Soft Constraints in Feature Models

Jorge Barreiros
Instituto Superior de Engenharia de Coimbra, Coimbra
Universidade Nova de Lisboa, Lisboa,Portugal
jmsousa@isec.pt

Ana Moreira
CITI/Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa, Lisboa, Portugal
amm@di.fct.unl.pt

***Abstract*—Feature Models represent admissible configurations of products in Software Product Lines. Constraints are used to represent domain specific knowledge, such as requiring or excluding a feature in the presence of another. Configurations failing to conform to these constraints are deemed invalid. However, in many cases useful domain information cannot be expressed comfortably with such forceful, hard constraints. Therefore, we propose the use of softer constraints of less forcing nature. We categorize possible semantics for such constraints, analyze their impact on the feature expression and describe some specific analysis procedures that are unique to the use of soft constraints.**

Keywords-Feature Models; Software Product Lines; Soft Constraints; Feature Consistency; Feature Interaction, Semantic Validation

I. INTRODUCTION

In opposition to traditional single system development, Software Product Line (SPL) development is concerned with the creation of families of software products. In SPLs, product variants belonging to the same family are created by specifying a feature configuration, which is then realized by the composition of corresponding artifacts from a common pool of assets (such as requirements documents, design models, code, etc.) [1].

Feature models are frequently used in SPL development for identifying valid product configurations, that is, configurations corresponding to a variant that can be created by an application engineer using the SPL [2]. Feature models identify valid configurations by using a feature tree annotated with additional domain constraints. These can be represented graphically (e.g., linking dependent features with a dependency arrow) or textually, by means of arbitrary cross-tree expressions (Boolean expressions depending on the configuration variables). Feature models can be represented using logic expressions according to well known transformations described in [3, 4]. A feature model expression is obtained by conjoining the feature tree expression with the domain constraints.

An example of a feature model can be found in Fig. 1, where *Sound*, *Keyboard* and *Screen* are mandatory subfeatures of the root feature node *Phone*, while *MP3Player* and *Camera* are optional subfeatures. *Polyphonic* and *Monophonic* are mandatory and alternative subfeatures of the *Sound* feature, and *Monochromatic* and *Polychromatic* are alternative subfeatures of the Screen feature. One domain constraint is represented: the *requires* arrow describes that selection of the *Camera* feature implies the selection of the *Color* feature.

Links such as the one connecting *Camera* and *Color* in Fig. 1 describe *hard constraints*. Any configuration that does not respect this constraint is invalid. It can be the case, however, that domain information is not comfortably representable using such strict constructs. For example, a situation can be considered where the overwhelming majority of configurations do indeed respect a certain restriction, but a few exceptions may exist. In this case, restrictions on admissible configurations cannot be as strict. A simple example will be the case of a default selection for a group of alternative selections: if the parent feature of such group is selected, then the preferred alternative configurations may be suggested.

We propose the use of *soft constraints*, of less forcing nature, in these situations. The concept of soft constraint has been described earlier in the context of probabilistic feature models [5]. Probabilistic feature models extend standard feature models by the addition of "soft" constraints that are associated with a degree of probability. These are often obtained as the result of a feature mining processes. We consider the use of a similar concept in in standard, deterministic feature models. This allows richer semantics to be represented in feature models, with advantages such as enhanced analysis and improved configuration support. An example of such a constraint in Fig. 1 would be "*Sound suggests Polyphonic*", expressing domain knowledge that indicates the more common sound configuration option. Naturally, soft constraints do not need to be restricted to parent-child features as described: other relations such as "*Monophonic suggests Monochromatic*" can be represented. This type of constraints can be useful for efficiently capturing useful domain information that might be lost otherwise, as it is usually absent in standard feature models. It can be used to good effect for multiple purposes, depending on the specific semantics that are adopted as described later, such as allowing interactive configuration tools to suggest configuration choices to the user.

 ISBN: 978-1-61208-165-6

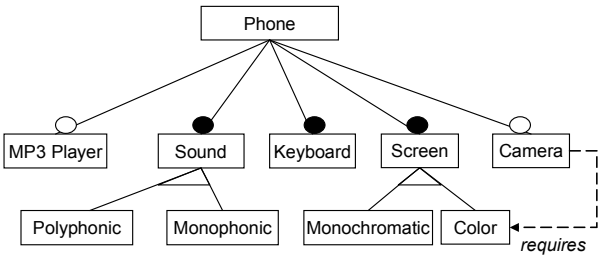

Figure 1. Mobile phone feature model.

Using soft constraints also allows some semantic consistency analysis that would otherwise be impossible, e.g., if a suggested dependency can never be realized in a feature model, then probably something is not right. Conflicting suggestions can also be found (e.g., multiple suggestions that cannot be satisfied simultaneously), highlighting that a trade-off analysis may be in order to compatibilize the inconsistent soft constraints.

The contributions of this work are the categorization of soft constraint semantics, the formalization of the impact (if any) of these constraints on the logic representation of the feature model and the description of automated analysis procedures made possible by the use of soft constraints.

In Section II, we present motivating examples for our work. In Section III, we discuss benefits of the use of soft constraints and propose a categorization of the different types of soft constraints. In Section I, we suggest a formalization and analysis techniques for detecting unsatisfiable and conflicting soft constraints. In Section V we present related work and we conclude in Section VI.

II. MOTIVATION

Consider the example in Fig. 2, adapted from [5], where a feature model is used to describe configuration variability for an automobile vehicle. In this case, hard domain restrictions are used to enforce the selection of manual transmission in sports vehicles and to make sure that emission control techniques are always used in products destined for markets with stricter environmental legislations. While observance of such constraints is always found in valid products, soft constraints are used to represent relevant relations between features that, while not as critical or universally applicable as the hard constraints, are also important. In this case, it is well known that the USA market tends to favor vehicles with automatic transmission over those with manual transmission, while the converse is true for the European market. Using soft constraints, such information can be readily represented in the feature diagram, bringing in additional semantics that can be used to good effect.

Another example of the use of soft constraints can be found in Fig. 3. In this case, the feature model is used to represent dynamic variability of the runtime behavior of a real-time system. The system should adapt its behavior to conform to variations in its environment. The state of the operation environment is assessed by appropriate sensors and the corresponding features are (de)selected accordingly, with corresponding impact on the runtime behavior as dictated by the constraints. A base control task is to be active at all times, while fan control is only suggested if the temperature is medium, but mandatory if it reaches a high level. A filtering task is suggested if electric noise is detected.

The need to use soft constraints to describe the variability in this scenario is supported by the fact that the suggested (non mandatory) features may not always be selected because of limited resources (e.g., available CPU load). This means that a feature such as *Fan Control* may in fact remain unselected in the presence of its suggestor (i.e., the *Noisy* feature), which cannot be comfortably expressed using only hard constraints.

These examples suggest that soft constraints can be used to good effect in feature models, by allowing the inclusion of important domain information of non-forcing nature.

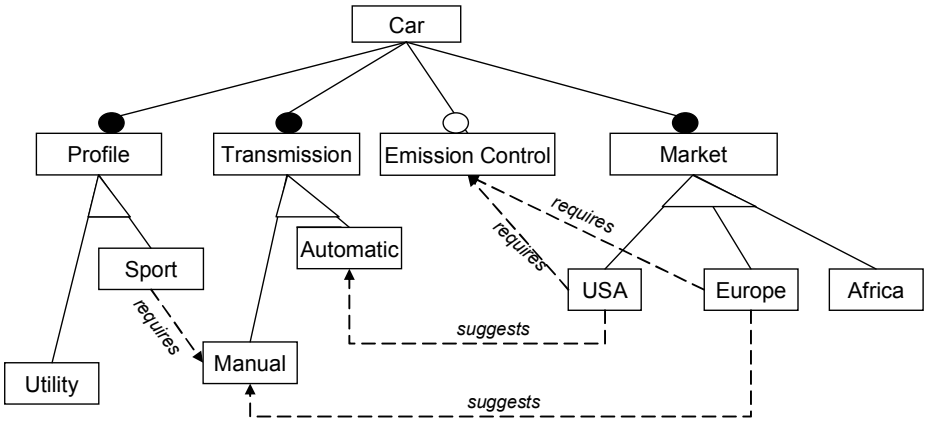

Figure 2. Feature model for car configuration

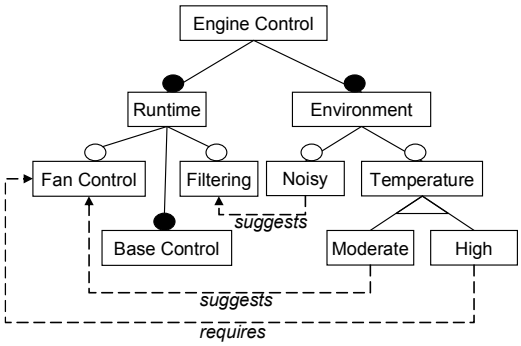

Figure 3. Engine control system

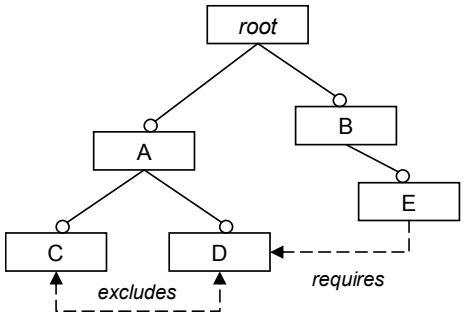

Figure 4. Iterative configuration example

III. SOFT CONSTRAINTS

In this section, we discuss the benefits gained by using soft constraints in feature models and present a categorization of alternative semantics.

A. Benefits

Benefits of soft constraints in feature models include:

- **Improved configuration support**: Interactive configuration and completion techniques can assist the configuration of feature models by assessing the liveliness of features after each configuration step. Starting from an empty configuration where all features are considered to be unspecified (neither selected or deselected), after a feature is selected or deselected by the user, the liveliness of all features is re-evaluated with respect to the partial configuration already defined. Features that are found to be dead (always unselected) in that partial configuration can be safely deselected automatically. Conversely, features that are common to all configurations that include the partial configuration so far specified can be automatically selected. For example, if the developer specifies feature *C* in Fig.4 to be selected, then features *D* and *E* can be automatically deselected by the configuration tool, as no valid configuration including feature *C* will contain either (i.e., both are dead in all configurations where *C* is selected). Similarly, *A* and *root* are common to all such configurations, so they can be selected automatically, leaving only feature B unspecified. Interactive configuration and completion tools can use soft constraint information to make configuration suggestions to the user. For example, if "A suggests B", the configuration tool can propose the selection of *B* by default whenever *A* is selected and B is unspecified. In the case of normative soft constraints, increased restrictions on admissible configurations also help to narrow down the correct configurations. Also, if a valid configuration fails to conform to a large percentage of soft constraints, it can be flagged to the developer as suspicious.
- **Improved semantic-oriented consistency checks**: Standard consistency analysis of feature models is concerned with ensuring that valid configurations do exist. If soft constraints are present, it is possible to make sure that configurations are available that verify the suggested dependencies. If that is not the case, this may be a sign that an analysis or modeling error has occurred. For example, if it was actually impossible to configure a car for the European market with manual transmission despite such association being suggested (e.g., because of the unintended side effect of some hard constraints), this would be highly suspicious and should be reported to the developer for additional consideration. This could be the case if hard domain restrictions would make it impossible to select a configuration where both such features are selected.
- **Controlled generalization of feature models**: A generalization of a feature model is a transformation that increases the number of admissible configurations, making sure that previously valid configurations remain valid. In some cases, soft constraints can be used as a mechanism for controlled generalization of feature models. For example, if it was found, after creating the feature model in Fig. 2, that it should actually be possible, under certain circumstances, to produce vehicles without emission control for the USA market, the hard restriction that forbids such products from being created could be transformed into an equivalent soft constraint. This would have the benefit of preserving important domain information while accommodating the need to allow for spurious "rogue" configurations.

B. Semantics and Categorization

Soft constraints can be interpreted according to different semantics, from unassuming configuration suggestions (e.g., describing a predominant configuration as in [5]) to stricter impositions that must be enforced if possible (i.e., a feature must be selected if possible). According to the adopted interpretation, different types of analysis and interpretations may be possible. Therefore, we must consider the possible semantics. These can be broadly categorized in two different categories:

- **Annotational**: A soft constraint with an annotational semantics does not impose any additional restriction when added to a feature model. Its main purpose is to embed domain information in the feature model to assist the configuration automation and semantic consistency checking. The validity of any specific product configuration is never influenced by the presence of an annotational soft constraint.
- **Normative**: A normative soft constraint must be considered when assessing the validity of a product configuration. These constraints represent configuration information that may potentially condition the validity of some configurations. A normative soft constraint must be satisfied if possible, but can be ignored otherwise. The concept of "possible satisfaction" is, generally, always dependent on the characteristics of the feature model and is also potentially dependent on domain-specific information (external to what is represented on the feature model: see below). A normative soft constraint may change the validity of a configuration (with respect to the unconstrained feature model), but it may never cause a feature model to become inconsistent. Normative constraints can be interpreted informally as meaning "requires-if-possible", "may-require", "require-if-does-not-make-configuration-invalid" or some other similar formulation.

Applying normative constraints entails the need to assess the "possibility" of selecting a specific feature. The topology of the feature model and cross-tree-constraints is always a decisive factor in making that assessment (i.e., it cannot be reasonably considered "possible" to select a feature when doing so would generate an invalid configuration). However, it may be the case that the feature model information is not sufficient to assess the possibility of selecting a feature: in this case, external factors, not represented in the feature model would come into play. This suggests the following characterization of normative constraints:

- **Internal:** The feature model holds all the information required to assess selection possibility.
- **External:** The information in the feature model alone is not sufficient for assessing possibility of selection. External factors come into play.

In the example of Fig. 2, if the soft constraints are interpreted under annotational semantics, then any configuration that upholds the hard constraints is considered valid, regardless of complying or not with the soft constraints. On the other hand, if an (internal) normative semantic is considered, the following interpretation holds: "If the *USA* feature is selected, then the *Automatic* feature must be selected, unless doing so would generate an invalid configuration". That is, a normative soft constraint should be interpreted as a hard constraint, unless doing so would turn an otherwise valid configuration into invalid. In Fig. 3, a potential example of external normative soft constraints is represented: in this case, the *Fan Control* feature should always be selected if the *Moderate* heat feature is selected, unless that is not possible, according to domain information that is not necessarily integrated in the feature model. For example, knowing that the implementations of the *Base Control*, *Fan Control* and *Filtering* features compete for a limited resource (CPU load), assessing of the possibility of including the *Fan Control* feature must be conducted with respect to external information. It is out of the scope of this work to discuss how such external information would be obtained or retrieved – as examples, an oracle could be used to provide the required information or a domain specific ontology could be queried.

Table I presents a summary of the characterization of hard and soft constraints.

TABLE I. SOFT AND HARD CONSTRAINTS CHARACTERIZATION

Nature	Subtype	Description	Affects FM consistency?	Affects config validity?	Semantics
Hard		*A requires B*	Yes	Yes	A =>B
		A excludes B	Yes	Yes	A => ¬B
Soft	Normative	*A may-require B*	No	Yes	Equivalent hard restriction should be upheld unless doing so would make the configuration invalid. May be further catgorized as "external" or "internal"
		A may-exclude B	No	Yes	
	Annotational	*A encourages B*	No	No	Measure of belief concerning the correlation between the configuration of both features.
		A discourages B	No	No	

 ISBN: 978-1-61208-165-6

IV. SOFT CONSTRAINT ANALYSIS

In this section, we present some formalization and analysis techniques specific for feature models with soft constraints. Although we propose a specific terminology for each different type of soft constraints in Table I, in the remaining text we use a link labeled "suggests" to indicate either "mayRequire" or "encourages" when the distinction is not important. For economy of space, exclusion-oriented constraints are not specifically discussed, but most results apply with minimal, usually obvious, adaptations.

A. Feature Expression for Normative Soft Constraints

Internal normative soft constraints may change the assessment of the validity of configurations with respect to the unconstrained feature model. This results in a change of the model expression when a new soft constraint is introduced in an existing feature model. The effect of inserting an internal normative soft constraint (A suggests B) results in a new feature model expression defined by:

$$F_S(A,B,...) = F(A,B,...) \wedge ((A \Rightarrow B) \vee \neg F(A,\neg B,...)) \quad (1)$$

where F is the feature model expression without the soft constraint and F_S is the resulting feature model expression.

An advantage of using internal normative soft constraints is that standard feature model techniques apply normally, e.g., satisfiability-based techniques are commonly applied to the analysis of feature model expressions [6], for tasks such as finding dead features This can be also done in a feature model annotated with soft constraints by considering the relevant F_S.

Equation (1) can be applied iteratively with respect to all soft constraints to obtain the feature expression corresponding to a feature model with multiple soft constraints. However, as described in Section IV.C, conflicting constraints may warrant additional care.

B. Unsatisfiable Constraints

Soft constraints can be used to include meaningful domain information in the feature model. One of the benefits this provides is the possibility of verifying if the feature model admits the existence of solutions that satisfy these soft constraints. That is, verifying if the feature model is *semantically* consistent with well known domain properties represented by soft constraints. If that is not the case, it is almost certainly an indication that an analysis error has been made and the feature diagram should be evaluated. This is not the same problem as the standard consistency assessment of a feature model as in that case we are only concerned with ensuring that at least one valid configuration exists. Consider the example in Fig. 5; in this case, because *B* and *C* are alternative features, it is not possible to find any configuration that conforms to the soft constraint suggestion. If the soft constraint represents a well known domain property, then it can be reasonably assumed that an analysis error has been made and that a re-evaluation of the feature model or the soft constraint might be advisable.

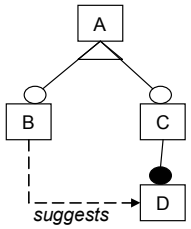

Figure 5. Unsatisfiable soft constraint

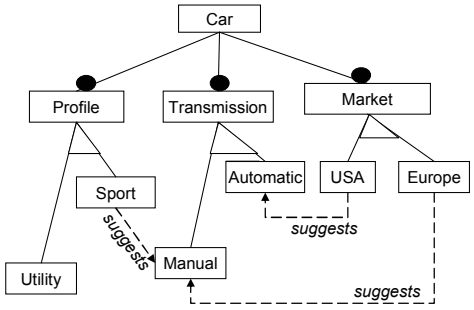

Figure 6. Conflicting soft constraints example

Unsatisfiable soft constraints can be identified by assessing the unsatisfiability of:

$$\neg(F_S(A,B,..) \Leftrightarrow F(A,B,...)) \quad (2)$$

Where F and F_S are defined as in (1). Unsatisfiability of (2) is indicative of an unsatisfiable soft constraint. Unsatisfiable constraint analysis can be performed not only with respect to normative constraints but also annotational ones. This is one of the advantages of including annotational soft constraints in feature models. Although these do not actually change the feature expression in any way, the same equations can be used for the purpose of constraint satisfiability analysis.

C. Conflicting Soft Constraints

Consider that, in the example of Fig. 2, after constructing the feature model, the developer finds that, although unusual, in some cases it may be necessary to allow configurations with the *Sport* profile and *Automatic* transmission. One way to handle this situation is to reduce the strength of the hard domain constraint that imposes *Manual* transmission for *Sport* vehicles by transforming it into a corresponding soft constraint. A partial representation of the resulting feature model is found in Fig.6.

It can be observed that simultaneous selection of the *USA* and *Sport* features will entail conflicting suggestions of transmission configuration. In such a situation, we describe the corresponding constraints to be conflicting. It is worth noting that this model is not inherently wrong as would be the case if hard constraints were involved.

The following procedure can be used to determine if soft constraints (A→B) and (C→D) will conflict when added to in a consistent feature model with expression *F(A,B,...)*:

1. Verify the satisfiability of $F(A,B,...) \wedge A \wedge C$. If it is not satisfiable, then no conflict exists.
2. If that is not the case, verify the satisfiability of $F(A,B,...) \wedge (A \Rightarrow B) \wedge (C \Rightarrow D)$. If it is not satisfiable, then a conflict exists.

When conflicting soft constraints are to be applied to a feature model, the order by which (1) is iterated to obtain the feature expression, as described in Section IV.A, is relevant to the outcome. Assuming all conflicting suggestions are of equal force, this is not desired and the following process should be used instead:

1. Identify all groups of conflicting soft constraints.
2. Iterate over all groups of conflicting soft constraints and compute:

$F_{S,n}(A,B,...) = F_{,n-1}(A,B,...) \wedge \bigvee_i ((A_i \Rightarrow B_i) \vee \neg F(A_i, \neg B_i,...))$

with $F_{s,0}(A,B,..) = F(A,B,...)$

This will create a feature expression where all conflicting suggestions are integrated. No preference is given to any suggestion over other, that is, in the example of Fig. 6, configurations with {*Sport*, *Manual*, *USA*} are just as admissible as {*Sport*, *Automatic*, *USA*}, If an interactive configuration tool was being used, {*USA*, *Sport*} were selected and both *Automatic* and *Manual* were unspecified, both of these features could be presented as configuration suggestions. Nevertheless, in some situations it may be desirable to perform a trade-off analysis and prioritize the relative importance of soft constraints. This would be the case if, for example, the *Sport* feature was a dominating factor on the choice of transmission. In this case, rather than following the process outlined above, (1) should be used instead, in order of the desired priority. That is, first consider the effect of the *Sport* feature on the feature model and only then compute the effect of the *USA* feature (on the previously computed feature model). This would allow for disambiguation of the suggestions represented by the soft constraints.

V. Related Work

In [5], probabilistic feature models are described that use soft constraints as descriptions of features that have high probabilities of being concurrently selected in the same configuration. Probabilistic feature models and corresponding samples spaces are suited to represent feature models obtained through feature mining processes. The fundamental purpose of probabilistic soft constraints in that context is to represent the results of the mining process. According to the classification in Section III.B, probabilistic soft constraints are inherently annotational, and as such do not affect the validity of any specific configuration, as is the case of the normalizing soft constraints we describe and analyze. We envision the use of soft constraints more as a fundamental construct of feature models, rather than being an auxiliary artifact.

"Encourages" and "discourages" constraints have been proposed for feature models in [7]. However, no precise semantics have been provided, precluding automated analysis and reasoning as described in our work.

In [8], fuzzy logic is applied to related feature configurations to costumer profiles. Fuzzy logic is a powerful tool for handling uncertainty. Nevertheless, normative semantics may be difficult to include in such an approach.

VI. Conclusions

We presented an exploratory analysis of the use of soft constraints in feature models. Possible semantics were specified and specific analysis techniques described. We found that soft constraints are useful in a diversity of contexts and offer the possibility of bringing additional important domain information to the feature model.

Future work includes application of soft constraints to well known industrial and academic case studies. Our prototype tool will be integrated with configuration tools providing enhanced configuration support.

VII. Acknowledgments

This work has been partially supported by the Portuguese Government through the PROTEC program grant SFRH/PROTEC/49834/2009 and by the Portuguese research centre CITI through the grant PEst-OE/EEI/UI0527/2011.

References

[1] P. Clements and L. Northorp, *Software Product Lines:Practices and Patterns*: Addison-Wesley, 2001.

[2] K. Czarnecki and U. Eisenecker, *Generative Programming: Methods, Tools, and Applications*: Addison-Wesley Professional, 2000.

[3] D. S. Batory, "Feature Models, Grammars, and Propositional Formulas," in *Software Product Lines, 9th International Conference, SPLC 2005* Rennes, France, 2005, pp. 7-20.

[4] K. Czarnecki and A. Wasowski, "Feature Diagrams and Logics: There and Back Again," in *11th International Software Product Line Conference (SPLC)* Kyoto, 2007, pp. 23-34.

[5] K. Czarnecki, S. She, and A. Wasowski, "Sample Spaces and Feature Models: There and Back Again," in *Software Product Lines, 12th International Conference, SPLC* Limerick, Ireland, 2008, pp. 22-31.

[6] M. Mendonça, A. Wasowski, and K. Czarnecki, "SAT-based analysis of feature models is easy," in *Software Product Lines, 13th International Conference, SPLC 2009*, San Francisco, California, USA, 2009, pp. 231-240.

[7] H. Wada, J. Suzuki, and K. Oba, "A feature modeling support for non-functional constraints in service oriented architecture.," *IEEE Computer Society,* pp. 187-195, 2007.

[8] S. Robak and A. Pieczynski, "Employment of fuzzy logic in feature diagrams to model variability in software families.," in *10th IEEE International Conference on Engineering of Computer-Based Systems (ECBS 2003)* Huntsville, AL, USA, 2003, pp. 305-311.

 ISBN: 978-1-61208-165-6