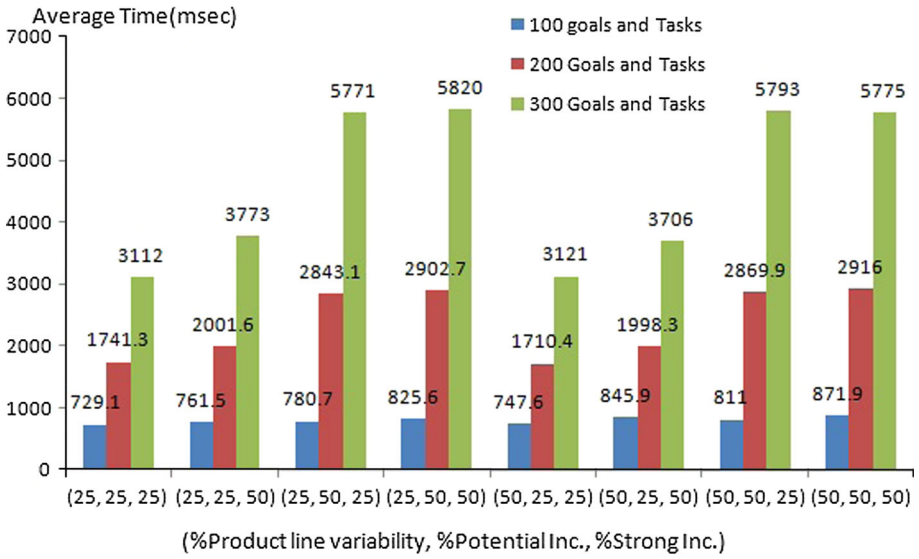

Fig. 6 The average running time for models in the first experiment: the *Y* axis shows the running time and the *X* axis shows different distribution of product line variability, potential inconsistencies and strong inconsistencies

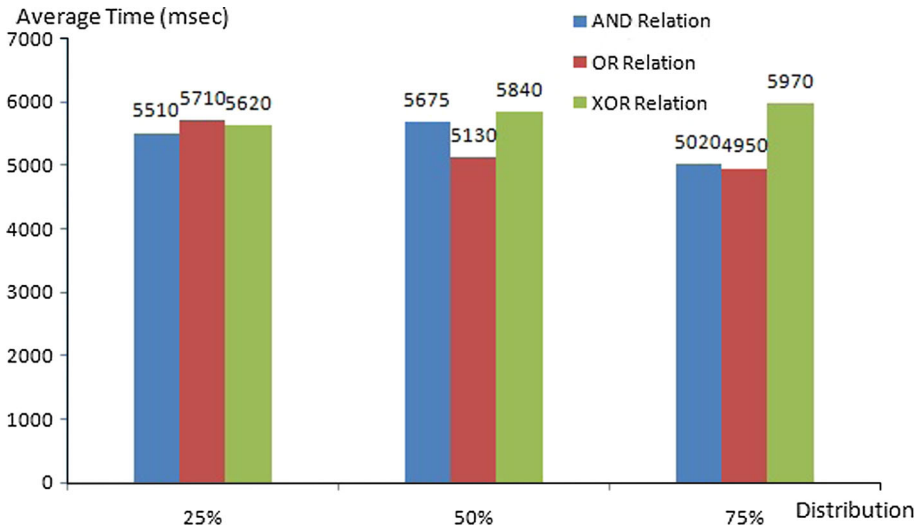

Fig. 7 The average running time for models in the second experiment: The *Y* axis shows the running time, and the *X* axis shows different distribution of intentional relations

According to the results shown in Figs. 6 and 7, we reflect and discuss each of our question.

– RQ1: As answer to this research question, we investigate the main effect of the model size on the running time, and the results indicate that the mean value of the execution time was significantly higher in the goal model with 200 elements ($M = 2{,}372.91$, SD $= 524.38$) than in the goal model with 100 elements ($M = 796.66$, SD $= 51.69$) and also the mean value of the execution time was significantly higher in the goal model with 300 elements ($M = 4{,}608.65$, SD $= 1{,}209.49$) than in the goal model with 200 elements ($M = 2{,}372.91$, SD $= 524.38$). According to the result, we can conclude that the size of models has impact on the running time. However, we can observe the execution time of our algorithm for the largest experimented model in less than 6 s on a computer with rather modest hardware capabilities. As we mentioned in experimental setting, a common size of goal models is less than 300 goals and tasks.
– RQ2: Based of our analysis, the main effect of product line variability indicates that the mean value of the execution time was not significantly greater for the 50 % product line variability ($M = 2{,}597.02$, SD $= 1{,}734.38$) than for the 25 % product line variability ($M = 2{,}588.42$, SD $= 1{,}755.15$). Therefore, different distributions of product line and software variabilities do not impact the running time of the verification algorithm. As we mentioned in Sect. 7.2.2, in the generated models, the software and product line variability distributions are computed based on the total number of generated XOR and OR relations. For example, the 75 % product line variability is computed by randomly annotating 75 % of OR and XOR relations with VP. Interestingly, the results

show that having different distributions of product line variability in the family goal models does not affect the running time, significantly.

- RQ3: The increase in the inconsistencies increases the execution time of the algorithm. This can be observed for each model size. The results show that the mean value of the execution time was significantly higher in the 50 % strong inconsistency ($M = 2{,}683.07$, SD $= 1{,}750.81$) than in the 25 % strong inconsistency ($M = 2{,}502.42$, SD $= 1{,}734.03$). Also, the mean value of the execution time was significantly higher in the 50 % potential inconsistency ($M = 3{,}164.91$, SD $= 2{,}046.67$) than in the 25 % potential inconsistency ($M = 2{,}020.58$, SD $= 1{,}113.35$). According to the results, the influence of the number of potential inconsistencies is larger than the influence of the strong inconsistencies. This result can be attributed to the two main reasons: First, as we expected, each strong inconsistency is also a potential inconsistency, and the representation of potential inconsistency is a more complex concept expression due to the negation in the verification concept. Second, according to Table 1, the total number of possible potential inconsistencies is larger than the total number of possible strong inconsistencies; consequently, 25 and 50 % potential inconsistencies are higher than 25 and 50 % strong inconsistencies.
- RQ4: According to the results for this research question, distribution change in intentional relations has different impact over running time, while the higher number of OR relations decreases running time, the higher number of XOR relations increases the running time. Interestingly, if the half of the relations are AND relations, the running time is the higher. Also, OR relations require less verification time, while XOR relations are the most expensive one. This result is expected based on the logical representation in our knowledge base, as XOR descriptions are quite complex concept expressions using negation. However, our purpose is to check whether our approach can cope with different kinds of distributions, and this seems to be confirmed by the experimental evaluation.

As already stated, the aim of this evaluation was to demonstrate that the modeling and reasoning approach is tractable in practical software product lines. The size of 300 goals is a realistic size of the family requirements model to capture real software product lines. The evaluation shows that even if up to 50 % of all possible inconsistencies occur, the execution time is feasible (6 s) for the largest experimented model.

Finally, the evaluation confirmed the that the verification algorithms are correct as all inconsistencies are recognized by the reasoner in the DL knowledge base.

7.2.4 Threats to validity

Threats to internal validity refer to the confounding variables that might have impact on the running time of the verification algorithm and were neglected during the experiment. We prevented these kinds of threats by designing two different sets of experiments and by controlling confounding variables. For example, in the first experiment setup, we considered a fixed distribution of intentional relations and focused on the impact of other independent variables (i.e., size of model, product line variability, and number of potential and strong inconsistencies).

External validity investigates whether the results of the experiments are generalized. With respect to the size of family requirements models, our investigation over existing publications in SPL and requirements engineering communities confirmed that the sizes of generated models are aligned with the size of existing models in the literature.

In order to trace the percentages of inconsistency injected in family requirements model, we generated feature models from family goal models and added inconsistencies based on Table 1. This method leads to structural similarity between family goal models and feature models. Hence, one may concern that the samples are not proper representatives of existing models in the real world. Family goal model shows the intentions of the stakeholders and their refinements into subgoals and tasks. On other hand, feature models show hierarchical representations the features of a product line which implements the tasks in family goal model. Therefore, there are similarities between features model structure and family goal model structure. Additionally, many researchers in software product line requirements engineering domain [8,9,38,39] proposed techniques to transfer goal models into feature models, which supports our claim for similarity between family goal models and feature models.

With respect to distribution of intentional relations and variability relations, we reduced this threat by covering wide variety of distributions including 25, 50 and 75 %. Additionally, we automatically generated 100 samples for each combination to include the structure diversity in generated models.

Another threat to external validity of our approach is that the execution time is influenced by the proposed modeling and formalization approach of family requirements model. Hence, for various representation, we may have slight different execution times for each distribution of independent variables. Thus, our results cannot be generalized to other approaches such as SAT solvers, which could be used as alternative for implementation of the proposed verification approach. However, the experimental results show that employing description logic can cope with different kinds of distributions.

8 Related work

In this section, we discuss the related works and compare them with our work using a set of criteria found in the literature.

8.1 Goal modeling techniques in software product line engineering

Goal models, seen as a complementing technique for other requirements engineering approaches, are started to be applied in the context of SPLE. Yu et al. [7,8] developed an approach to (1) generating design models from goal models and (2) using the goal models for configuration of the generated design models. The authors of [7] introduce a set of annotations for generating feature models from goal models, which they extend in [8] to design models such as statecharts and components.

Similarly, Lapouchian et al. [9] introduce an approach to automatically generating business process models and their configuration with extended goal models. They introduce variation points for specifying design time and run-time variabilities. In order to generate business processes automatically from goal models, they propose ordering annotations between goals and tasks that are involved in a AND-decomposition relations.

Silva et al. [38] employ aspectual i* to support variability in software product lines. Mandatory features are considered internal elements of i*, while optional, alternative and OR features are treated as aspectual elements in their approach. Composition rules are defined for all variabilities in the models. Introducing aspects in i* models and describing composition rules for all variabilities increased the complexity of models, which leads to scalability problems. Finally, this work is based on the assumption that each optional and alternative feature is mapped into one aspect.

Goal and feature model relations are further investigated by Antonio et al. [20]. They propose an approach to deriving feature models from i* goal models. A feature is defined as a relevant characteristics of the system, while a system characteristic allows for achieving a certain goal. They integrated cardinalities into i* models to represent (optional and alternative cases) variability in the models. Furthermore, a means-end relation may be transferred into OR relation or alternative relation according to the cardinality of means involved in the relation. In addition to the extension on i* models, they introduced a set of heuristics to produce a feature model from i* models.

Borba and Silva [39] extend the i* language to represent variability and commonality. They introduce mandatory means-end, optional means-end, cardinality means-end, alternative means-end and means-end group cardinality. They also provide a set of heuristic rules along with high-level processes for transforming feature models to goal models. This approach performs product configuration using the notion of soft goals and contribution links between tasks and soft goals.

Santos et al. [40] introduce an aspect-oriented requirements modeling language called PL-AOVGraph to represent variability in software product lines. The language is an extention of AOV-graph goal models that include cardinalityMin, cardinalityMax, groupFeature, cardinalityGroupMin, cardinalityGroupMax and isFeature to support variability. In their approach, they provide a bidirectional transformation between feature models and PL-AOVGraphs, which produce one of the models when the other model is available.

Mussbacher et al. [41] apply the AoURN framework in the software product line domain. They use GRL for modeling stakeholders' objectives and propose a URN profile for defining feature models. In their approach, they create goal models and feature models independently. Next, for atomic features in a feature model, their behavior and structure are generated. Afterward, features are mapped into goals, and their impacts are identified over the goals using contribution links.

Liaskos et al. [21] exploit the intuitiveness of goal models for configuration of customizable software. In their approach alternative system, configurations are matched with alternative ways of satisfying high-level goals. Hence, by performing reasoning over a goal model, they can customize a software system.

Jureta et al. [42] proposed an abstract requirements modeling language called Techne, which introduces optional, mandatory and preference notions into the standard goal modeling languages. Techne describes inference, conflict, preference, is-mandatory and is-optional relations and models the requirements and their relations in terms of graphs called r-net. In order to identify the preferred solutions, r-nets are encoded into propositional formula and logical reasoning is employed.

Ernst et al. [43] extended the Techne language with some operations to perform paraconsistent reasoning over models represented in Techne models. The approach, called KOMBINE, describes a set of consistency criteria over requirements problems to find a solution in presence of conflict. It applies the PARACONSIST-MIN-GOAL-ACHIEVEMENT and PARACONSIST-GET-CANDIATAE-SOLUTION operations, which identify the mandatory requirements and non-mandatory requirements, respectively.

Finally, Ali et al. [44] enriched Tropos goal models with contextual information and proposed contextual goal models. They defined contextual information in terms of propositional formula and considered alternative behaviors of the system to be variants derived based on contextual information. Their technique identifies inconsistencies of the contexts specified as preconditions for an alternative behavior

Table 5 Comparisons of goal-oriented software product line approaches (+ means criterion is met with the approach and − means criterion is not met with the approach)

Approaches	Criteria							
	Goal model	Feature model	B/P variability	Conflict identification	Impact analysis	Stakeholders'v trade-off	Configuration	G/F validation
Yu et al. [7]	+	+	−	+	−	+	+	−
Lapouchian et al. [9]	+	+	+	+	−	+	+	−
Silva et al. [38]	+	+	−	+	−	−	+	−
Antonio et al. [20]	+	+	−	+	−	−	+	−
Borba and Silva [39]	+	+	−	+	−	−	+	−
Santos et al. [40]	+	+	−	+	−	+	+	−
Mussbacher et al. [41]	+	+	−	+	+	+	+	−
Liaskos et al. [21]	+	−	−	+	−	+	+	−
Techne [42]	+	−	+	+	−	+	+	+
KOMBINE [43]	+	−	−	+	−	+	+	+
Contextual Goal Models [44]	+	−	−	+	−	+	+	+
Our approach	+	+	+	+	+	+	+	+

and inconsistencies in the changes on the context caused by execution of tasks [45]. They applied SAT techniques to discover inconsistencies.

8.2 Criteria based comparison with related works

In order to compare the goal-oriented approaches in product lines, we developed a set of criteria by adapting the criteria proposed in Mussbacher et al. [41] and Varela et al. [46]. *Feature model* and *goal model* criteria investigate whether the approach provides these two models. The *behavioral and Product line variability* criterion refers to discrimination of software and product line variability in goal models. The *Impact analysis, Conflict Identification* and *Stakeholders trade-offs* criteria explore if the impacts of features over intentions of stakeholders are addressed, if the conflict in the soft goals are identified and if reasoning over stakeholders' needs is preformed, respectively. Finally, the *Configuration* and *Goal Model and Feature Model Validation* criteria investigate whether the approach configures a feature model according to the requirements of stakeholders and checks consistencies of variability relations between goal models and feature models.

Table 5 illustrates differences between our approach and the existing goal modeling approaches in the context of software product lines.

Our approach mainly differs from other approaches in providing a comprehensive validation approach that takes potential changes in feature models, which can appear during the design of SPLs and validates whether variability relations in the feature model are aligned with the intentional relations in the family goal model. Additionally, we discriminate between product line variability and behavioral variability in goal models, which is neglected in most of the works except in [9,42]. However, it is crucial to consider the difference between these two types of variability when using goal models in the software product line domain. For conflict analysis, we rely on the GRL techniques, and through mapping, we can identify the impact of the features over tasks, and consequently, by using propagation algorithms in the GRL, we can see the impacts of the features on the higher-level objectives.

Other related work consider the detection of inconsistency in feature configurations, but without goal models. Czarnecki et al. [47] introduce an approach based on SAT solvers that detects violations of OCL-based well-formedness constraints of design models in feature configurations. Thaker et al. [48] extend even further this research direction and detect the absence of references in feature models to undefined classes, methods and variables. Janota et al. [49] and van der Storm [50] use propositional logic to validate the correctness of mappings between feature and component models. All of these approaches are used to detect inconsistencies between design models and feature models where inconsistencies are the result of the change in one model. In this work, we go a step further and validate inconsistencies between feature and goal models.

9 Conclusions and future work

As demonstrated in the paper, our contribution advances the state-of-the art in GORE for SPLs with a formal and correct validation approach for family requirements models. The proposed validation assures that intentional variability of an

SPL, captured in goal models, does not violate constraints of technical variability captured in feature models. The proposed validation approach compares the influence of intentional relations given by a goal model with feature relations from the feature model.

Both goal models and feature models successfully have been used in software engineering research and practice to capture stakeholders' needs and intentions [14,44,51] and managing product line variability [24,52,53]. Their combinations advance requirements engineering and configuration in the software product line context. Applying goals in software product line not only facilitates identifying features in domain engineering life cycle, but also ease the selections of features based on stakeholders' intentions and needs in application engineering life cycle. Several related works as well as our framework and tooling support demonstrate the practical applicability of the goal-oriented requirements engineering in the context of software product line engineering. Therefore, we believe that the results of our work will be useful for the development of professional tools.

In our future work, we will also describe the validation procedure for the further steps of product line configuration. First, we can validate if existing relations in the reference architecture models, in our research reference process models, are aligned with the relation defined in the family goal models. Second, we are going to develop an approach to ensure validity of customized process models with respect to relations in reference process models and feature models. Finally, we aim at extending the validation technique to cover other properties such as safe composition (i.e., for all application goal models, we have at least one feature model configuration and vice versa).

References

1. Pohl, K., Böckle, G., van der Linden, F.J.: Software Product Line Engineering: Foundations, Principles and Techniques. Springer, Berlin (2005)
2. Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, A.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Sofware Engineerin Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-90-TR-021. http://resources.seicmu.edu/library/asset-view.cfm?AssetID=11231 (1990)
3. Batory, D., Benavides, D., Ruiz-Cortes, A.: Automated analysis of feature models: challenges ahead. Commun. ACM **49**, 45–47 (2006)
4. Clarke, D., Proença, J.: Towards a theory of views for feature models. In: SPLC Workshops, pp. 91–98 (2010)
5. Acher, M., Collet, P., Lahire, P., France, R.B.: Separation of concerns in feature modeling: support and applications. In: Proceedings of the 11th Annual International Conference on Aspect-Oriented Software Development, pp. 1–12. AOSD '12, ACM, New York, NY, USA (2012)
6. Mylopoulos, J., Chung, L., Yu, E.: From object-oriented to goal-oriented requirements analysis. Commun. ACM **42**, 31–37 (1999)
7. Yu, Y., do Prado Leite, J.C.S., Lapouchnian, A., Mylopoulos, J.: Configuring features with stakeholder goals. In: Proceedings of the 2008 ACM Symposium on Applied Computing, pp. 645–649. SAC '08, ACM. New York, NY, USA (2008)
8. Yu, Y., Lapouchnian, A., Liaskos, S., Mylopoulos, J., Leite, J.: From goals to high-variability software design. In: Foundations of Intelligent Systems. Volume 4994 of LCNS, pp. 1–16. Springer, Berlin (2008)
9. Lapouchnian, A., Yu, Y., Mylopoulos, J.: Requirements-driven design and configuration management of business processes. In: 5th International Conference on Business Process Management, BPM, pp. 246–261. LNCS, Springer, Berlin (2007)
10. Asadi, M., Bagheri, E., Gasevic, D., Hatala, M.: Goal-driven software product line engineering. In: 26th ACM Symposium on Applied Computing (2011)
11. Metzger, A., Heymans, P., Pohl, K., Schobbens, P.Y., Saval, G.: Disambiguating the documentation of variability in software product lines: a separation of concerns, formalization and automated analysis. In: 15th IEEE International Requirements Engineering Conference, pp. 243–253. RE, IEEE (2007)
12. Mendonca, M.: Software product lines online tools (splot). http://www.splot-research.org/ (2012)
13. Yu, E., Giorgin, P., Maiden, N., Mylopoulos, J. (eds.): Social Modeling for Requirements Engineering. MIT, Cambridge, MA (2011)
14. Giorgini, P., Mylopoulos, J., Sebastiani, R.: Goal-oriented requirements analysis and reasoning in the TROPOS methodology. Eng. Appl. Artif. Intell. **18**, 159–171 (2005)
15. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering, vol. 5, 1st edn. Springer, Berlin (1999)
16. van Lamsweerde, A.: Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley, New York (2009)
17. ITU-T: Recommendation Z.151 (10/12): User Requirements Notation (URN) Language Definition, Geneva (2008)
18. Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R.: Reasoning with goal models. In: 21st International Conference on Conceptual Modeling (ER). Volume 2503 of LNCS, pp. 167–181. Springer, Berlin (2002)
19. Sebastiani, R., Giorgini, P., Mylopoulos, J.: Simple and minimum-cost satisfiability for goal models. In: CAiSE. Volume 3084 of LNCS, pp. 20–35, Springer, Berlin (2004)
20. António, S., Araújo, J.A., Silva, C.: Adapting the i* framework for software product lines. In: Proceedings of the ER 2009 Workshops on Advances in Conceptual Modeling—Challenging Perspectives, pp. 286–295. Springer, Berlin (2009)
21. Liaskos, S., Lapouchnian, A., Wang, Y., Yu, Y., Easterbrook, S.: Configuring common personal software: a requirements-driven approach. In: 13th IEEE International Conference Requirements Engineering, pp. 9–18 (2005)
22. Weiss, D.M., Lai, C.T.R.: Software Product-line Engineering: A Family-based Software Development Process. Addison-Wesley Longman (1999)
23. Liaskos, S., Litoiu, M., Jungblut, M.D., Mylopoulos, J.: Goal-based behavioral customization of information systems. In: Proceedings of the 23rd International Conference on Advanced Information Systems Engineering. CAiSE'11, pp. 77–92. Springer, Berlin (2011)
24. Kang, K.C., Kim, S., Lee, J., Kim, K., Shin, E., Huh, M.: Form: A feature-oriented reuse method with domain-specific reference architectures. Ann. Softw. Eng. **5**, 143–168 (1998)
25. Schobbens, P.Y., Heymans, P., Trigaux, J.C.: Feature diagrams: a survey and a formal semantics. In: Requirements Engineering, 14th IEEE International Conference, pp. 139–148 (2006)
26. Kang, K.C., Kim, S., Lee, J., Kim, K., Kim, G.J., Shin, E.: Form: A feature-oriented reuse method with domain-specific reference architectures. Ann. Softw. Eng. **5**, 143–168 (1998)

27. Liaskos, S., Lapouchnian, A., Yu, Y., Yu, E., Mylopoulos, J.: On goal-based variability acquisition and analysis. In: Requirements Engineering, 14th IEEE International Conference, pp. 79–88 (2006)
28. Czarnecki, K., Antkiewicz, M.: Mapping features to models: a template approach based on superimposed variants. In: GPCE'05. Volume 3676 of LNCS, pp. 422–437. Springer (2005)
29. Heidenreich, F., Kopcsek, J., Wende, C.: FeatureMapper: mapping features to models. In: Companion Proceedings of the 30th International Conference on Software Engineering (ICSE'08), pp. 943–944. ACM (2008)
30. Gröner, G., Asadi, M., Mohabbati, B., Gasevic, D., Parreiras, F.S., Boskovic, M.: Validation of user intentions in process models. In: 24th International Conference on Advanced Information Systems Engineering (CAiSE), pp. 366–381. Springer, Berlin (2012)
31. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook, 2nd edn. Cambridge University Press, Cambridge, MA (2007)
32. Wang, H., Li, Y., Sun, J., Zhang, H., Pan, J.: Verifying feature models using OWL. J. Web Semant. **5**(2), 117–129 (2007)
33. Kellner, M.I., Madachy, R.J., Raffo, D.M.: Software process simulation modeling: Why? what. J. Syst. Softw. **46**, 91–105 (1999)
34. Guo, J., Wang, Y., Trinidad, P., Benavides, D.: Consistency maintenance for evolving feature models. Expert Syst. Appl. **39**(5), 4987–4998 (2012)
35. White, J., Benavides, D., Schmidt, D.C., Trinidad, P., Dougherty, B., Cortés, A.R.: Automated diagnosis of feature model configurations. J. Syst. Softw. **83**(7), 1094–1107 (2010)
36. Liaskos, S., McIlraith, S.A., Sohrabi, S., Mylopoulos, J.: Representing and reasoning about preferences in requirements engineering. Requir. Eng. **16**(3), 227–249 (2011)
37. Mussbacher, G., Amyot, D., Araújo, J.a., Moreira, A.: Transactions on Aspect-Oriented Software Development vii. Springer, Berlin, pp. 23–68 (2010)
38. Silva, C.T.L.L., Alencar, F.M.R., Araújo, J., Moreira, A., de Castro, J.B.: Tailoring an aspectual goal-oriented approach to model features. In: SEKE, pp. 472–477 (2008)
39. Borba, C., Silva, C.: A comparison of goal-oriented approaches to model software product lines variability. In: Proceedings of the ER 2009 Workshops on Advances in Conceptual Modeling-Challenging Perspectives, pp. 244–253. Springer (2009)
40. Santos, L., Silva, L., Batista, T.: On the integration of the feature model and PL-AOVGraph. In: Proceedings of the 2011 International Workshop on Early Aspects, pp. 31–36 (2011)
41. Mussbacher, G., Arajo, J., Moreira, A., Amyot, D.: Aourn-based modeling and analysis of software product lines. Softw. Qual. J., **20**(3–4), 1–43. doi:10.1007/s11219-011-9153-8
42. Jureta, I., Borgida, A., Ernst, N., Mylopoulos, J.: Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling. In: Requirements Engineering Conference (RE), 2010 18th IEEE International, pp. 115–124 (2010)
43. Ernst, N.A., Borgida, A., Mylopoulos, J., Jureta, I.J.: Agile requirements evolution via paraconsistent reasoning. In: Proceedings of the 24th International Conference on Advanced Information Systems Engineering. CAiSE'12, pp. 382–397. Springer, Berlin (2012)
44. Ali, R., Dalpiaz, F., Giorgini, P.: A goal-based framework for contextual requirements modeling and analysis. Requir. Eng. **15**(4), 439–458 (2010)
45. Ali, R., Dalpiaz, F., Giorgini, P.: Reasoning with contextual requirements: Detecting inconsistency and conflicts. Inf. Softw. Technol. **55**(1), 35–57 (2013) (Special section: Best papers from the 2nd International Symposium on Search Based Software Engineering 2010)
46. Varela, P., Araújo, J.a., Brito, I., Moreira, A.: Aspect-oriented analysis for software product lines requirements engineering. In: Proceedings of the 2011 ACM Symposium on Applied Computing, pp. 667–674. SAC '11, New York, NY, USA, ACM (2011)
47. Czarnecki, K., Pietroszek, K.: Verifying feature-based model templates against well-formedness OCL constraints. In: GPCE, pp. 211–220 (2006)
48. Thaker, S., Batory, D.S., Kitchin, D., Cook, W.R.: Safe composition of product lines. In: Proceedings of the GPCE '06, pp. 95–104 (2007)
49. Janota, M., Botterweck, G.: Formal approach to integrating feature and architecture models. In: FASE. Volume 4961 of LNCS, pp. 31–45 (2008)
50. Van Der Storm, T.: Generic feature-based software composition. In: 6th International Conference on Software Composition. Volume 4829 of LNCS, pp. 66–80 (2007)
51. Amyot, D., Ghanavati, S., Horkoff, J., Mussbacher, G., Peyton, L., Yu, E.: Evaluating goal models within the goal-oriented requirement language. Int. J. Intell. Syst. **25**, 841–877 (2010)
52. Czarnecki, K., Eisenecker, U.W.: Generative Programming: Methods, Tools, and Applications. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA (2000)
53. McGregor, J., Muthig, D., Yoshimura, K., Jensen, P.: Successful software product line practices. Softw. IEEE **27**(3), 16–21 (2010)

Mohsen Asadi is a research assistant and Ph.D. candidate in Simon Fraser University (SFU) working on "Model-Driven Development of Families of Semantically enabled Service-oriented Architectures" project. He obtained his M.Sc in computer science (2009) from Sharif University of Technology (SUT), Iran. His main research interests are as follows: Software Product Line Engineering, Service Oriented Architecture, Requirements Engineering and Model-driven Development.

Gerd Gröner received his Ph.D. in 2011 from the University of Koblenz-Landau on the topic of "Process Model Management Using Description Logics". From 2011 until June 2013, he was senior researcher at the Institute for Web Science and Technologies at the University of Koblenz-Landau. Gerd Groener has a strong background in the application and development of Semantic Web technologies in software and systems engineering. Since July 2013, he is senior researcher at paluno - The Ruhr Institute for Software Technology at the University of Duisburg-Essen, Germany. His current research is on software architectures, software evolution, software adaptation, conceptual modeling and business process management.

Bardia Mohabbati received his M.Sc degree in computer science from Amirkabir University of Technology, Iran. He is currently a Ph.D. candidate in computer science and information technology at Simon Fraser University. His research mainly focused on service-oriented computing with a particular focus on engineering service-oriented software product lines, quality-aware software configuration and optimization. His research goal is to make enterprise service-based applications and business processes more flexible and adaptable to changing users' requirements.

Dragan Gašević is a Canada Research Chair in Semantic Technologies and an Associate Professor in the School of Computing and Information Systems at Athabasca University. He is also an Adjunct Professor in the School of Interactive Arts and Technology at Simon Fraser University, an IBM CAS Faculty Fellow, and an associated research member of the GOOD OLD AI Research Network at the University of Belgrade. His research interests include semantic technologies, software language engineering, technology-enhanced learning and service-oriented architectures. Being a principal investigator in several national and international research projects, he is a recipient of Alberta Innovates Technology Futures' 2008 New Faculty Award for his research on families of semantically enhanced business processes and service-oriented systems. A (co-)author of numerous research papers and often keynote speaker, he is the lead author of the book monograph Model Driven Engineering and Ontology Development published in two editions by Springer in 2006 and 2009. Committed to the development of international research community and the concept of cross-community engineering, Dragan has (co-)founded several new events (e.g., SLE and LAK), has served as a chair and member of numerous steering and program committees, and edited several journal special issues and books.