A Goal Oriented Approach to Identify and Configure Feature Models for Software Product Lines

Carla Silva[1], Clarissa Borba[2] and Jaelson Castro[2]

[1] Departamento de Ciências Exatas, Centro de Ciências Aplicadas e Educação, Universidade Federal da Paraíba (UFPB), Rio Tinto, Brazil
carla@dce.ufpb.br
[2] Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Recife, Brasil
ccb@cin.ufpe.br, jbc@cin.ufpe.br

Abstract. A goal oriented approach can be used as a way to discover variable and common requirements of a software product line (SPL), as well as to reduce costs associated with the configuration of a specific product in a product family. A goal oriented requirements engineering approach which has been used to develop complex system is the i* framework. It provides a manner to identify and specify the stakeholders' goals in relation to the intended system, as well as the characteristics of the system itself. This work proposes an extension of the i* modeling language, called **i*-c** (i* with cardinality), that allows inserting cardinality in some of its modeling elements. The G2SPL (Goals to Software Product Line) approach defines a process to identify and model common and variable features of a SPL using i* models with cardinality, as well as guides the configuration of a specific product in the SPL.

Keywords: Goal Oriented Approaches, Software Product Line, i* framework.

1 Introduction

Requirements Engineering (RE) can be applied both to develop unique software products and to develop software product lines (SPLs) [1]. A SPL, also called products family, is a set of software systems that share a common and managed set of features to satisfy specific need of a particular market segment [2]. A feature can be seen as a property or functionality of the system that is relevant to some stakeholders and used to describe variable and common features among products of the same family [3].

In the RE for a SPL, feature models are often used to capture similarities and variabilities of product families [4]. However, it is a great challenge to establish a relationship among the features of a software product and the stakeholders' goals. In particular, there is no systematic way to justify which features are going to be part of the SPL nor means to establish which features are optional, mandatory or alternatives. Besides, feature models are not appropriate for capturing the rationale, i.e., stating why a specific set of features should be selected to configure a specific product in a

SPL. On the other hand, Goal-Oriented Requirements Engineering (GORE) approaches are suitable to discover variable and common requirements of a SPL, as well as to reduce costs associated with the configuration of a specific product in a product family [5, 6, 7]. Indeed, some GORE approaches have been proposed to model requirements variability, such as Goals Model [5], PL-AO-VGraph [6] and Aspectual i* [7]. Recently, a comparison among these three approaches pointed that they have limited expressiveness to represent variability in SPLs [8]. To address some of the identified limitations obtained in this comparison, we proposed the G2SPL (Goals to Software Product Line) process that includes activities for (i) creating the i* models from scenarios specifications, (ii) identifying SPL features in the i* models, (iii) inserting cardinality in the identified features, (iv) deriving the feature model from the i* models, (v) reorganizing the feature model, (vi) configuring a specific product using the i* models, and (vii) obtaining the correspondent product configuration model. In previous work, we have presented the activities (ii), (iii) and (vi) of the G2SPL process, but we didn't present the activities (iv) and (v) [9]. This paper presents the whole process, but focuses mainly on these two activities in detail.

This paper is organized as follows. Section 2 introduces i* using, as example, the Mobile Media SPL. Section 3 overviews the i* with cardinality (i*-c), an extension that allows the representation of variability in SPLs. Section 4 describes the G2SPL process, which helps to derive and configure a feature model from i* models. Section 5 discusses related works and Section 6 summarizes our contributions and points out to some future works.

2 General View of the i* Framework

In the i* framework [10], stakeholders are represented as actors that depend on each other to achieve their goals, perform tasks and provide resources. Each goal is analyzed from its actor point of view, resulting in a set of dependencies between pairs of actors. The Strategic Dependency (SD) model provides a description of the relationships and external dependencies among organizational actors. While the Strategic Rationale model (SR) enable an analysis of how the goals can be fulfilled through contributions from the several actors. Throughout this paper we use the Mobile Media SPL [11] as a running example. The main purpose of the Mobile Media is to handle photos, music and video in mobile devices, such as telephones.

In the SR model presented in Fig. 1, we depict two actors (Mobile Telephone and User) and the dependencies between them. The main goal of the User actor is Media Handled. One way to achieve this goal is by performing the Add Photo task. Therefore, this task is linked to the goal through a means-end link. The Add Photo task is decomposed in four sub-tasks through a task-decomposition link. Observe that some (internal) sub-tasks (e.g. Select Album and Select Option of Adding Photos) may rely on external dependencies for their achievement. Similarly, some external dependencies (e.g. Photo, Accuracy [Path], Name) may depend on internal elements from the User actor to be fulfilled.

Observe, in the Mobile Telephone actor, the presence of contribution links (Help and Hurt). They are used to relate a task to a softgoal, indicating if the task

contributes positively (e.g., help) or negatively (e.g., hurt) towards the satisfaction of that softgoal. For example, the Save Automatically task contributes positively (Help) to the satisfaction of the Quickness [Storage] softgoal, while the Save by User task contributes negatively (Hurt) to the satisfaction of the same softgoal. Thus, we highlight two distinct forms of achieving the Photo Saved goal. Moreover, the means-end links from Save Automatically task and Save by User task to the Photo Saved goal indicate a variation point for the features of the SPL represented by the Mobile Telephone actor. In this case, two different Mobile Telephone products could be derived. One in which Photos are saved in a default location (automatically) and another in which the User has to define the location where the photo will be saved. A non-functional attribute such as Quickness [Storage] can be used to help configuring the best product for a given stakeholder's need.

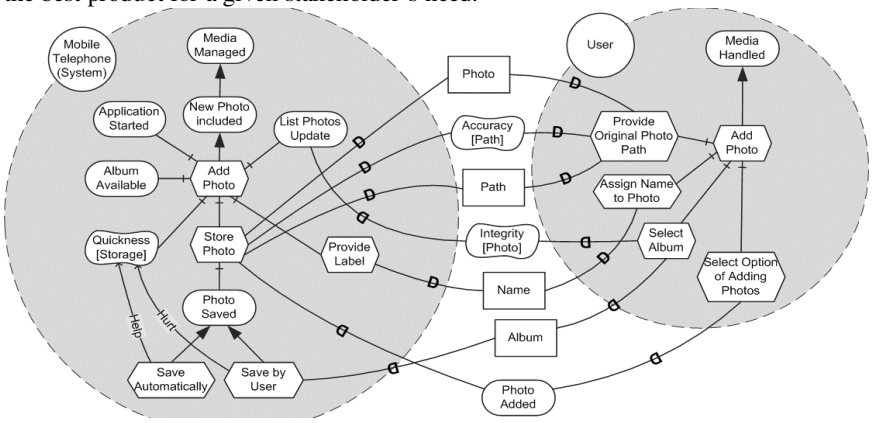


Fig. 1. SR model to Add Photo scenario.

The ability to capture alternative ways to achieve a goal, aligned with the ability to establish contribution links between functional (tasks) and non-functional (softgoals) requirements of a system, has motivated us to investigate the use of i* framework as a candidate technique for representing the variability and configuration information, i.e., select features for one or more products in a SPL [7, 8, 9]. However, the i* framework was not originally proposed in the context of Software Product Line Engineering. Therefore, it is required to adjust it to capture information related to the optional or mandatory characteristics present in features of a SPL.

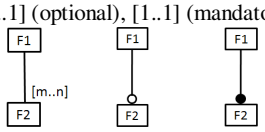
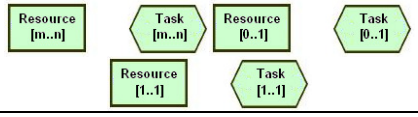
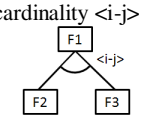
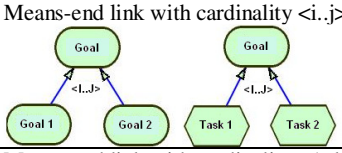
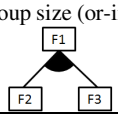
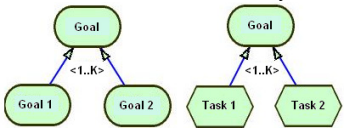
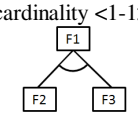
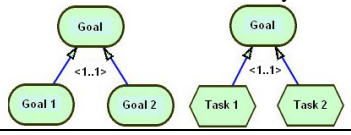
3 i*- c (i* with Cardinality)

In previous work, we proposed an extension of i* to include cardinality [9]. The extended language is called i*-c (it stands for i* with cardinality). It allows the representation of variability in SPLs by using the cardinality from the feature models presented in [12].

A feature can indicate any system characteristic or functionality and it is used to capture similarities or variabilities in products of a SPL [3]. From this definition, in

the i*-c language, elements of type task or resource are the intentional elements of the i* framework that presents the possibility of having cardinality, since they determine functionalities and characteristics of a system (see definition A in Table 1).

Table 1. Relation between feature model and i*- c elements.

	Feature Model	i*-c
(A)	Feature with cardinality [m..n], [0..1] (optional), [1..1] (mandatory) 	Resource and Task elements with cardinality [m..n], [0..1], [1..1] 
(B)	Grouped Features with cardinality <i-j> 	Means-end link with cardinality <i..j> 
(C)	Grouped Features with cardinality <1-k>, where k is the group size (or-inclusive) 	Means-end link with cardinality <1..k> 
(D)	Grouped Features with cardinality <1-1> (or-exclusive) 	Means-end link with cardinality <1..1> 

A means-end link, in the i* models, represents that a goal (the end) can be achieved through a sub-element from where the link starts (i.e., the means), representing an or-inclusive link (i.e., with cardinality <1-k>). However, to represent links of or-exclusive type (i.e., with cardinality <1-1>) and or-with-cardinality type (i.e., with cardinality <i-j>), we added cardinality to the means-end link in the i*-c language. Features obtained from the sub-elements (usually tasks), in this link, will be grouped in the feature model and the cardinality will belong to the relationship (see definitions B, C and D in Table 1).

The correspondence between the elements of cardinality-based feature models [12] and elements of the i*-c language is described in Table 1.

4 G2SPL Process

The G2SPL process, introduced in [9], has as its purpose to allow the identification and elaboration of a SPL's feature model from the i* models, as well as the configuration of specific products in the SPL. The process includes some activities

and heuristics that will be illustrated through its application to a use case (Add Photo) of the Mobile Media SPL. The complete modeling of this SPL, according to the G2SPL approach, can be found in [13].

The process consists of seven activities: Creation of the SR Model (optional), Identification of the Candidates Elements to be Features, Reengineering of the SR Model, Elaboration of the Feature Model, Reorganization of the Feature Model (optional), Configuration of the Product SR Model, and finally, Elaboration of the Product Configuration Model. The Domain Engineer role is in charge of the first five activities. The Configuration Engineer role is in charge of the activities Configuration of the Product SR Model and Elaboration of the Product Configuration Model. For the sake of space, we couldn't present the diagram representing the process, but an interested reader can find it in [9].

4.1 Creation of the SR Model

The first activity of the process in the Creation of the SR Model, that has as entry the use cases document of the system to be modeled. This activity can be considered optional if there is a SR model of the system under development. To perform this activity, we have taken as base the PRiM (Process Reengineering i* Method) approach [14] which is a method applied for business processes reengineering. We have adapted only the second phase of this method, since it refers to the creation of the i* SR model from scenarios. However, to create this model, PRiM uses its own technique for describing business scenarios, called "Detailed Interaction Script" (DIS). This technique is very similar to use case scenario description technique and, for this reason, we could adapt the second phase of PRiM method to create i* SR models from use cases' description.

The specification of the MobileMedia system is based on use cases. To illustrate the use of the G2SPL process, in this paper, we select the Add Photo use case. We highlight that the main purpose of this activity is to produce the SPL SR models in a systematic way. However, the requirements engineer is free to choose other ways to create i* models. We just suggest a method based on PRiM to present an alternative, based on use case specifications, for creating i* models systematically. The adaptation of PRiM method for this activity is detailed in [13].

4.2 Identification of the Candidates Elements to be Features

Once the SR model is obtained, the Domain Engineer must identify which elements from this model will be present in the feature model (activity Identification of the Candidates Elements to be Features). As discussed previously, a feature can indicate any characteristic or functionality of a system and it is used to capture similarities or variabilities in products of a SPL [3]. In our approach, features can be extracted from elements of type Task and Resource, since they determine system functionalities and characteristics, respectively.

As presented in section 2, SR models have an actor representing the system. In a SPL modeling, this actor will represent this SPL. Thus, only the actor's internal

elements and the dependencies (tasks or resources) directly linked to this actor are considered for the identification of features.

Using as entry the SR model presented in Fig. 1, we obtain, after performing this activity, the following result: each Task elements present inside the actor representing the SPL (Mobile Telephone) are highlighted as a feature candidate. They are: Add Photo, Store Photo, Provide Legend, Save Automatically, Save by User. Resource dependencies linked directly to the tasks of the Mobile Telephone actor (Name, Path, Photo and Album) are also highlighted as feature candidates.

Fig. 2 represents the artifact produced by this process activity: a SR model with the feature candidates highlighted in blue.

4.3 Reengineering of the SR Model

After identifying several features, this activity is concerned with restructuring the SR model generated from the previous activity to add cardinality to it. This restructuring is made through the application of some heuristics and by the definition of the i*-c language. The cardinality to be used in the i* models has the same meaning of those present in the feature models presented in [12], as presented in Table 1.

Having as base the application of the heuristics to the SR model generated in the previous activity, we obtain, as outcome of this activity, a SR model with cardinality (Fig. 2). Further details of this activity can be found in [9].

4.4 Elaboration of the Feature Model

This activity has as main purpose to derivate the feature model of a SPL and, to do this, it uses, as entrance artifacts, some heuristics, a table showing the mapping between i* model's cardinality and feature model's cardinality (Table 1) and the artifact generated from the previous activity (i.e., SR model with cardinality).

Feature models are composed of features organized in a hierarchical diagram similar to a tree, where each feature can be refined into sub-features. The feature model has a root that generally represents a concept (for example, a software system). This defines the first heuristic of this activity.

H1. The root feature of a feature model will be the system being modeled.

To elaborate the feature model, firstly we need to name the features based on the SR model. One intend to obtain features that represent more simple concepts. The way to obtain feature names is related to the properties that describe the intentional elements, instead of associating all the intentional elements description to a feature [15]. A straight manner to obtain the name of features through intentional elements of the i* models is described in the next heuristic.

H2. Feature names can be extracted through properties describing intentional elements. If the intentional element is a resource, the feature name is the same of the resource. If the intentional element is a task, the feature name can be a simplification of the task name, if it is necessary.

After naming the features, we create a table (Table 2) to insert some information to facilitate a creation of the feature model. This table has the following columns:

Element, Cardinality, FatherElement and Feature. The column “Element” must be filled with the name of the element in the i* model.

The column “Cardinality” is subdivided into two columns: “Type” and “Value”. Type field must be filled with the values “element” or “group” indicating if the cardinality pertains to the element or to the relationship, respectively. Value Field must be filled with the cardinality value.

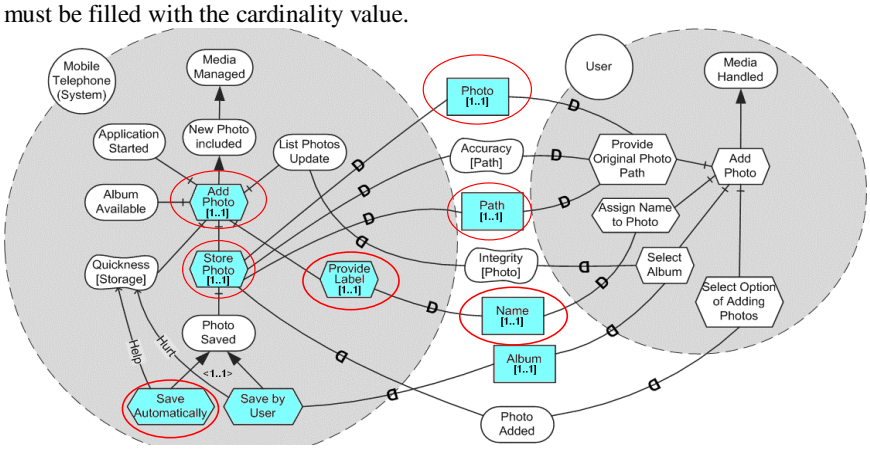


Fig. 2. SR Model for the use case Add Photo highlighting the feature candidates in blue and the chosen elements for a specific product configuration model circled with ellipses.

The column “Father Element” must be filled with the name of the element that has the element in question as a sub-element. However, (i) if the element’ “father” isn’t highlighted in the SR model as a candidate feature (for example, if the “father” is a goal), then one must choose the following element up in the hierarchy as the “father element”; (ii) if the element in question is not a sub-element, i.e., it is in the top of the tree, then the “Father Element” field if filled with the value “-”. This information will be useful to create the hierarchy of the feature models. The column “Feature” must be filled with the feature name according to the heuristic H2. This table must be filled with information of all candidate features. The following heuristics are related to the use of the information present in the Table 2 to create the feature model.

H3. All lines in the table will be mapped to the feature model with the name present in the column “Feature”. If two or more features have the same name, the information related to them will be united to become only one feature.

Firstly, the lines that do not have the element “Father Element” filled are going to be analyzed.

H4. All features that do not have the field “Father Element” filled will be related to the root feature, in the feature model.

H5. All feature in the table are sub-feature if the column “Father Element” is filled. These sub-features are related to the feature present in the field “Father Element”.

Relationships between features in the feature model are obtained from the column “Cardinality” of Table 2, as explained previously.

The root feature of the feature model will be the system being modeled, i.e., Mobile Media (H1). The names of the feature based on the resource elements Photo,

Path, Name and Album will be the same in the feature model. The names of the feature based on the task elements Add Photo, Store Photo, Provide Label, Save by User and Save Automatically are going to have the same names in the feature model (H2). Each line of the Table II is mapped to a feature with the name present in the column "Feature" (H3). According to Table 2, the feature Add Photo will be related to the root feature, i.e., Mobile Media, since it doesn't have the field "Father Element" filled (H4). Analyzing the lines of the Table II that have the field "Father Element" filled with the element identified in the previous heuristic (i.e., Add Photo), we find the features Store Photo and Provide Label (H5). We continue this analysis until all lines of the Table 2 are visited. The result is presented in Fig. 3.

Table 2. Information of the elements of the use case Add Photo that are going to be present in the feature model.

Element	**Cardinality**		**FatherElement**	***Feature***
	Type	**Value**		
Add Photo	Element	[1..1]	_	Add Photo
Store Photo	Element	[1..1]	Add Photo	Store Photo
Provide Label	Element	[1..1]	Add Photo	Provide Label
Save by User	Group	<1..1>	Store Photo	Save by User
Save Autom.	Group	<1..1>	Store Photo	Save Autom.
Photo	Element	[1..1]	Store Photo	Photo
Path	Element	[1..1]	Store Photo	Path
Name	Element	[1..1]	Provide Label	Name
Album	Element	[1..1]	Save by User	Album

4.5 Reorganization of the Feature Model

The activity Reorganization of the Feature Model is considered optional, since it will only be performed if it is necessary to reorganize the feature model, however, this activity can be performed as many times as the domain engineer thinks it is necessary.

There are several situations in which it is necessary to reorganize a feature model: sub-feature with more than one father, repeated features, features modeled in an improper location in the model, different features but with the same meaning, features than can be mixed with another because they refer to the same concept, etc. Depending on the situation, these features can be removed from the model or relocated or mixed into another feature. For example: (i) if a sub-feature is related to more than one feature, probably it will be relocated; (ii) if there are repeated features in the model but with the same meaning, repetition is eliminated; (iii) if there are different features different but with the same meaning, they will become only one feature. We didn't define heuristics for this activity, since the identification of these "bad smells" in feature models depends on the experience of the domain engineer.

We reorganize the feature model so that it could address the following two groups of operations: we created the features Photo Management and Album Management, and as theirs sub-features, we have placed the features Photo and Album, respectively. Doing this, we have removed the repeated features Photo that were related to the features representing Photo operations. Also, we have removed the repeated features Album that were related to the features representing Album operations (see Fig. 4).

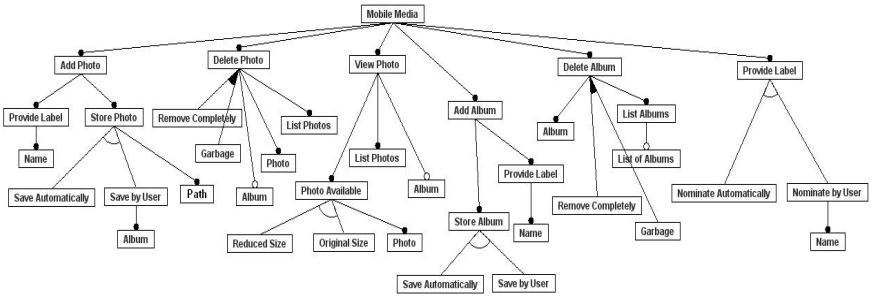


Fig. 3. Resulting Feature Model.

Continuing this analysis, we verify that the feature Provide Label is repeated as *sub-feature* in the group of operations related Photo and Album and as sub-*feature* of the root feature, **Mobile Media**. This repetition is unnecessary because in all occurrences it has the same purpose and the same cardinality (see Fig. 3). Thus, we removed that *sub-feature* from Add Photo and Add Album (see Fig. 4).

Observing Fig. 3, we still identify repeated features (**Save Automatically, Save by User, Album, Remove Completely, Garbage,** and **List Photos**). However, they shouldn't be removed because they are part of a specific product configuration (e.g. the configuration for the feature Remove Photo is independent of the configuration for the feature Remove Album).

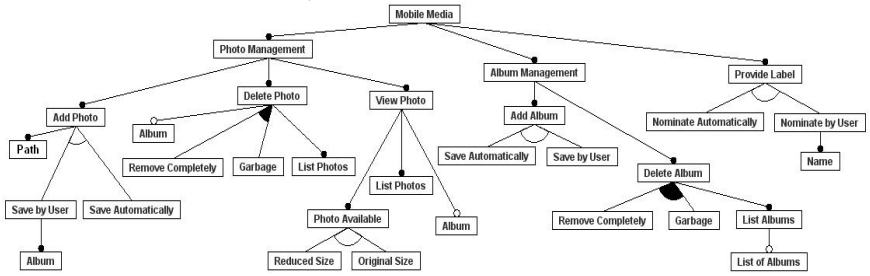


Fig. 4. Reorganized Feature Model.

4.6 Configuration of the Product SR Model

The activity Configuration of the Product SR Model has as purpose to select the candidate features that will be part of a specific product. It is from this activity that our approach starts to give support to the product development phase. This activity is repeated for each product to be configured in a SPL.

i* model is able to express the stakeholder's goals and the characteristics of the intended system by using intentional elements that illustrate the possible alternatives to be considered in a business process. This additional information, found in the i* models, complements the feature model, helping the choice and justification of a configuration for a specific software product. For example, according to Fig. 2, if

Quickness [Storage] is a softgoal, in the SR model, has a high priority for the stakeholder, probably the requirements engineering will select the candidate features that best satisfy this softgoal as, for example, the task Save Automatically.

In this phase, stakeholders must indicate which features will be part of the product. Besides this indication is subjective, softgoals in the i* models can help in choosing a specific configuration to make the software product to achieve the organization's goals. Some heuristics were defined concerning the hierarchy of the selected elements, their mandatory presence in the software product, and their contribution to satisfy stakeholders' preferences. These heuristics can be found in [9].

Performing this activity, we obtain the following result: The Add Photo, Store Photo, Provide Label, Photo, Path and Name elements were selected to be present in product configuration model, since they have cardinality [1..1]. We observe that the Album element, besides having cardinality [1..1], will not be present in product configuration model, because his parent element (Save by User task) was not selected. The cardinality of the Save Automatically and Save by User grouped elements is <1..1>, indicating the photo or will be saved by the system or by the user (or-exclusive). In this case, since only one alternative must be selected in the group, we choose to Save Automatically, because this task has a positive influence in relation to the softgoal Quickness [Storage].

4.7 Elaboration of the Product Configuration Model

The purpose of this activity is to build the configuration model for a specific product, from the SR model highlighting (using ellipses in Fig. 2) those elements that will be part of this product. For example, in Fig.2, we observe that the elements Add Photo, Store Photo, Provide Label, Save Automatically, Photo, Path and Name were highlighted to be part of the product configuration model valid for the Add Photo use case (not presented here). Further details about this activity can be found in [9].

5 Related Works

Some goal oriented requirements engineering approach for SPL have been proposed to capture the feature model semantics: goal models [5], PL-AOVGraph model [6, 16] and aspectual i* model [7]. However, a comparison between the approaches [5, 6, 7] has pointed that they have limited expressiveness, since they are not capable of representing the cardinality present in the feature model [8]. This limitation has motivated us to propose the i*-c language used by the G2SPL approach, in which we add cardinality, allowing to capture more information about the variability of a SPL.

Other approach based on goal-oriented models, more specifically in the i* models, is the IStarLPS [15]. This approach, similarly to the G2SPL, adapts the i* framework to software product lines, but with significant differences:

- The G2SPL approach uses information from the SR model to guide the configuration of a specific software product in a SPL, besides using it to identify features in the SPL. The IStarLPS approach also is based on the SR model to

identify features, but it does not guide the selection of features to configure a specific product;

- In the IStarLPS approach, all kinds of intentional elements of the SD and SR models are mapped to features, while the G2SPL approach considers as features only task and resource elements (see section 4.2) because they are the elements that represent functionalities and characteristics of a system;
- In IStarLPS, cardinality is included in all types of i* intentional elements, while, in G2SPL, we add cardinality only in those elements that can be a candidate feature, i.e., in tasks and resources, as well as in the means-end link (see section 4.2). In IStarLPS, when a goal element has cardinality [0..1], it represents the possibility of this goal be achieved of not in a product of the SPL, but it does not make sense to use the cardinality [m..n] in goals, since a goal is a state of affairs to be achieved (what does it mean to achieve a state of affairs at least m and at most n times?). Due to this observation, we chose to consider only resources and tasks as candidate features;

The G2SPL approach keeps in the i* model its original information, helping to choose and justify a configuration for a specific product through the contribution relationships between tasks and softgoals. In IStarLPS, besides softgoal elements were not considered in the approach, there is no justification when choosing a specific configuration for a product in the SPL. However, the IStarLPS guides the identification of requires constraints of the feature models [4] from i* models.

5 Conclusions and Future Works

This paper presented a goal oriented approach, called Goals to Software Product Line (G2SPL), that guides the identification of common and variable features in SPLs, as well as the configuration of a specific product in a SPL. Using a GORE approach, such as the i* framework, has brought various benefits to the engineering of SPL as, for example, tracing system features and stakeholder's goals and choosing a configuration for a specific product in the SPL based on the stakeholder's preferences.

In particular, this paper focused mostly in presenting two activities of G2SPL process: Elaboration of the Feature Model and Reorganization of the Feature Model, not detailed in [9]. These activities guides the elaboration of a feature model for a SPL obtained from goal-oriented models described using the i*-c (i* with cardinality) language [9]. The addition of cardinality in the i* model was able to represent variability in a SPL with improved expressiveness, making it possible to identify, from goal-oriented models, common and variable features and the relationships among them. G2SPL can be used in both reactive and proactive approaches for SPL.

Other contributions of the G2SPL approach are to keep the i* model with the original information, to help choosing and justifying the configuration of a specific product in the SPL, but enriched with cardinality, to help generating the feature model. It is worth noting that the contributions of tasks to fulfill softgoals aid the configuration knowledge activity by reducing time and costs to perform it.

As future works, we plan to: (i) perform case studies in different domains to evaluate, with improved accuracy, the strengths and weaknesses of the G2SPL

approach; (ii) develop a tool support for the G2SPL approach; and (iii) investigate how to identify feature model constraints from i* models.

Acknowledgments. This work was supported by CNPq and CAPES, Brazilian agencies for research support.

References

1. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering: Foundations, Principles, and Techniques", Springer (2005)
2. Clements, P., Northrop, L.: Software Product Lines: Practices and Patterns, Addison-Wesley (2002)
3. Czarnecki, K., Eisenecker, U. W.: Generative Programming: Methods, Tools, and Applications. ACM Press/Addison-Wesley Publishing Co (2000)
4. Kang, K. et al.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Software Engineering Institute, Technical report, CMU/SEI-90-TR-021 (1990)
5. Yu, Y. et al.: Configuring features with stakeholder goals. In: Proc. of the ACM Symposium on Applied Computing (SAC), pp. 645-649. ACM, Fortaleza, Ceará, Brazil (2008)
6. Batista, T. et al.: A Marriage of MDD and Early Aspects in Software Product Line Development. In: Early Aspects Workshop at 12th International Software Product Line Conference (SPLC'08), pp. 97-104, Limerick, Ireland (2008)
7. Silva, C. et al.: Tailoring an Aspectual Goal-oriented Approach to Model Features. In: Proc. of the 20th Intl. Conf. on Software Engineering and Knowledge Engineering (SEKE'08), pp. 472-477. Knowledge Systems Institute Graduate School, San Francisco, USA (2008)
8. Borba, C., Silva, C.: A comparison of goal-oriented approaches to model software product lines variability. In: Workshop on Requirements, Intentions and Goals in Conceptual Modeling (RiGIM'09) at 29th International Conference on Conceptual Modeling (ER'09), Gramado, Brazil, LNCS, Vol. 5833, pp. 244-253, Springer-Verlag, 2009.
9. Silva, C., Borba, C., Castro, J.: G2SPL: Um Processo de Engenharia de Requisitos Orientada a Objetivos para Linhas de Produtos de Software. In: 13th Workshop on Requirements Engineering (WER 2010), Cuenca, Equador (2010).
10. Yu, E.: Modelling Strategic Relationships for Business Process Reengineering, Ph.D Thesis, Dept. of Computer Science, University of Toronto, Canada (1995)
11. Figueiredo, E., et al.: Evolving Software Product Lines with Aspects: An Empirical Study on Design Stability. In: Proc. of the 30th Intl. Conf. on Software Engineering (ICSE'08), Leipzig, Germany, pp. 261-270 (2008)
12. Czarnecki, K., Helsen, S., Eisenecker, U.: Formalizing cardinality-based feature models and their specialization. Software Process Improvement and Practice, 10(1), pp.7-29 (2005)
13. Borba, C. : Uma Abordagem Orientada a Objetivos para as Fases de Requisitos de Linhas de Produtos de Software, Master Dissertation, Center of Informatics, Federal University of Pernambuco (2009) (in portuguese)
14. Grau, G. et al.: PRiM: An i*-based process reengineering method for information systems specification. Information and Software Technology 50, pp. 76-100 (2008)
15. António, S., Araújo, J., Silva, C.: Adapting the Framework i* for Software Product Lines. Workshop on Requirements, Intentions and Goals in Conceptual Modeling (RiGIM'09) at 29th International Conference on Conceptual Modeling (ER'09), Gramado, Brazil, LNCS, Vol. 5833, pp. 286-295, Springer-Verlag, 2009.
16. Fernandes, L., Batista, T., Soares, S., Santos, L.: On the Role of Features and Goals Models in the Development of a Software Product Line. In: Early Aspects Workshop at 9th Annual Aspect-OrientedSoftware Development Conference (AOSD'10), Rennes, France (2010).