GWT to Angular: Migration of graphic applications through Models

Abderrahmane Seriai, Laurent Deruelle
Berger-Levrault, France

Benoît Verhaeghe, Anne Etien,
Nicolas Anquetil, Stéphane Ducasse
Université de Lille, CNRS, Inria,
Centrale Lille, UMR 9189 – CRIStAL, France

***Abstract*—In the context of the graphical application evolution, it happens that the application must switch of implementation langage. Berger-Levrault is a major IT company witch owns the biggest GWT application in the world. The company needs to switch the application implementation of all its software from GWT to Angular. Rewriting graphical application is complex and can lead to errors. The company has estimated the duration of the migration to . To reduce the among of time needed, we've created a tool that semi-automatically do the migration of the application. To achieve this tool, we've designed a meta model to represent a graphical user interface. Then we've created a three parts application migration strategy and the tools to apply such strategy. Firstly, we've instantiated a meta model from the source code. Then, we've generated a model that respects the meta model we've defined to represent a graphic application from the model of the source code. Finally, we've exported the *"graphic"* model to the target code.**

***Index Terms*—Graphical User Interfaces, Industrial case study, Java, Angular, GWT**

I. Introduction

Berger-Levrault is a major IT company. It uses the framework GWT to develop its graphics applications. GWT doesn't evolve these years with only one major release since 2015 whereas there was six major release between 2010 and 2013. So, Berger-Levrault has decided to migrate all its application from Java using GWT to the Angular technology. The applications Berger-Levrault are long life applications (more than ten years), they have more than X MLOC and represent X web pages. This specificities plus the fact that developers don't work in the same company for years implies a problem of perfect knowledge of the applications. No one know all the code of an application and during a migration so it leads to migration of unused code or *hack* important in the source language but witch leads to error in the target one. This is why such migration is time consuming and error prone. Berger-Levrault has estimated the duration of the migration to At the same time, the company needs to continue the development of their applications during the migration.

The migration of an application is not only is the transformation of the source code to the target language. It also implies to find the frameworks to use in the target language and to think about the architecture of the applications in the new language. The migration should have a really tiny impact for the client of the company and for the developers of the application. In the case of Berger-Levrault, multiple applications are developed following the same architecture but some other used different framework or language like ReactJS, Aurelia, Laravel or Silverlight. So, it is important to have a solution that can be reused for all the graphic applications with a minimum of modification.

To extract information from the application, it is possible to use a dynamic solution or static one.

[1] have proposed a dynamic solution to migrate Swing application to Ajax. Their tool run an instance of the source application in a browser. Then, they are able to detect the widget displayed by the interface and the different actions available. The authors have decided to use a dynamic solution to explore the interface of the application. Their contributions are a method and a middleware toolkit for re-architecting Swing application to Ajax one.

[2] have developed a static solution to reverse engineer RAD (Rapid Application Development) based graphic user interface (GUI). The authors have created different meta model to represent a GUI. The authors have used those to meta models to create a chain of transformation between them witch lead to the transformation of the source application the target one. Although we can't reapply the strategy of GUI extraction to our project, because RAD based application are too *simple* compare to GWT based application, the different meta models and the chain of transformation inspired our work.

We present a new static solution to represent a GUI developed with Java-Swing, a strategy to generate this model, a way to generate the target application whatever the target framework Architecture. Our solution is source and target independent. The tool detect well X% of the widget for a web page. After the migration, X% widgets are well placed and X% are visually identical.

The main contributions of our work are:

- a modular tool that can be reuse for other graphic application migration.
- a meta model to represent a GUI application.
- a migration strategy to migrate graphic application.

In Section II, we give background information on our problem. Then, in Section III, we describe the solution we've proposed. Section IV presents the results and threats to validity of our works. Finally, we present the related works and conclude in Section V and VII respectively.

II. Problem Description

Berger-Levrault is a major IT company witch needs to migrate their GWT-based applications to Angular. The migration of those applications will lead to X days of development. This is due to the X MLOC that composed the software they have to migrate. Our work was tested on the showroom application of Berger-Levrault. This application is used only by the company's developers as an example application of the usage of the widget available for the development.

The solution we propose must respect some criteria:

- *Source independence*. Our solution should be easy to adapt for all project written in GWT using Java and for application not written in Java but describing a GUI.
- *Target independence*. It has to be easy to change the target language without restructuring the meta model we have defined or change any stage of the migration.
- *Modularity*. The migration would be split into many little stages. This offer the possibility to easily extend on step or replace one step by another one without introducing instability.
- *Preserving Architecture*. After the migration, we should find the same architecture between the different component of the GUI (*e.g.* a button witch belongs to a panel in the source application still belongs to the same panel in the target application).
- *Layout-preserving visual*. It should not have visual difference between the source application and the target one.

The development team have to continue the maintenance of the source application. This is a constraint inherent to industrial company.

III. Proposed Solution

A. Meta model

B. Import

C. Export

IV. Discussion

V. Related Works

VI. Future Works

VII. Conclusion

Acknowledgements

This work was supported by Ministry of Higher Education and Research, Nord-Pas de Calais Regional Council, CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020.

References

[1] H. Samir, E. Stroulia, and A. Kamel. Swing2script: Migration of java-swing applications to ajax web applications. In *Reverse Engineering, 2007. WCRE 2007. 14th Working Conference on*, pages 179–188. IEEE, 2007.

[2] Ó. Sánchez Ramón, J. Sánchez Cuadrado, and J. García Molina. Model-driven reverse engineering of legacy graphical user interfaces. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*, pages 147–186. ACM, 2014.