Kalpa Publications in Computing

Volume XXX, 2022, Pages 24–26

Proceedings of V XoveTIC Conference. XoveTIC 2022

kalpa Computing

Product configuration of a software product line using a Domain Specific Language

David de Castro[1], Alejandro Cortiñas[1], Miguel R. Luaces[1], and Óscar Pedreira[1]

Universidade da Coruña, Centro de Investigación CITIC, Laboratorio de Bases de Datos, Facultade de Informática, Elviña, 15071 A Coruña, Spain
{david.decastro,alejandro.cortinas,
luaces,opedreira}@udc.es

Abstract

A Software Product Line (SPL) reuses software assets to implement products that share a significant set of their features. When a developer needs to generate a new product, the selection of features determines which components and source code are assembled together as the product. In recent years, the Database Laboratory has been working with SPL technologies in the field of Geographic Information Systems (GIS). Our SPL creates products from a specification that allows, in addition to defining the data model of the application, to customize specific elements of the application such as *maps* and their associated *layers*. However, during its use with real projects we detected that this customization was insufficient: since the selected features are included for the whole product, if we need a feature only for a specific element, we need to apply it to all the elements of the same type. In this paper we propose a solution that, using a Domain Specific Language, allows to associate features with specific elements of the generated application in order to achieve a greater customization of the generated GIS and to improve their quality. This way, it is possible to select a feature (e.g., *clustering*) for a specific element (e.g., *map-viewer*), thus limiting the functionalities of the application to those parts where they are really necessary.

1 Introduction

In the Database Laboratory, we have created a Software Product Line (SPL) oriented to the generation of Geographic Information Systems (GIS), web applications that allow the visualization and management of geographic data of various kinds. In order to generate a GIS with the SPL, a specification is defined that has two main elements: a set of features to be included, and the data model that represents the domain of the application in question. In the first version of the SPL, the specification does not allow the configuration of certain parts of the product to be generated, such as maps, layers and styles. In order to allow to define them, a Domain Specific Language (DSL) was defined and integrated with the SPL [1], achieving a higher level of customization. Thus, the current specification of a product includes, in addition to the selected features and the data model, the definition of the set of maps, layers and styles that the application should have.

However, while the DSL allows us to create more customizable products, the fact that the feature selection is applied globally still presents a problem, especially when we generate complex or large GIS. As an example, a GIS product may have certain map views whose data must be displayed clustered (i.e., grouped by proximity), but other maps with less data must be viewed without clustering so that the individual data is visible. If the features are selected globally, it is impossible to have a product with both maps simultaneously.

In this paper, we present a solution to the problem modifying the nature of some features of our model, becoming part of the DSL and affecting the application locally. This change increases the level of customization supported by the SPL because it allows to customize certain functionalities and to link features with specific entities of the data model.

2 Our proposal

Figure 1 shows a fragment of the original SPL feature tree. Some of the features that can be selected are: *LayerManager*, which is used to have a layer manager in which style selectors, opacity, etc. will be arranged depending on whether the corresponding features are enabled; *Clustering*, which is used to group map elements that are in close proximity to avoid overlaps; or *OpacitySelector*, which adds the functionality in the *LayerManager* to change the opacity of a layer. If the *LayerManager* feature is selected, all maps in the application will have a layer selector regardless of whether they actually needed it. The same happens with *Clustering*, or with the *OpacitySelector* feature, among others.

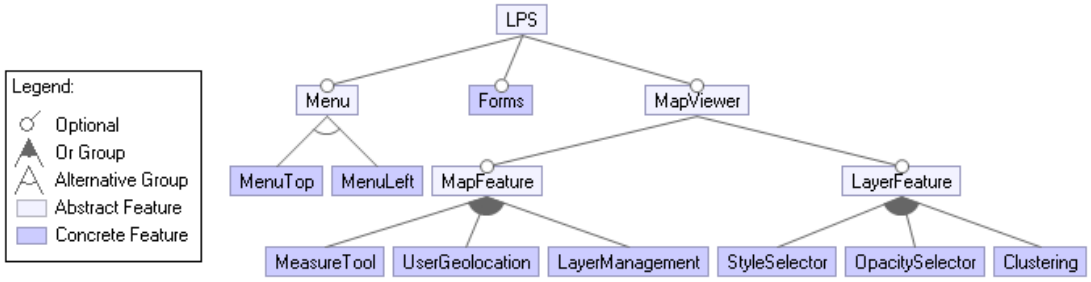

Figure 1: SPL's feature tree

On the right side of Figure 2 we can see the metamodel of the DSL that allows us to define the maps and the layers. In order to associate local features to each element (i.e. to each map and layer), a modification of the DSL must to be done. This modification, represented in the same figure, consists of associating to the entity *Map* the feature tree *MapFeatures*, which represents the features that can be applied to each map of the application. In the same way we have the model *LayerFeatures*, associated in this case to the entity *Layer*.

In order to support this change by the SPL, its code has to be adapted to take into account not only the selected global features, but also the local features associated with each element; to achieve this, it is necessary to generate control mechanisms in the code that only allow the user to access certain features when the instance of the class allows it, i.e., when the specific feature is associated with it. In this way, the correct integration of the features with the elements of the application is achieved, reaching a greater customization of the final product. This, together with the possibility of defining multiple different maps (thus eliminating the single map that contains all the geographical entities of the product) achieves a significant improvement in

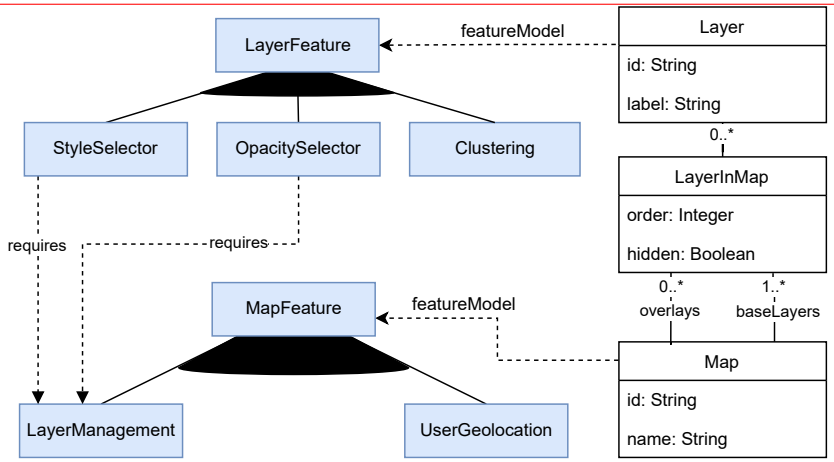

Figure 2: Example of the feature's integration in the DSL model

performance and a much higher level of customization than that achieved so far. This solution solves the problems described in the introduction, and it opens the possibility of associating features to specific elements, such as layers, thus eliminating unnecessary functionalities from those that do not require them. For example, it is now possible to define a map that clusters the geographic objects and another map that does not cluster the objects.

3 Conclusions and future work

The combination of Software Product Line and Model Driven Engineering technologies allows the semi-automatic generation of Geographic Information Systems with basic functionalities and features, although there are still certain points that need to be improved to reach a level of maturity that allows a higher degree of customization of the generated products. The changes proposed in this paper resolve these issues and help to create higher quality software, simplify the development of complex GIS and improve their performance.

As future work, all the changes proposed in this paper will be implemented. However, it is necessary to evaluate the effectiveness of these changes in a real context, so another future objective is the implementation of the solution developed for the generation of a real product, such as WebEIEL, which will allow us to verify the real effectiveness of these changes.

Funding: MCIN/AEI/10.13039/501100011033, NextGenerationEU/PRTR, FLATCITY-POC: PDC2021-121239-C31; MCIN/AEI/10.13039/501100011033 MAGIST: PID2019-105221RB-C41; MCIN/AEI/10.13039/501100011033 EXTRACompact: PID2020-114635RB-I00; GAIN/Xunta de Galicia/ERDF CEDCOVID: COV20/00604; Xunta de Galicia/FEDER-UE GRC: ED431C 2021/53; MICIU/FEDER-UE BIZDEVOPSGLOBAL: RTI-2018-098309-B-C32.

References

[1] Suilen H. Alvarado, A. Cortiñas, M. R. Luaces, O. Pedreira, and A. S. Places. Developing web-based geographic information systems with a dsl: proposal and case study. *Journal of Web Engineering*, 2020.