Scaffolding and in-browser generation of web-based GIS applications in a SPL tool

Alejandro Cortiñas
Universidade da Coruña
Database Lab
A Coruña, Spain
alejandro.cortinas@udc.es

Miguel R. Luaces
Universidade da Coruña - Database Lab
Enxenio
A Coruña, Spain
luaces@udc.es

Oscar Pedreira
Universidade da Coruña - Database Lab
Enxenio
A Coruña, Spain
opedreira@udc.es

Ángeles S. Places
Universidade da Coruña - Database Lab
Enxenio
A Coruña, Spain
asplaces@udc.es

ABSTRACT

The SME (Small and medium-sized enterprise) Enxenio has developed many web-based Geographic Information Systems within the last decade. Since the demand for GIS is increasing, Enxenio decided to apply Software Product Line Engineering to this domain to facilitate the development of complete web-based GIS applications, increasing their quality, improving the *time-to-market* and, at the same time, reducing their cost to its clients. This demo shows the resulting tool of this process, which is able to generate the source code of a web-based GIS from the set of desired features and the definition of its data model. This tool can be run within a web browser and the derivation engine in charge of generating the code is based on the *scaffolding* technique.

CCS CONCEPTS

• **Software and its engineering** → **Software product lines**; • **Information systems** → *Geographic information systems*;

KEYWORDS

Software product line engineering, web-based geographic information systems, scaffolding

ACM Reference format:
Alejandro Cortiñas, Miguel R. Luaces, Oscar Pedreira, and Ángeles S. Places. 2017. Scaffolding and in-browser generation of web-based GIS applications in a SPL tool. In *Proceedings of SPLC '17, Sevilla, Spain, September 25-29, 2017,* 4 pages.
https://doi.org/10.1145/3109729.3109759

This work has been funded by MINECO (PGE & FEDER) [TIN2016-78011-C4-1-R, TIN2016-77158-C4-3-R, TIN2015-69951-R, TIN2013-46238-C4-3-R, TIN2013-46801-C4-3-R]; CDTI and MINECO [Ref. IDI-20141259, Ref. ITC-20151305, Ref. ITC-20151247].

SPLC '17, September 25-29, 2017, Sevilla, Spain
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5119-5/17/09…$15.00
https://doi.org/10.1145/3109729.3109759

1 INTRODUCTION

Geographic Information Systems (GIS) [6] have been in continuous growth lately. With the improvements in communication technologies and the proliferation of high-speed Internet over the world, applications using GIS or location-based features are used every day by millions of users and in many contexts (general purpose applications, warehouse logistics, Internet of Things, Cyber Physical Systems, etc.). The development of a web-based GIS from scratch constitutes a very costly task, and many customers can only afford solutions with limited functionality instead of developing the appropriate one due to budget restrictions.

The SME (Small and medium-sized enterprise) Enxenio[1] is a leading provider of GIS at its region, Galicia. It has also been an important contributor at the Spanish national level for some years. Enxenio's close relationship with the Database Laboratory of the University of A Coruña has lead to close collaboration in many ways during the past years, as it can be seen in several previous publications [5, 12, 13].

For some time now, Enxenio has tried to define a Software Product Line for web-based Geographic Information Systems [3, 4, 8]. Reducing the *time-to-market* of new products and improving the quality of the software are some of the reasons motivating Enxenio to deal with this process, but the main one is to allow more customers to access to complete developments of web-based GIS by reducing their cost. In this work we show the tool resulting from the process detailed in [8], which allows for the automatic generation of web-based complete GIS applications implementing most of the features existing in this family of products.

In a previous attempt to develop a SPL for web-based GIS, shown in [3], we designed and implemented a tool *ad hoc* without following any proper methodology. This tool does not comply with the original requirements, and it ended up having a confusing user interface that mixes the feature selection task (Domain Engineering) with the configuration and parameterization of the products (Application Engineering). Furthermore, this tool focused more on providing lots of parameterization choices for standard web applications or non-GIS related features instead of providing the features initially required. Nevertheless, GISBuilder is still being

[1]http://www.enxenio.es

used at Enxenio as a prototype generation tool for standard web applications, far away from its original purpose.

In this ocassion we carried out a much more complete and exhaustive process [8], extracting GIS features not only from our expertise but also from existing products (both developed by Enxenio and others) and prioritizing the identified features in order to have an implementation planning. We also avoided non important details like most of the configuration of the previous tool (i.e., menu entries or user interface style) so the focus is on the GIS features.

2 OUR TOOL

This section describes a new tool to generate products of our SPL in the context of web-based GIS. We begin by explaining the requirements defined for the tool. In the following section, we describe the workflow for the generation of a new product, together with the description of the functional components of the tool. Finally, we detail the technology behind the tool.

2.1 Requirements

In the process carried out to define our SPL [8], we have established three requirements for the derivation of a product. Therefore, our tool must comply them.

First, it must allow the analyst to provide or define the data model for a product, because each web-based GIS works over a different domain. This has been already done in the past [9], but without providing the product any other variability, i.e., all the products provide the same features for different domains.

Next, the analyst shall select the desired features, and this selection must be validated. In case there is any problem, a message must be shown to the analyst. Feature models can provide a series of analysis operations [1], being the *valid product* operation the most important for the derivation process.

Finally, the derivation engine should generate the required code. A current web application uses many different languages both for the server side and the client side (e.g., Java, Javascript with its variants, HTML, CSS, XML, etc.). We also want to guarantee the right evolution of the platform (for example, in case a new language appears). For these two reasons, we consider that the annotative approach is the most adequate in this context, even though it has been historically criticized [10, 14] (lately this tendency is changing [2, 11]).

2.2 Workflow

As we can see in Fig. 1, our tool is conformed by two main components: a Web Interface, used by the analyst to define the products, and a Derivation Engine, invoked after the definition of the product has finished to generate the final source code of the product.

When the analyst wants to configure a new product, he or she uses the **Web Interface**, the application in charge of handling the whole process. In this application there are two different components:

- The **Data Model Definition UI** provides a graphical editor to design the data model using UML components: classes, enums and relationships among them. The classes defined are created in the product as entities, and their equivalent are created on the database as tables. When the different properties of the classes are defined, the type selection is restricted to the ones supported by the SPL. This is, the analyst cannot indicate the type manually, he or she can only select one of the allowed ones, which include the most common ones for alphanumeric data and dates. The analyst can also choose geographic types for these properties, creating geographic layers from entities.
- The **Feature Selection UI** shows the analyst a tree similar to the one provided by classic SPL tools such as Feature IDE (see Fig. 2). By default, most of the nodes of the tree are closed. When the analyst selects a node with children, these will be shown. In case the node has a mandatory child, this is also selected automatically. In case there is a problem with the selection, such as the analyst not selecting any child in an alternative feature or not fulfilling a cross-tree constraint [1], the interface provides a warning message.

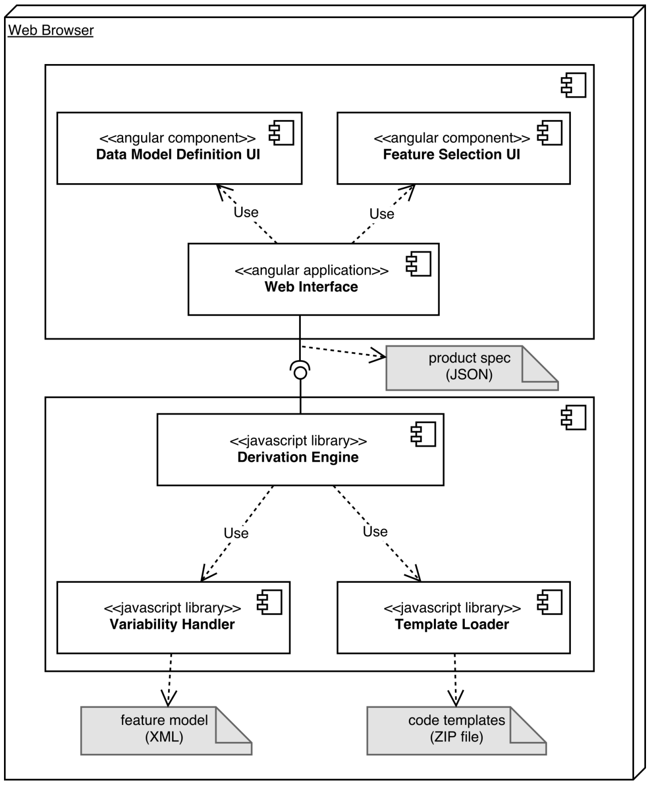
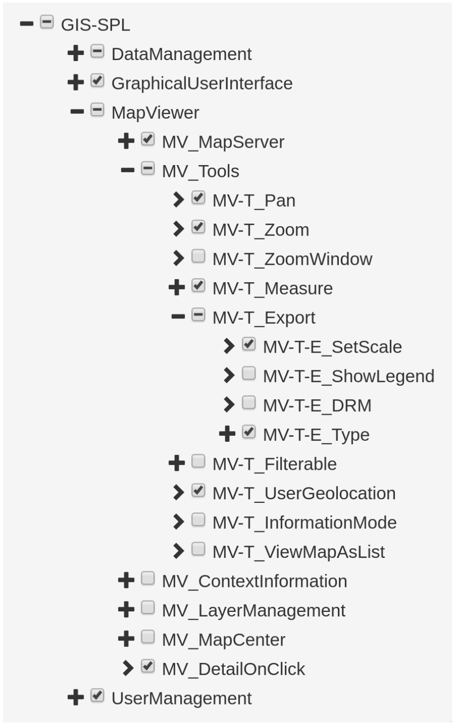


Figure 1: Component diagram of the tool

Once the analyst finishes the configuration of the product, the web interface invokes the derivation engine by using its API, and the generation of the final source code of the product begins. The **Derivation Engine** is the main component of the tool. It is responsible for generating the source code of a product from its specification. To generate the source code, it needs access to the annotated code (since it follows an annotative approach) through the **Template Loader**. Before the generation, the engine also checks whether the specification is valid using the **Variability Handler** and the *valid product* operation. The *feature model* of the SPL is



- ⊟ GIS-SPL
  - ⊟ DataManagement
  - ☑ GraphicalUserInterface
  - ⊟ MapViewer
    - ☑ MV_MapServer
    - ⊟ MV_Tools
      - ☑ MV-T_Pan
      - ☑ MV-T_Zoom
      - ☐ MV-T_ZoomWindow
      - ☑ MV-T_Measure
      - ⊟ MV-T_Export
        - ☑ MV-T-E_SetScale
        - ☐ MV-T-E_ShowLegend
        - ☐ MV-T-E_DRM
        - ☑ MV-T-E_Type
      - ☐ MV-T_Filterable
      - ☑ MV-T_UserGeolocation
      - ☐ MV-T_InformationMode
      - ☐ MV-T_ViewMapAsList
    - ☐ MV_ContextInformation
    - ☐ MV_LayerManagement
    - ☐ MV_MapCenter
    - ☑ MV_DetailOnClick
  - ☑ UserManagement

Figure 2: Feature model within the tool

loaded into the Variability Handler in order to do that. The derivation engine also provides the feature model to the Web Interface in order to draw it in the Feature Selection UI, as we can see in Fig. 2. The generated source code for the product is downloaded finally by the analyst as a zip file.

2.3 Technology

The annotative approach, when it comes to how it works, is pretty close to the technique of *scaffolding*. This technique is used to speed up the development processes through code generation. From some specifications provided by a software developer and a set of pre-defined templates (usually hidden to the developer), a lot of repetitive code can just be generated instead of writing it from scratch. As opposite to the code produced within a SPL, the code from scaffolding is suppose to be generated in the initial stages of a product development so it can be refined by the development team. Scaffolding was popularized by Ruby on Rails back in 2005, but it is currently used by many other frameworks such as Django, Yeoman or Spring Roo.

Our **Derivation Engine** is implemented as a Javascript library based on scaffolding. In order to generate every source file, a pre-defined template (in this case, an annotated source code file) is required, as well as the specification of the product describing how the template must be processed. An example of an annotated source code file can be seen in Fig. 3. The engine uses Javascript syntax embedded within comments to define the annotations. In the example we can see how different parts of a navigation bar are included depending on the particular features selected for the product. In this case, since the template is for HTML code, the annotations are within HTML comments. In case the template was for any other language, the annotations would be inside the specific language syntax for comments.

```
<ul class="nav nav-tabs">
    <!--% if (feature.csvImport) { %-->
    <li ui-sref-active="active">
        <a ui-sref='importTabs.csv' translate>tab.csv</a>
    </li>
    <!--% } %-->
    <!--% if (feature.spreadsheetImport) { %-->
    <li ui-sref-active="active">
        <a ui-sref='importTabs.spreadsheet' translate>tab.spread</a>
    </li>
    <!--% } %-->
    <!--% if (feature.shapefileImport) { %-->
    <li ui-sref-active="active">
        <a ui-sref='importTabs.shapefile' translate>tab.shape</a>
    </li>
    <!--% } %-->
</ul>
```

Figure 3: Annotated HTML source code

The **Variability Handler** is the component in charge of validating the feature selection made by the analyst. To do that, we first need to load the feature model of our SPL. We can do that by providing the feature model in two different formats, both of them allowing to define cross-tree constraints: as a JSON document following a schema designed by ourselves, or as a XML document with the same schema used by Feature IDE[2]. Therefore, we can design our feature model graphically with this tool and then take the XML file that represents it. The feature model can also be described programmatically using an API provided by the Variability Handler. The Variability Handler validates the feature selection made by the analyst in real time: as soon as the analyst selects one feature or another, it evaluates the feature model and the cross-tree constraints and returns the proper warning message. If this evaluation concludes that any other feature must be selected, such as a mandatory child feature or any other feature that should be selected due to the constraints, this information is also returned so the Feature Selection UI can act accordingly.

This library has been designed independently of the rest of the tool so it can be integrated in any other tool that requires feature model validation. Also, there are many other operations for feature models [1] that can be added in the future.

We can see an excerpt of our feature model in Fig. 4, showing some features that enable tools within the map viewer: *panning* and *zooming*, both of them mandatory; *measurement tools* to measure distances or map elements; *user geolocalization*, to locate the user position; or the feature to display a list from the elements currently shown in the map.

Regarding the annotated code source, it is provided as a ZIP file and it is handled by the **Template Loader**, which basically creates a filesystem and provides an API to the Derivation Engine for accessing the files. A similar approach is is used also to generate

[2]http://wwwiti.cs.uni-magdeburg.de/iti_db/research/featureide/

```
<and mandatory="true" name="MV_Tools">
    <feature mandatory="true" name="MV-T_Pan"/>
    <feature mandatory="true" name="MV-T_Zoom"/>
    <feature name="MV-T_ZoomWindow"/>
    <or name="MV-T_Measure">
        <feature mandatory="true" name="MV-T-M_Distance"/>
        <feature mandatory="true" name="MV-T-M_Line"/>
        <feature mandatory="true" name="MV-T-M_Polygon"/>
        <feature mandatory="true" name="MV-T-M_MapElement"/>
    </or>
    <feature name="MV-T_UserGeolocation"/>
    <feature name="MV-T_ViewMapAsList"/>
</and>
```

Figure 4: Feature model excerpt

the products, the Derivation Engine creates every output file within a virtual filesystem that is later exported as a ZIP file.

The **Web Interface** is built as an Angular[3] *Single Page Application*. Angular is a framework based on MVC pattern that favours modularity on applications by using Inversion of Control to load the different components. In our case we have two Angular components: the **Data Model Definition UI**, implemented with *JointJS*[4], where the analyst can design a class diagram with entities, properties, relationships, etc.; and the **Feature Selection UI**, implemented with *angular-ivh-treeview*[5], used by the analyst to make the selection of the features for every product.

The products to be generated are specified in JSON format. The design made in the Data Model Definition UI is converted into a JSON document with a specific schema, defined with JSON Schema[6], so it can be easily validated. This JSON document is completed with the selection of features, represented simply as an array with the identifiers of the features.

All the technologies used to implement our tool are based on web standards: Javascript, HTML and CSS. Therefore, our tool can be run entirely within a web-browser, without interaction with a server apart from getting the resources. That is, a basic web server without any configuration is enough to run it, and therefore it can be hosted virtually anywhere. This idea comes from an upgrade made to our previous tool, described in [7].

3 CONCLUSIONS AND FUTURE WORK

In this paper we have presented a new tool based on Software Product Line Engineering for the generation of web-based Geographic Information Systems. The tool is very portable, since it can run within a web browser without any server infrastructure. The tool also provides an analyst with an exhaustive set of features for generic web-based GIS applications and it validates the specification made by the analyst using the feature model of the SPL. As future work, we want to add *runtime live previewing* capabilities, as in [7], but in this case it will be more complex due to the more advanced and complete set of GIS features. We also want to add project management capabilities and to apply continuous integration techniques to facilitate the deployment of the products.

[3]https://angularjs.org/
[4]https://www.jointjs.com/opensource
[5]http://ivantage.github.io/angular-ivh-treeview/
[6]http://json-schema.org/

REFERENCES

[1] David Benavides, Sergio Segura, and Antonio Ruiz-Cortés. 2010. Automated analysis of feature models 20 years later: A literature review. *Information Systems* 35, 6 (sep 2010), 615–636. https://doi.org/10.1016/j.is.2010.01.001

[2] Thorsten Berger, Steven She, Rafael Lotufo, Andrzej Wasowski, and Krzysztof Czarnecki. 2013. A Study of Variability Models and Languages in the Systems Software Domain. *IEEE Transactions on Software Engineering* 39, 12 (dec 2013), 1611–1640. https://doi.org/10.1109/TSE.2013.34

[3] Nieves R Brisaboa, Alejandro Cortiñas, Miguel R Luaces, and Oscar Pedreira. 2016. GISBuilder: a framework for the semi-automatic generation of web-based geographic information systems. *Proceedings of the 20th Pacific Asia Conference on Information Systems (PACIS 2016)* (2016).

[4] Nieves R Brisaboa, Alejandro Cortiñas, Miguel R Luaces, and Matías Pol'la. 2015. A Reusable Software Architecture for Geographic Information Systems based on Software Product Line Engineering. In *Proceedings of the 5th International Conference on Model & Data Engineering (MEDI 2015)*, Vol. 9344. Springer, 320–331. https://doi.org/10.1007/978-3-319-23781-7_26

[5] N R Brisaboa, J A Cotelo-Lema, A Fariña, M R Luaces, J R Parama, and J R R Viqueira. 2007. Collecting and publishing large multiscale geographic datasets. *Software: Practice and Experience* 37, 12 (oct 2007), 1319–1348. https://doi.org/10.1002/spe.807

[6] Peter A. Burrough and Rachael A. McDonnell. 1999. Principles of Geographical Information Systems. (1999), 422 pages. https://doi.org/10.2307/144481

[7] Alejandro Cortiñas, Carlo Bernaschina, Miguel R Luaces, and Piero Fraternali. 2017. Improving GISBuilder with Runtime Product Preview. *Proceedings of the 17th International Conference on Web Engineering (ICWE 2017)* (2017).

[8] Alejandro Cortiñas, Miguel R Luaces, Oscar Pedreira, Ángeles S Places, and Jennifer Pérez. 2017. Web-based Geographic Information Systems SPLE: Domain Analysis and Experience Report. *Proceedings of the 25th International Systems and Software Product Line Conference (SPLC 2017)* (2017). Pending publication.

[9] Paule-Annick Devoine, Bogdan Moisuc, and Jerome Gensel. 2012. GENGHIS: an Environment for the Generation of Spatiotemporal Visualization Interfaces. In *Innovative Software Development in GIS*. 121–150.

[10] J.-M. Favre. 1997. Understanding-in-the-large. *Proceedings Fifth International Workshop on Program Comprehension. IWPC'97* May (1997). https://doi.org/10.1109/WPC.1997.601260

[11] Christian Kästner, Sven Apel, and Martin Kuhlemann. 2008. Granularity in software product lines. In *Proceedings of the 30th international conference on Software engineering - ICSE '08*. ACM Press, New York, New York, USA, 311. https://doi.org/10.1145/1368088.1368131

[12] Miguel R Luaces, David Trillo Pérez, J Ignacio Lamas Fonte, and Ana Cerdeira-Pena. 2009. An Urban Planning Web Viewer Based on AJAX. In *Web Information Systems Engineering - WISE 2009 (Lecture {Notes} in {Computer} {Science})*, Gottfried Vossen, Darrell D E Long, and Jeffrey Xu Yu (Eds.). Springer Berlin Heidelberg, 443–453. http://link.springer.com/chapter/10.1007/978-3-642-04409-0

[13] Ángeles S. Places, Nieves R. Brisaboa, Antonio Fariña, Miguel R. Luaces, José R. Paramá, and Miguel R. Penabad. 2007. The Galician virtual library. *Online Information Review* 31, 3 (jun 2007), 333–352. https://doi.org/10.1108/14684520710764104

[14] Henry Spencer and Zoology Computer. 1992. # ifdef Considered Harmful , or Portability Experience With C News. *Usenix* (1992), 185–198.