# Improving GISBuilder with Runtime Product Preview

Alejandro Cortiñas[1], Carlo Bernaschina[2], Miguel R. Luaces[1], and Piero Fraternali[2]

[1] Universidade da Coruña, Databases Laboratory, A Coruña, Spain
{alejandro.cortinas, luaces}@udc.es
[2] Politecnico di Milano, DEIB, Milan, Italy
{carlo.bernaschina, piero.fraternali}@polimi.it

**Abstract.** Software product lines allow users with little development experience to configure and generate applications. On the web this approach is becoming more and more popular due to the low time required to bring a new release to the final users. The architecture of web applications though require complex development environments in order to allow users to test and evaluate a new configuration. In this work we present a novel approach, based on in-browser generation and emulation techniques, which can be applied to real-world state of the art software product lines, reducing test deployment complexity and enabling an agile development cycle.

**Keywords:** software product lines, agile software development, rapid prototyping, geographic information systems

## 1 Introduction

Software Product Lines Engineering is a discipline that tries to apply industrialisation techniques, such as mass-customisation and reusing strategies, to software development with the focus on improving quality of products and, at the same time, decreasing costs and time-to-market of new products [1]. Developing costs of a SPL are compensated from the third product built [1], so their application is very convenient for software development companies that usually create similar systems, i.e., a product family. This is the case of Enxenio[3], a Spanish SME (small and medium-sized enterprise) with a certain grade of expertise in GIS [2,3,4]. In a previous work [5], we have approached the design of a SPL for the automatic generation of web-based GIS applications for its usage at Enxenio as an internal tool. GISBuilder provides a web interface where an analyst can design and generate the source-code of web-based GIS. The generated applications must be deployed afterwards within a Java web server. From the point of view of *agile* development, this is far from optimum since it requires a full redeployment for any small change that the analysts want to try in a product. [6] describes a

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

totally different approach, illustrated with a web tool able to design, generate and run web/mobile applications, doing it all within a web browser.

Following the mentioned approach, we have adapted GISBuilder web specification interface to provide the analyst a run-time preview of the product he or she is designing. In the demo, we will use the adapted version of GISBuilder to design, preview and generate some simple web-based GIS applications.

## 2 GISBuilder

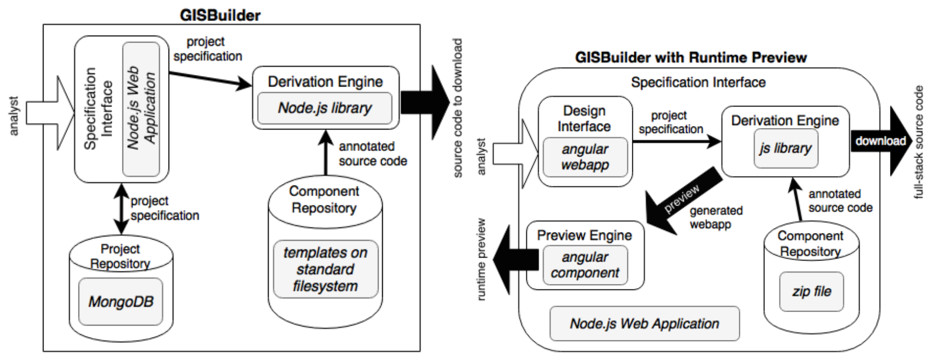
GISBuilder architecture is shown in Fig. 1a. We describe it thoroughly in [5], but we can summarize its workflow as follows. When a new application has to be created, an analyst interacts with the *specification interface*, a Node.js Web Application (more specifically, an Express app). In this interface he or she can decide: 1) Which features the application provides. Examples of features are *csv importer* or *user management*. 2) The data model for the application: entities, properties and relationships. The analyst can define lists, forms and maps from this data model that can be then linked through menu items. 3) Several aspects of the graphical user interface, such as the menu configuration, the static pages or the theme.

GISBuilder is not a traditional SPL because its capabilities are enhanced through the usage of a *scaffolding-based derivation engine*, able not only to assemble static software assets common to every product (the *features*) but also to generate product-specific code (lists related to data model or specific menu configuration for an application). This **derivation engine** is invoked with the product specification, which is nothing but a JSON document that complies with a JSON Schema, when the analyst finishes the configuration of the product. The product specification is also stored in the **project repository**, a MongoDB instance. The **derivation engine** takes the product specification and assembles/generates the source code of the final products getting the required components/templates from the **component repository**. Then, the product source code is generated and the analyst gets it as a downloadable zip file.

In order to deploy and try the generated product, the analyst needs a computer with some previously installed software: Node.js, npm or yarn, Java 8 and PostgreSQL with PostGIS. Then, he or she needs to run some processes to download all the dependencies and compile the product, and also to modify a text file to set the database connection configuration. These steps are very easy to follow for a developer, but i) it is still somehow slow to do that for every little change on the product configuration and ii) one of the goals of GISBuilder is that the analyst does not need to have any advanced knowledge on IT.

## 3 GISBuilder with Runtime Live Preview

In [6] we have presented a methodology for Rapid Prototyping in a Model-Driven Development context. IFMLEdit.org, a prototype created to validate this approach, is a web tool that allows the user to design web applications



(a) Original architecture (b) Adapted architecture

Fig. 1: Architecture changes in GISBuilder

using IFML [7], and to try them directly in the web browser before downloading them. If the user makes a change in the design, he or she can relaunch the application within the web browser and examine the change at runtime.

We have applied a similar approach on GISBuilder. This is, we have made GISBuilder able to generate and show a preview of the designed products at runtime, directly on the browser, without the need of any server-side structure. To achieve that, we have made several changes on the architecture of the SPL, as we can see in Fig. 1b. The biggest difference with respect to IFMLEdit.org is that its web applications are just prototypes, and the previewing engines are insufficient to deal with the complex single web page applications created by GISBuilder. The main changes made are:

1) We have implemented a new version of the **derivation engine** able to run entirely on the web browser. Instead of getting the component templates from the file system, they are loaded from a zip file. Similarly, it can generate the products in memory and provide a zip file with them. The **derivation engine** is the component analogous to IFMLEdit.org's *transformation engine*.
2) The **derivation engine** is integrated within the **specification interface**, as well as the **component repository**.
3) In IFMLEdit.org there are two components to preview applications: one for thin-client web-applications and another for web based thick-client mobile applications. To address for the advanced architecture of GISBuilder we have fused and extended these components, introducing full support for AJAX requests on which advanced frameworks, like Angular, are based. The component is able to load a dynamic webapp in an iframe, which intercepts XHR requests and returns mock responses to each specific REST petition.
4) GISBuilder produces full-stack web applications with Spring in the server side and Angular in client side. In the adapted version, GISBuilder creates two different versions of the products, depending on whether the analyst wants to preview them or to download the full-stack version.
5) To make the tool runnable without any deployment structure, the **project repository**, a MongoDB instance, is now optional. Since project specifica-

tions are just JSON documents, GISBuilder now provides features for downloading and uploading them. Therefore, the tool is totally portable.

The new workflow is as follows. An analyst interacts with the **design interface** to specify a new application. If the analyst wants to preview the application in its current status, the **derivation engine** takes the product specification and the zip file with the component templates and generates a *in-memory* zip file with the source code of the webapp. This *in-memory* zip file is sent to the **preview engine** which loads it in a simulated embedded browser and starts to intercept the Angular XHR requests. The analyst can then return back to the **design interface** and modify the configuration of the product. When the product is finished, the analyst can download the full-stack version of the application.

## 4 Conclusions and Future Work

In this paper we present a new version of GISBuilder, a SPL for web-based GIS, that provides *runtime live previewing* capabilities. This is, an analyst can see the product to build *on the fly*. We have shown how the approach in [6] can be applied in a different context to provide live previewing to a different automatic software generation tool. As future work, we want to apply Continuous Integration techniques to the GISBuilder, allowing it not only to facilitate the preview to the analyst but also the final deployment of the full-stack built products.

**Acknowledgments.** The work of the authors from UDC has been funded by MINECO (PGE & FEDER) [TIN2016-78011-C4-1-R, TIN2016-77158-C4-3-R, TIN2013-46238-C4-3-R, TIN2013-46801-C4-3-R]; CDTI and MINECO [Ref. IDI-20141259, Ref. ITC-20151305, Ref. ITC-20151247]; Xunta de Galicia (FEDER) [Ref. ED431G/01].

## References

1. Pohl, K., Böckle, G., Linden, F. Van Der. (2005): *Software Product Line Engineering: foundations, principles and techniques.* Springer Science & Business Media.
2. Luaces, M. R., Trillo, D., Lamas, J.I., Cerdeira-Pena, A. (2009): *An Urban Planning Web Viewer based on AJAX*, in WISE'09.
3. Brisaboa, N. R., Cotelo, J. A., Fariña, A., Luaces, M. R., Paramá, J. R., Viqueira, J. R. (2007): *Collecting and Publishing large multiscale geographic datasets*, in Software: Practice and Experience.
4. Places, A. S., Brisaboa, N. R., Fariña, A., Luaces, M. R., Paramá, J. R., Penabad, M. R. (2007): *The Galician Virtual Library*, in Online Information Review.
5. Brisaboa, N. R., Cortiñas, A., Luaces, M. R., Pedreira, O. (2016): *GISBuilder: a framework for the semi-automatic generation of web-based geographic information systems*, in PACIS'16.
6. Bernaschina, C., Comai, S., Fraternali, P. (2017): *Online Model Editing, Simulation and Code Generation for Web and Mobile Applications*, in MiSE'17.
7. Brambilla, M., Fraternali, P. (2014): *Interaction flow modeling language: model-driven UI engineering of web and mobile apps with IFML.* Morgan Kaufmann.