Conper: Consistent Perspectives on Feature Models

Julia Schroeter[1], Malte Lochau[2] and Tim Winkelmann[2]

[1] TU Dresden
Institute for Software- and Multimedia-Technology
julia.schroeter@tu-dresden.de
[2] TU Braunschweig
Institute for Programming and Reactive Systems
{m.lochau,t.winkelmann}@tu-bs.de

Abstract. Domain feature models express commonality and variability among variants of a software product line. For separation of concerns, e.g., due to legal restrictions, technical considerations, and business requirements, views restrict the configuration choices on feature models for different stakeholders. Our tool *Conper* allows to create views that obey different feature model semantics. We call such views perspectives on a feature model. We use a structured view model based on the Eclipse Modeling Framework (EMF) to define concern-relevant views as well as dependencies and hierarchies between views w.r.t. their concerns. Our tool supports the composition of views to create perspectives and offers an efficient algorithm to verify their consistency. The tool is applied in a staged configuration process for feature model preconfiguration. Additionally, as it is possible to define restricted perspectives per stakeholder, *Conper* supports customization on feature model level.

Keywords: Software Product Lines, Feature Models, Preconfiguration, Customization, Automated View Composition

1 Introduction

Feature models are used in software product line (SPL) engineering to express variability and commonality among product variants [3]. They specify the variant space. Due to various reasons the variant space is further restricted prior variant derivation. Reasons for this are driven by business concerns, e.g., to enable a variable pricing strategy for selling features in packages as well as by technical concerns, e.g., to identify a representative subset of variants for efficiently SPL testing (cf. [5]). It seems promising to express such concerns in a separate view model. Prior the derivation of a variant, concern-related views are selected and the domain feature model is filtered accordingly [6]. We call such a semantic preserving preconfiguration a *perspective* on the domain feature model. Various approaches to create views on feature models exist in literature [1, 2]. Though, these approaches focus on separation of concerns, whereas a particular view

ISBN 978-87-643-1014-6

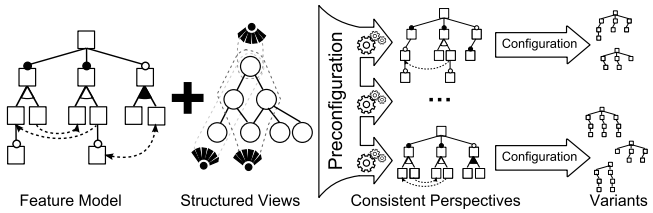

Fig. 1. *Conper* allows to define structured views on a feature model and create consistent perspectives by composing them.

is not intended to derive a complete variant, but rather to allow for specific configuration decisions only. Our tool *Conper* supports the definition of views on feature models in general and their aggregation to create consistent perspectives in particular.

The remainder of this paper is structured as follows. We briefly explain the concepts of perspectives on feature models in Sect.2. Sect. 3 describes the tool *Conper* followed by Sect. 4 which explains our findings and conclusion.

2 Consistent Perspectives on Feature Models

A consistent perspective imposes a specialization of the configuration space, i.e., a refinement of the original feature model semantics. A perspective is created by composing various views. A view represents a certain concern of the feature model. In our approach, a domain feature model and a view model are unified in a multi-perspective model, which imposes a conservative extension to the domain feature model. Due to the fact that only a valid subset of domain feature model variants is required to be derivable, not all possible view combinations form perspectives. Therefore, we use the concept of viewpoints to explicitly define valid view aggregations. The creation of a perspective is considered as a preconfiguration step in the variant derivation process as we show in Fig. 1. Furthermore, perspectives allow customization on feature model level in the way that stakeholder-specific features are only available to a particular stakeholder in the stakeholder's perspective.

Ensuring multi-perspective model consistency is, in general, hard to maintain due to the crosscutting nature w.r.t. the feature model, its cross-tree constraints and its potential overlapping of feature groups in a view model. Therefore, besides a comprehensive brute force approach, we developed an incremental heuristic for a scalable consistency verification of perspectives [6].

3 Conper: A Tool for Creating Consistent Perspectives

We implement *Conper* as plug-ins for the Eclipse Modeling IDE and extend the existing feature modeling and variant derivation environment *FeatureMapper* [4]. Further information, the source code as well as some example projects

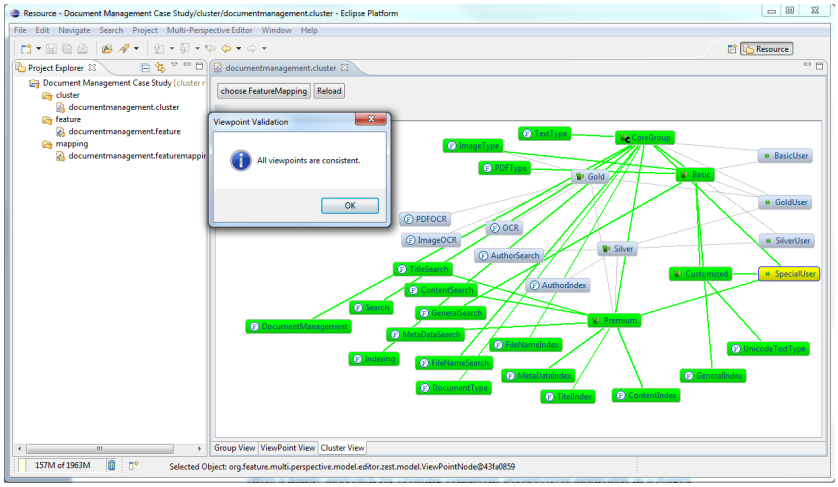

Fig. 2. Conper: Visualization of the relation between features and views.

and various screencasts of *Conper* are provided online[3], as well as an Eclipse update-site[4].

Our tool integrates the concepts of view models, feature models and mappings between them, as well as mechanisms to derive consistent perspectives. A *View Model* captures the relationships between views and viewpoints. We implement this concept using the Eclipse modeling framework (EMF). As the *FeatureMapper* offers an EMF-based feature meta-model that allows to create feature models with group cardinalities, we were able to seamlessly combine our approach with this environment. In addition, the *FeatureMapper* defines mappings between feature models in the problem space and EMF-based solution space artifacts. We reuse this functionality in our approach to create the assignments between features of the feature model and groups of the view model.

Conper provides various editors focussing on different aspects of the multi-perspective model. In Fig. 2, we show a screenshot of *Conper*. In the shown *Cluster View*, a viewpoint named "SpecialUser" is selected, and related view groups and features are highlighted. Furthermore, the Group View is a top down view, showing the view model starting from the core group. It is used during domain engineering to create view groups and viewpoints. In combination with the mapping view provided by the *FeatureMapper*, features are assigned to view groups in this view. Another Eclipse view, the *Viewpoint View* complements the *Group View* as it represents the view model bottom up, starting from a selected viewpoint and therefrom showing those groups the viewpoint is directly and indirectly assigned to. The *Cluster View* visualizes the mapping between feature model and the view model. It shows the assignment of features to view groups

[3] https://github.com/multi-perspectives/cluster/wiki
[4] http://juliaschroeter.de/conper/update

and which features belong to a certain viewpoint. Finally, we use the *Mapping View* provided by the *FeatureMapper* to assign features to view groups. By selecting a viewpoint in one of the editors, it is possible to derive a perspective. If the consistency check succeeds the corresponding perspective is derived by automated view composition and persisted in the Eclipse workspace stating a preconfigured feature model for subsequent product configurations.

4 Lessons Learned

The Eclipse IDE is a convenient platform to easily integrate new functionality. We used the frameworks EMF, EMFText and Zest as they offer powerful graphical and textual modeling capabilities. An issue we experienced with EMF is that our view model is a lattice graph as we support multiple inheritance relations among group views, whereas the EMF meta-model is tree-based. Therefore, some implementation effort was needed. One key finding of our experiments considering very large feature models with up to $10,000$ features is, that our heuristic consistency check algorithm is efficient as it scales well. Depending on the size of the view model, it identifies within milliseconds which viewpoints are consistent and which are not. In general, our experiences with various case studies show that perspectives are a promising concept for tailoring and customizing the variant space of a domain feature model to multiple stakeholders' concerns. Another finding is that there is a strong need for an appropriate visualization concept of the assignment of features in the feature model to groups in the view model. Our Zest-based cluster view provides a good overview of a complete mapping, but we need to support the mapping process graphically as well. In general, our tool *Conper* offers a valuable approach for creating consistent perspectives integrable in a staged configuration process with support for customization on feature model level.

Acknowledgements The presented work is co-funded by the European Social Fund, Federal State of Saxony and SAP AG in the research project #080949335.

References

1. Abbasi, E., Hubaux, A., Heymans, P.: A toolset for feature-based configuration workflows. In: Proceedings of SPLC'11 (2011)
2. Clarke, D., Proença, J.: Towards a theory of views for feature models. In: Proceedings of FMSPLE'10 (2010)
3. Czarnecki, K., Eisenecker, U.: Generative Programming: Methods, Tools, and Applications. Addison-Wesley (2000)
4. Heidenreich, F., Wende, C.: Bridging the gap between features and models. In: Proceedings of AOPLE'07 (2007)
5. Lochau, M., Oster, S., Goltz, U., Schürr, A.: Model-based pairwise testing for feature interaction coverage in software product line engineering. Software Quality Journal (2011)
6. Schroeter, J., Lochau, M., Winkelmann, T.: Extended version of multi-perspectives on feature models. Tech. Rep. TUD-FI11-07-Dezember 2011, TU Dresden (2011)