IFMLEdit.org: a Web Tool for Model Based Rapid Prototyping of Web and Mobile Applications

Carlo Bernaschina
Politecnico di Milano
Milano, Italy
carlo.bernaschina@polimi.it

Sara Comai
Politecnico di Milano
Milano, Italy
sara.comai@polimi.it

Piero Fraternali
Politecnico di Milano
Milano, Italy
piero.fraternali@polimi.it

***Abstract*—We demonstrate IFMLEdit.org, an online tool for the rapid prototyping of web and mobile applications. IFMLEdit.org is based on the Interaction Flow Modeling Language (IFML), an OMG standard for the description of the interaction between users and applications by means of flows of information in reaction to user events. It has been developed using a novel framework called ALMOsT (AgiLe MOdel Transformation), an agile modular framework which allows developers to rapidly bootstrap and evolve their MDD tools (Editor, Code Generators, ...). In the demo, attendees will be able to edit IFML specifications with IFMLEdit.org, investigate their properties by transforming them into Place Chart Nets, a variant of Petri Nets, and generate the code of web and mobile applications from the validated IFML model.**

Video: http://youtu.be/y_hDVeUbi7g
Website: http://www.ifmledit.org

***Keywords*-Model-driven development, Computer aided software engineering, Mobile applications**

I. 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 functional and non-functional 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.

A common approach to tackle the problem is supported by cross-platform mobile development tools, which allow the creation and distribution of mobile applications to multiple platforms: many solutions leverage existing web development skills and require coding in languages such as Javascript/Java/C#, CSS and HTML5. Examples of such tools include Appcelerator Titanium [1], IBM MobileFirst Platform Foundation [2], PhoneGap [3], RhoMobile [4], Salesforce [5], Telerik AppBuilder [6], Xamarin [7] and many others: the developer writes code only once and the tool derives the implementation for different target platforms, including native applications, standard web applications (typically based on HTML5, Javascript and CSS), and hybrid applications (e.g., embedding HTML5 apps inside native containers that provides access to native platform features).

An alternative approach is to exploit the Model Driven Development (MDD) paradigm [8]. The application is represented at high level by means of a textual or visual language and model-to-model and model-to-code transformations are used to generate the final product. MDD differs from cross-platform development in that it shifts the development effort from code to models and transformations, promoting the deployment of early prototypes and mitigating the divergence between specifications and implementation typical of fast evolving application requirements.

Several MDD tools for mobile application development exist. Earlier approaches developed their own Domain Specific Languages (DSLs) and code-generators, e.g., based on the Xtext framework and Eclipse [9]; examples include Applause [10], an early open source project, and Canappi, currently evolved into BuildUp App Builder [11]. More recent MDD approaches specific for cross-platform mobile applications are surveyed in [12]: they are mainly based on textual DSLs, rely on ad-hoc, informally specified modeling languages and transformations, and are currently in a prototypical status.

In the demo, we showcase IFMLEdit.org, an open-source online tool for rapid prototyping of web and mobile applications that starts from a standard OMG MDA language (IFML), provides a Petri Net formal semantics as the base of model interpretation and model-to-code transformation, and implements a lightweight environment for developers to specify web portals and apps and instantly validate them and generate multi-platform code. In the demo, attendees will peruse the online tool to specify their requirements in IFML, map the model to a Place Chart Net, simulate the execution at the semantic level, and generate and deploy the code for a web application or cross-platform mobile app.

II. Background: the Interaction Flow Modeling Language

IFML (Interaction Flow Modeling Language [13]) 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, which can be 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*.

III. System Architecture

IFMLEdit.org is an online environment for the specification of IFML models, the investigation of their properties by means of a mapping to Place Chart Nets [14], and the generation of code for web and mobile architecture. The tool supports the following workflow: 1) the developer edits the IFML model of the application in the online editor, possibly providing hints for the generation of the fast prototype (e.g., sample data); 2) he (optionally) maps the model into a PCN and simulates the network to understand the dynamics of the application in response to events; 3) he generates the code of a fast prototype, for the web or for a cross-platform mobile language, executes and validates the prototype; 4) he turns the validated prototype into a real app, by customizing look&feel and replacing mock-up data access and operational APIs calls with real ones.

IFMLEdit.org is realized using web technologies; it exploits client-side editors and code generators developed in JavaScript, which, once loaded, can be used also offline. This guarantees that developers new to the platform can start production without complex installations and configurations, but at the same time allows them to work offline and maintain full control and privacy of their projects.

The system comprises four components:

- **Model editor:** a general-purpose visual editor [15] that allows the developer to insert (by means of drag&drop) elements in the model, edit their property and connect them.
- **Model-to-JSON transformation framework:** a generic rule-based transformation framework [15] developed in JavaScript that, given as input a model object, can generate an output JSON structure that can resemble another model (model-to-model transformation), a file system structure (model-to-text transformation) or anything that can be described as a JSON object.
- **Browser-Server emulator:** a pure JavaScript component able to emulate a web browser, a Node.js server and the whole request response cycle that connects the two. It is used to support online and offline work seamlessly.
- **Mobile emulator:** a JavaScript component able to emulate a mobile cross-platform environment (now Cordova); it supports the instantaneous execution of the generated cross-platform mobile code.

The system offers to the developer four tools:

- **IFML editor:** a customization of the generic model editor to draw IFML models.
- **PCN generator and Simulator:** based on the IFML model editor and the model-to-JSON transformation framework, it allows one to generate a PCN from the IFML model and simulate its behavior.
- **Web code generator and emulator:** based on the Browser-Server emulator and the model-to-JSON transformation framework, it allows one to generate a Node.js application from the IFML model and run it in the browser.
- **Modile code generator and emulator:** based on the mobile emulator and model-to-JSON transformation framework, it supports the generation of a Cordova app from the IFML model and its emulation in the browser.

IV. 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 simple mail client application; its model is shown in Figure 1 and comprises a single page, with three components: 1) **MailList:** a list of mails that the user can select. 2) **MailContent:** a component showing the content of the email selected from the list, in particular the *subject* and the *body*. 3) **AttachList:** the list of attachments related to the currently selected mail.

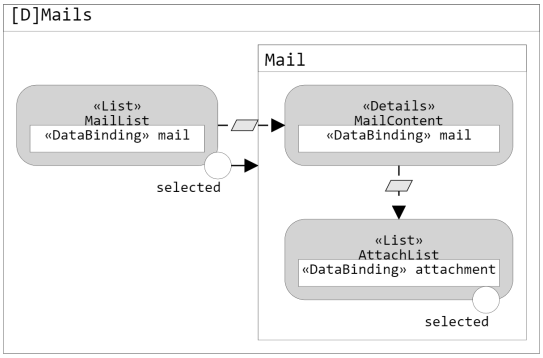


Fig. 1: Application model

A. IFML model editing

During the demo, users will compose the model of Figure 1, or a model of their choice, using the integrated editor (see Figure 2). IFML elements are inserted in the model by means of drag&drop from the palette on the left side. *NavigationFlows* and *DataFlows* are drawn by clicking on the source *ViewComponent* or *Event* and using drag&drop to

connect them to the target *ViewContainer*, *ViewComponent* or *Action*.

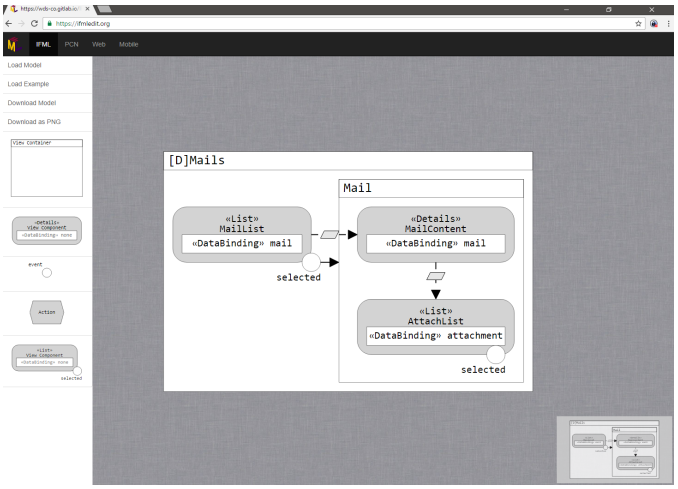
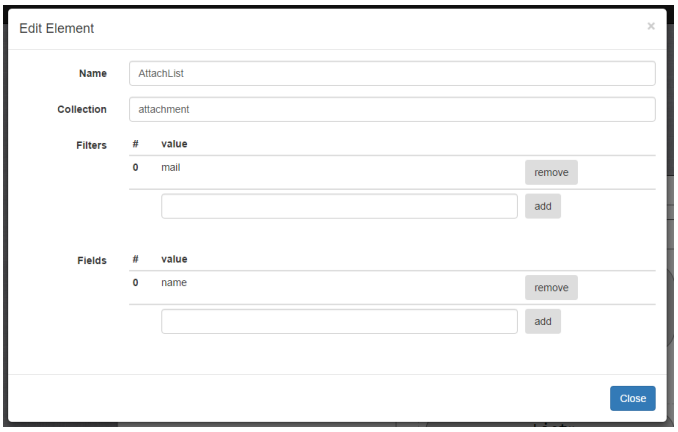


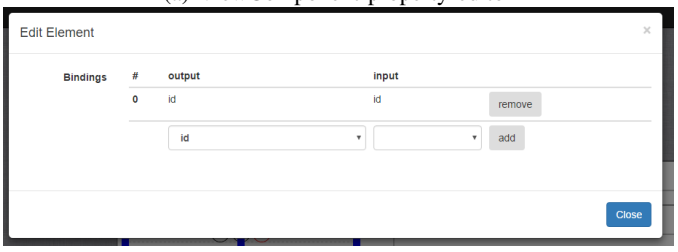
Fig. 2: IFML model editor

B. Data-bindings

Once the structure of the application is modeled, the developer can use the property editor (Figure 3) to specify how *ViewComponents* connect between them and to the data sources.



(a) ViewComponent property editor



(b) InteractionFlow property editor

Fig. 3: IFML element property editors

Figure 3a shows how to edit the properties of a ViewComponent so to specify, among other things, its database binding.

- **Collection:** defines the set of Domain Model entities used as data-source for the *ViewComponent*.
- **Filters:** define the data-source attributes and predicates used to select relevant instances of the entities. The provenance of values used to evaluate filters at runtime is specified by editing the DataBinding IFML element associated with inbound *InteractionFlows*, explained next.
- **Fields:** defines the data-source attributes displayed in the UI and exposed as output by outbound *InteractionFlows*.

Figure 3b shows how to edit the properties of the DataBinding associated with an*InteractionFlow* connecting *ViewComponents*. A **DataBinding** defines $(field, filter)$ pairs that dictate how parameters are transferred from a source to a target *ViewComponent*.

C. Model semantics and simulation (Optional)

The developer can generate a formal description of the application by running the model-to-model transformation from IFML to PCNs. The application behavior is rendered visually by means of tokens moving in the net, displaying the control flow in the interface and the change of status of *ViewElements*. Figure 4 illustrates the PCN model generated from the IFML diagram of Figure 1; the PCN simulation helps the developer identify inconsistencies in the specified application, such as unreachable states and race conditions.

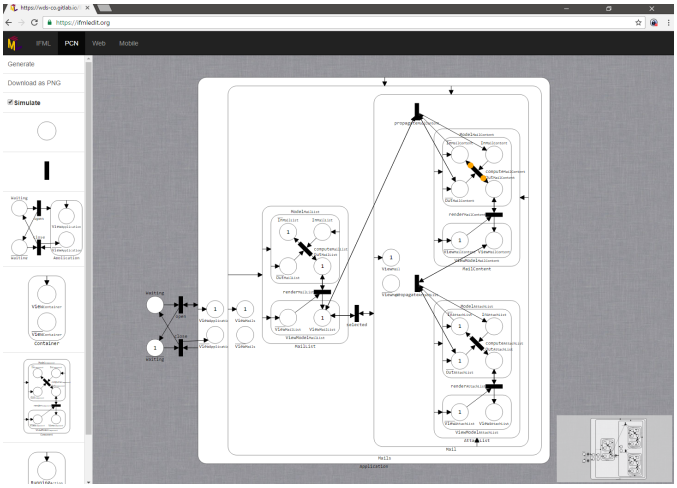


Fig. 4: Model semantics simulation

D. Code generation

The developer can generate a fully functional prototype for both the web and mobile architecture, launching a model-to-code transformation. Figure 5 shows the generated web prototype, run on top of the web server emulated inside the browser. Figure 6 shows the generated mobile prototype, run within the mobile 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.

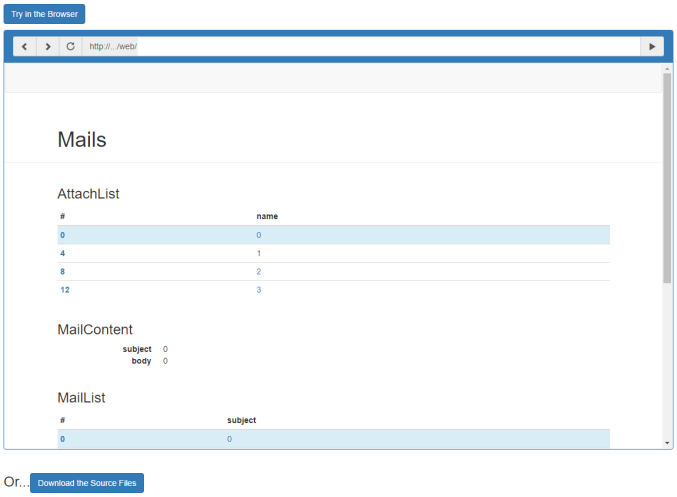


Fig. 5: Web code generation

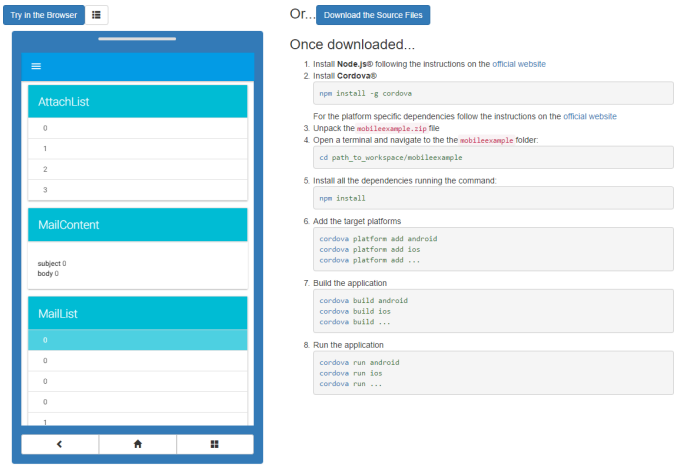


Fig. 6: Mobile code generation

E. Mock-up database customization

The prototype uses automatically generated sample data, so to populate the UI, allowing the developer to test the interface without the need of loading data. To test the prototype in more realistic conditions, the developer can edit, add or remove entities from the automatic database. Figure 7 shows the integrated database editor. Changes to the database can be applied, with different policies:

- **Save:** changes are saved, but have no immediate effect to the current UI state, which is refreshed lazily only at the next user interaction.
- **Save and Reload:** changes are saved and the prototype is fully reloaded, restoring the initial state of the UI and erasing any data update operation made by the user.
- **Save and Hot-Reload:** changes are saved and the prototype is refreshed; the UI is updated but the current state of the interaction is preserved.

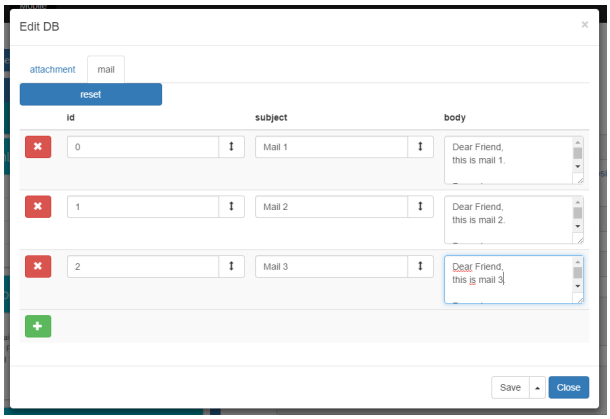


Fig. 7: The integrated database editor

F. Prototype download

The previous steps can be reiterated to evaluate different application structures (e.g., single vs multiple pages) and interaction approaches (e.g., update on object selection vs explicit update events). The generated prototype can be downloaded and refined to produce the final application. Each IFML Action and ViewComponent data query is encoded as a web service, which can be replaced by an external implementation.

V. Conclusions

The demo presents IFMLEdit.org, an online tool for the modeling and rapid prototyping of web and Mobile applications based on IFML. Attendees will 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 deployed applications.

References

[1] "Appcelerator platform," http://www.appcelerator.com/.
[2] "IBM MobileFirst platform foundation," https://www.ibm.com/support/knowledgecenter/SSNJXP/welcome.html.
[3] "Adobe Phonegap," http://phonegap.com/.
[4] "Rhomobile suite," http://rhomobile.com.
[5] "Salesforce platform," https://www.salesforce.com.
[6] "Telerik appbuilder," http://www.telerik.com/platform/appbuilder.
[7] "Xamarin platform," https://www.xamarin.com/.
[8] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice*, 1st ed. Morgan & Claypool Publishers, 2012.
[9] "Xtext platform," https://eclipse.org/Xtext/.
[10] "Applause project," http://applause.github.io/.
[11] "BuildUp app builder," http://www.buildup.io/.
[12] L. Gaouar, A. Benamar, and F. T. Bendimerad, "Model driven approaches to cross platform mobile development," in *Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication*, ser. IPAC '15. New York, NY, USA: ACM, 2015, pp. 19:1–19:5.
[13] OMG, "Interaction flow modeling language (ifml), version 1.0," http://www.omg.org/spec/IFML/1.0/, 2015.
[14] M. Kishinevsky, J. Cortadella, A. Kondratyev, L. Lavagno, A. Taubin, and A. Yakovlev, "Coupling asynchrony and interrupts: Place chart nets," in *Application and Theory of Petri Nets 1997, 18th International Conference, ICATPN '97, Toulouse, France, June 23-27, 1997, Proceedings*, ser. LNCS, P. Azéma and G. Balbo, Eds., vol. 1248. Springer, 1997, pp. 328–347.
[15] "ALMOsT: AgiLe MOdel Transformation Framework," https://www.npmjs.com/package/almost.