Model Based Rapid Prototyping and Evolution of Web Application

Emanuele Falzone and Carlo Bernaschina

Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano — Piazza L. Da Vinci, 21. I-20133 Milano, Italy
emanuele.falzone@mail.polimi.it, carlo.bernaschina@polimi.it

Abstract. We demonstrate a development work-flow for the co-evolution of model and code, based on IFMLEdit.org, an online tool for the rapid prototyping of web applications, and on common Version Control Systems. IFMLEdit.org exploits the Interaction Flow Modeling Language (IFML), an OMG standard for describing the user's interaction with the application by means of flows of information in reaction to user events. In the demo, attendees will be able to edit IFML specifications with IFMLEdit.org, generate the first version of the code of a web/mobile application from the model, improve the generated code with manually added details (e.g. styling), evolve the original IFML model introducing new requirements and re-generate the code of the updated version, in a way that fully preserves the manually coded details. The demonstrated approach solves the well-know problem of model driven forward enginnering of breaking the automated development cycle when features that cannot be modelled are added manually to the generated code.

Keywords: Model Driven Development, Code Generation, Agile Development.

1 Introduction

The software development process is a refinement loop that iteratively transforms requirements into a final product. Iteration is fundamental to address incomplete, loosely defined, or rapidly changing requirements. In web and mobile applications development, the wide range of device screens and coding platforms makes the ability to rapidly evolve and evaluate new releases even more critical.

In Web applications, UI development demands a sharp division between structure (HTML) and style (CSS) for easy adaptation to various devices and clients capabilities [2]. Complexity of UI layout and styling is shifted from HTML, which describes the structure and semantics of the content, to CSS. While this scenario enables advanced use-cases, in practice it may not achieve a real separation of concerns in all cases. To obtain advanced layout and styling effects, the CSS rules often become dependent on the HTML structure, which increases complexity, maintenance cost, and code duplication.

In the past years, coding practices shifted towards an effective compromise between separation of concerns and development costs. Modern CSS frameworks, such as Bootstrap[1], Zurb Foundation[2], Materialize CSS[3], and many others, have shown how sharing layout concerns between HTML and CSS layers can enhance re-usability and eventually reduce development time. The same trend can be seen in the field of Mobile Applications with Framework7[4], Flutter[5] and many others. This compromise blurs the line between structure and styling making more and more difficult for M2T transformations to avoid conflicts at code level.

In this demo, we showcase IFMLEdit.org [1], an open-source on-line tool for rapid prototyping of web and mobile applications that starts from a standard OMG MDA language (IFML) and implements a lightweight environment for developers to specify web and mobile applications and instantly generate their code. In the demo, attendees will use the on-line tool to specify their requirements in IFML, validate their model with in-browser emulation, generate and deploy the code for a web application or cross-platform mobile application, manually evolve the generated code by introducing details like styling or back-end service endpoints, introduce new requirements at the model level and integrate the newly generated code into the code-base preserving the manually applied changes.

2 Background: the Interaction Flow Modeling Language

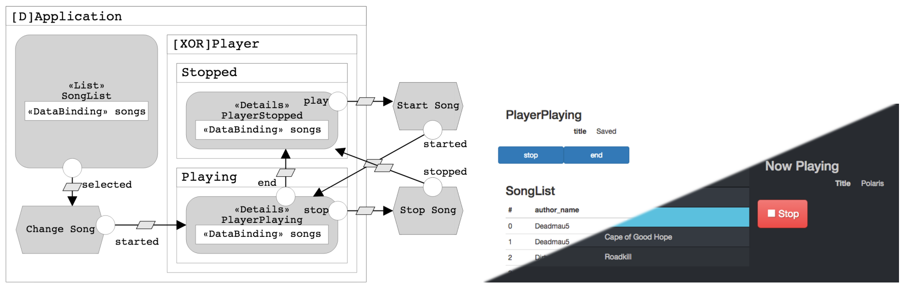
IFML (Interaction Flow Modeling Language [3]) is an OMG standard that supports the platform-independent description of graphical user interfaces (UIs) for devices such as desktop computers, laptops, mobile phones, and tablets. IFML focuses on the structure and behavior of the application as perceived by the end user, and references the data and business logic aspects insofar they influence the users experience, i.e., the domain objects that provide content displayed in the interface and the actions triggered from the interface. IFML allows developers to specify the following aspects of an interactive application:

- **The view structure and content:** the general organization of the interface is expressed in terms of *ViewElements*, along with their containment relationships, visibility, and activation. Two classes of ViewElements exist: *ViewContainers*, i.e., elements for representing the nested structure of the interface, and *ViewComponents*, i.e., elements for content display and data entry. *ViewComponents* that display content have a *ContentBinding*, which expresses the link to the data source.
- **The events:** the occurrences that affect the state of the user interface, produced by the users interaction, the application, or an external system.
- **The event transitions:** the consequences of an event on the user interface, which can be the change of the *ViewContainer*, the update of the content on display, the triggering of an action, or a mix of these effects. *Actions* are represented as black boxes.
- **The parameter binding:** the input-output dependencies between *ViewElements* and *Actions*.

[1] http://getbootstrap.com/
[2] http://foundation.zurb.com/
[3] http://materializecss.com/
[4] http://framework7.io/
[5] http://flutter.io/

3 Work-flow of the Demo

The demo will allow attendees to experience all the operations required to generate a prototype application. As a starting example, we have prepared a multimedia player application; its model is shown in Figure 1a and comprises a single page, with three components: a list, which shows the available songs, and two details that represent the media player status and allow the user to start and stop the selected song.



(a) Model (b) Generated & Customized Code

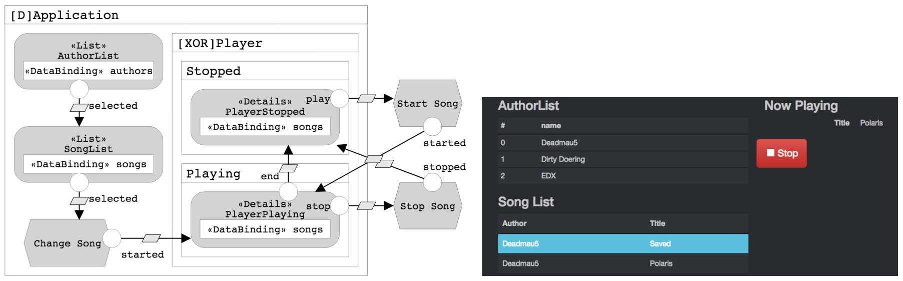
Fig. 1. Model Editing and Code Generation

IFML model editing. During the demo, users will compose the model of Figure 1a, or a model of their choice, using the integrated editor. IFML elements are inserted in the model by means of drag&drop from a palette on the left side.

Code generation. The developer can generate a fully functional prototype launching a model-to-code transformation. Figure 1b shows the generated web prototype, run on top of a client emulator inside the browser. In-browser emulation allows the developer to test the current web or mobile release of the prototype without installing any web server and also in absence of the Internet connection. The developer can evaluate different application structures (e.g., single vs multiple pages) and interaction approaches (e.g., update on object selection vs explicit update events) before the final code download.

Manual code editing. Once the generated code is downloaded, the developer can modify it in order to: **1)** connect data sources to existing data provisioning service endpoints; **2)** introduce business logic, editing the code that implements the Actions skeletons generated from the IFML model; **3)** achieve the desired look and feel by customizing the HTML template files and introducing custom CSS rules. Figure 1b shows the difference between the generated code (upper left, white background) and the customized code (bottom right, black background).

Model evolution. After a new requirement, the developer meeds to change the original IFML model to introduce new features or change/remove existing ones. Figure 2a shows a possible evolution of the base model with the introduction of a new list that allows the users to filter the songs by author.



AuthorList

#	name
0	Deadmau5
1	Dirty Doering
2	EDX

Song List

Author	Title
Deadmau5	Saved
Deadmau5	Polaris

(a) Model Editor (b) Generated Code

Fig. 2. Model Editing and Code Generation

Model and text co-evolution. The developer can generate a new prototype launching the model-to-code transformation. ALMOsT-Git, an automation tool based on Git[6], will reapply the previously introduced changes using the new prototype as a starting point. If changes alter the application structure drastically, manual conflict resolution by the developer may be required. Figure 2b shows the result of the procedure.

4 Conclusion

The demo allows attendees to explore the problems of MDD forward engineering, when manual and automatic code updates occur in parallel. IFMLEdit.org, an online tool for the modeling and rapid prototyping of web and Mobile applications based on IFML allows Attendees to try out an application development cycle that allows them to evaluate different variations of an application in a short amount of time, by rapidly modifying the application model and generating realistic prototypes, easily turned into deployable applications.

Acknowledgments This work has been partially supported by the European Community, through the H2020 project enCOMPASS (Grant #723059).

References

1. Bernaschina, C., Comai, S., Fraternali, P.: IFMLEdit.org: Model driven rapid prototyping of mobile apps. In: 4th IEEE/ACM International Conference on Mobile Software Engineering and Systems, MOBILESoft@ICSE (2017)
2. Hall, C.A.: Web presentation layer bootstrapping for accessibility and performance. In: Proceedings of the International Cross-Disciplinary Conference on Web Accessibility, W4A (2009)
3. OMG: Interaction flow modeling language (IFML), version 1.0. http://www.omg.org/spec/IFML/1.0/ (2015)

[6] http://git-scm.com