Extending the Interaction Flow Modeling Language (IFML) for Model Driven Development of Mobile Applications Front End

Marco Brambilla, Andrea Mauri, Eric Umuhoza

Extending the Interaction Flow Modeling Language (IFML) for Model Driven Development of Mobile Applications Front End

Marco Brambilla[1], Andrea Mauri[1], and Eric Umuhoza[1,2]

[1] Politecnico di Milano. Dipartimento di Elettronica, Informazione e Bioingegneria
Piazza L. Da Vinci 32. I-20133 Milan, Italy
{marco.brambilla, andrea.mauri, eric.umuhoza}@polimi.it
[2] AtlanMod team, Ecole des Mines de Nantes
4, rue Alfred Kastler 44307 Nantes Cedex 3, France
eric.umuhoza@mines-nantes.fr

Abstract. Front-end design of mobile applications is a complex and multidisciplinary task, where many perspectives intersect and the user experience must be perfectly tailored to the application objectives. However, development of mobile user interactions is still largely a manual task, which yields to high risks of errors, inconsistencies and inefficiencies. In this paper we propose a model-driven approach to mobile application development based on the IFML standard. We propose an extension of the Interaction Flow Modeling Language tailored to mobile applications and we describe our implementation experience that comprises the development of automatic code generators for cross-platform mobile applications based on HTML5, CSS and JavaScript optimized for the Apache Cordova framework. We show the approach at work on a popular mobile application, we report on the application of the approach on an industrial application development project and we provide a productivity comparison with traditional approaches.

1 Introduction

Front-end design is a complex and multidisciplinary task, where many perspectives intersect. Front-end becomes even more crucial in mobile applications, where the user experience must be perfectly tailored to the application objectives. However, development of mobile user interactions is still largely a manual task, which yelds to high risks of errors, inconsistencies and inefficiencies. Several researches have applied model-driven techniques to the specification of software application interfaces and user interaction in broad sense. Among them, we can cite the ones focusin on Web interfaces (W2000 (HDM) [2], OO-HDM [16], WebDSL [7], OOH-Method [8], WebML [5], RUX-Model [11] and HERA [20]). Furthermore some approaches apply model driven techniques for multi-device UI modeling such as IFML [4], TERESA [3], MARIA [14], MBUE [13], UsiXML [19] and UCP [15].

However, none of them specifically address the needs of mobile applications development. Therefore, in mobile applications, front-end development continues to be a costly and inefficient process, where manual coding is the predominant development approach, reuse of design artifacts is low, and cross-platform portability remains difficult. The availability of a platform-independent user interaction modeling language can bring several benefits to the development process of mobile application front-ends, as it improves the development process, by fostering the separation of concerns in the user interaction design, thus granting the maximum efficiency to all the different developer roles; it enables the communication of interface and interaction design to non-technical stakeholders, permitting early validation of requirements.

In this paper we propose a model-driven approach to mobile application development based on the OMG standard language called Interaction Flow Modeling Language (IFML). We propose an extension of the IFML modeling language tailored to mobile applications and we describe our implementation experience that comprises the development of automatic code generators for cross-platform mobile applications based on HTML5, CSS and JavaScript optimized for the Apache Cordova framework. We show the approach at work on a running example based on a popular existing application, and we report on one of our experiences in developing industrial mobile applications,with a short summary on the productivity comparison with traditional approaches.

The paper is organized as follows: Section 2 reviews the related work; Section 3 summarizes the core features of the IFML language; Section 4 presents our extensions to IFML talored to mobile application development; Section 5 shows a running example; Section 6 describes our implementation and our industrial experience (including a comparison to traditional approaches); and Section 7 concludes.

2 Related work

This work is related to a large corpus of researches that address conceptual modeling of software applications. Among the ones mainly focusing on the Web we can cite: (i) The Web Modelling Language (WebML) [5], defined as a conceptual model for data-intensive Web applications and conceived as a high level, implementation-independent conceptual model accompanied by the associated design environment, called WebRatio [1]; (ii) W2000 (formerly HDM) [2] which introduced a notion of model-based design, clearly separating the activities of authoring in-the-large (hypertext schema design) and authoring in-the-small (page and content production); (iii) OO-HDM [17], a UML-based approach for modeling and implementing Web application interfaces; (iv) Araneus [12], a modeling proposal for Web applications that allows one to represent the hypertext organization, with nested relations and inter-page links; (v) Web Application Extension for UML (WAE) [6], a UML extension for describing Web application interfaces and the client-server interactions; (vi) WebDSL [7], a domain-specific language consisting of a core language with constructs to define entities, pages

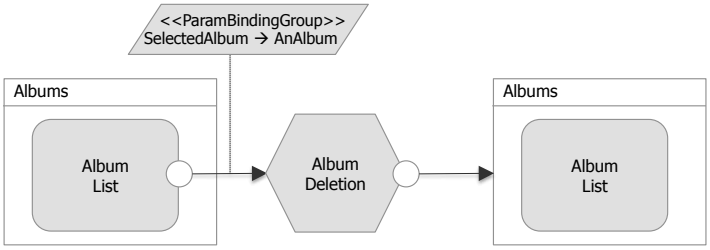

Fig. 1. IFML example: product search, listing and deletion.

and business logic, plus extensions; (vii) OO-HMETHOD [8], based on UML interaction diagrams; (viii) Hera [20], a model-driven design approach and specification framework focusing on the development of context-dependent or personalized Web information system. More traditional methods for user interface design use state machines (in different flavours) as the underlying formalism, such as Jacob [9], Leung et al. [10] and StateWebCharts [21]. Other recent proposals in the Web Engineering field represent the RIA foundations (e.g., Urbieta [18]) by extending existing Web engineering approaches. Commercial vendors are proposing tools for Web and mobile development, such as IBM WorkLight, Mendix and Outsystems.

Some researches apply model based approaches for multi-device user interface development. Among them we can cite: (i) TERESA(Transformation Environment for inteRactivE Systems representations)[3], based on a so-called *One Model, Many Interfaces* approach to support model based GUI development for multiple devices from the same ConcurTaskTree (CTT) model; (ii) MARIA [14], another approach based on CTT; (iii) MBUE(Model Based Useware Engineering); (iv) UsiXML (USer Interface eXtended Markup Language) [19] and (v) Unified Communication Platform (UCP).

3 The Interaction Flow Modeling Language (IFML)

The Interaction Flow Modeling Language (IFML) supports the platform independent description of graphical user interfaces for applications accessed or deployed on such systems as desktop computers, laptop computers, PDAs, mobile phones, and tablets. An IFML model supports the following design perspectives: (1) The *view structure specification*, which consists of the definition of view containers, their nesting relationships, their visibility, and their reachability; (2) The *view content specification*, which consists of the definition of view components, i.e., content and data entry elements contained within view containers; (3) The *events specification*, which consists of the definition of events that may affect the state of the user interface. Events can be produced by the user's interaction, by the application, or by an external system; (4) The *event transition specification*, which consists of the definition of the effect of an event on the user interface;

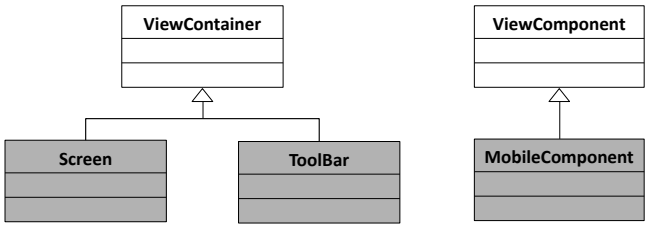

Fig. 2. IFML mobile extension: Screen,ToolBar and mobileComponent.

(5) The *parameter binding specification*, which consists of the definition of the input-output dependencies between view components and between view components and actions; and (6) The reference to *actions* triggered by the user's events. The effect of an event is represented by an *interaction flow* connection, which connects the event to the view container or component affected by the event. The interaction flow expresses a change of state of the user interface: the occurrence of the event causes a transition of state that produces a change in the user interface.

Figure 1 shows a simple example of IFML model where the user can search for a product by entering some criteria in the Product Search Form. The matching items are shown in a list. The selection of one item causes a delete action to be triggered on it. When the deletion is completed, the updated list of products is displayed again.

IFML concepts can be stereotyped to describe more precise behaviours. For instance, one could define specific stereotypes for describing web pages (a specific kind of view container); forms, lists and details (specific kinds of view component); submission or selection events; and so on.

4 Extending IFML for mobile apps

Mobile applications have rich interfaces that resemble on a smaller scale those of full-fledged desktop applications, possibly with additional complexity of the interaction patterns, at the purpose of exploiting at best the limited space available. This aspect, together with mobility and availability of sensors, such as camera and GPS, introduce features that are best captured by providing mobile-specific extensions of a platform-independent modeling language like IFML.

The proposed extensions come from extensive modeling experience on mobile applications using the IFML standard, which covered both modeling of several existing mobile applications (including *CamScanner*, *Instagram*, *iPhone Gallery*, *Twitter*, *RedLaser* and many others) and design and implementation of new mobile applications for industrial customers (as reported in Section 6).

The IFML standard as described in Section 3 does not cover the mobile-specific aspects. In the next paragraphs, we describe the IFML extensions allowing the modeling of mobile application. Those extensions will address both the

Fig. 3. IFML mobile extension: Example of the *LongPress* event.

components of the user interface and the events. In our experiments, we identified three categories of mobile events: (i) events generated by the interaction of the user such as *tap and hold*, *swipe*, etc.; (ii) events triggered by the mobile device features such as sensors, battery, etc.; and (iii) finally events triggered by user actions related to the device components such as taking a photo, recording a video or using the microphone.

4.1 IFML Mobile extension: Containers and Components

In this section we describe the concepts added in order to model the components that characterized the mobile context (as shown in Figure 2). A new class called *Screen* has been defined to represent the screen of a mobile application. Since the screen is the main container of a mobile application, it extends the core class *ViewContainer* of the IFML standard. The class *ToolBar* represents a particular subcontainer of the screen. It may contain other containers and may have on its boundary a list of events. It extends the core class *ViewContainer* of IFML standard.

A characteristic trait of mobile interfaces is the utilization of predefined ViewContainers devoted to specific functionalities(including *Notifications* area and *Settings* panel). These system level containers provide economy of space and enforce a consistent usage of common features. The *MobileSystem* stereotype has been defined to distinguish these special ViewContainers. A ViewContainer stereotyped as *MobileSystem* denotes a fixed region of the interface, managed by mobile operating system or by another interface framework in a cross-application way. The class *MobileComponent* denotes the particular mobile view component such as buttons, images, icons etc. A MobileComponent is subject to user events,described next. The *MobileSystem* stereotype can be applied also to ViewComponents to highlight that the interface uses the components built-in in the system(as shown in Figure 7).

4.2 IFML Mobile extension: MobileContext

The Context assumes a particular relevance in mobile applications, which must exploit all the available information to deliver the most efficient interface. There-

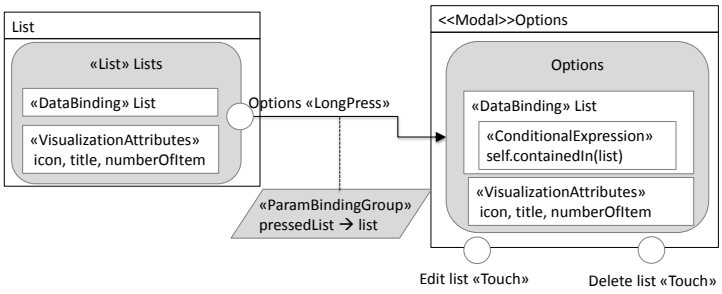

Fig. 4. IFML Mobile extension: Example of *LongPress* event used to display options of a pressed list.

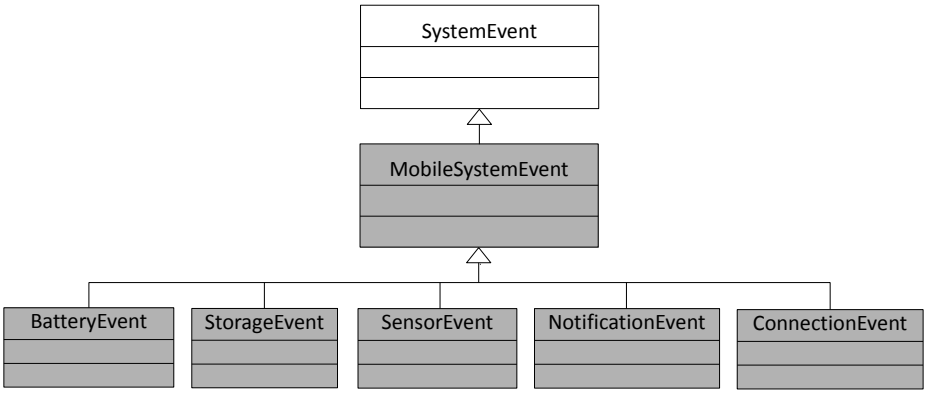

Fig. 5. IFML mobile extension: Mobile system events

fore, the context must gather all the dimensions that characterize the user's intent, the capacity of the access device and of the communication network, and the environment surrounding the user. A new class *MobileContext* extending the *Context* has been defined to express the mobile contextual features.

4.3 IFML Mobile extension: Events

In this section we describe the new event types that are defined within IFML for the mobile context. First, a new class *MobileUserEvent* allowing the modeling of the mobile user events have been defined. *MobileUserEvent* extends the core class *ViewElementEvent* of the IFML standard. The following classes extend MobileUserEvent for modeling the specific mobile user events: *DragDrop*; *DoubleTap*; *Swipe*; *Pinch*; *Spread*; *Touch*; *LongPress*; *Scroll*; and *Shake*. Each class represents an event related to the gesture which triggers it.

The screens in Figure 3 show an example of the usage of the *LongPress* gesture allowing the user to manage the selected list. Figure 4 shows a fragment of IFML model for lists management. When a user performs the *LongPress*

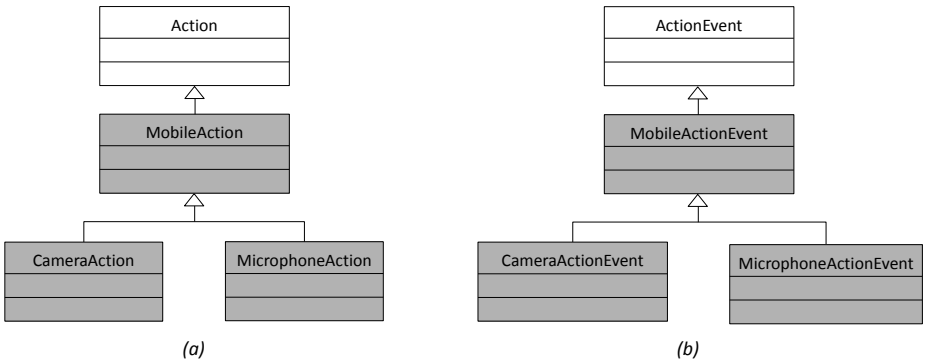

Fig. 6. IFML mobile extension:*(a) MobileAction*, the extension of the IFML *Action* to address specific mobile actions. *(b) MobileActionEvent*, the extension of the IFML *ActionEvent* class.

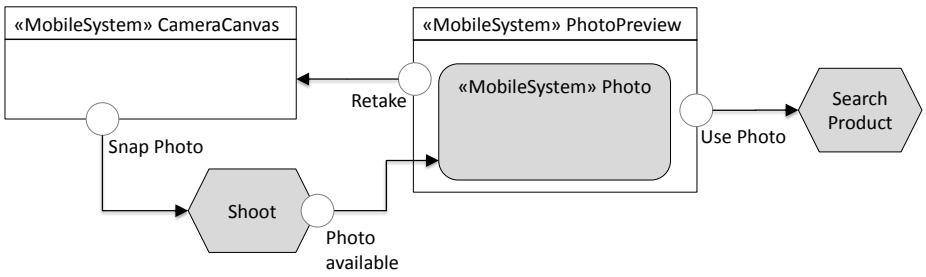

Fig. 7. IFML Mobile extension: Example of usage of MobileAction(*Shoot*), MobileActionEvent(*Photo available*) and *MobileSystem* streotype.

gesture on one element of the list a pop up containing information of the selected element is shown allowing her to edit or delete the list.

A new class *MobileSystemEvent* extending *SystemEvent* has been defined to express the mobile system events. The following classes extend *MobileSystemEvent* for specific system events.

- *BatteryEvent*,describing the events related to the state of the battery.
- *StorageEvent*,describing the events related to the archiving capacity.
- *NotificationEvent*,grouping the events related to the generic notifications handled by the operating system.
- *ConnectionEvent*,describing the events related to the connection state of the device.
- *SensorEvent*, defining events related to the sensors of the device. The *SensorEvent* extends ExternalEvent IFML core class. The most commonly used sensors are proximity sensor, motion sensor, magnetometer sensor,Gyroscope and position sensor. The classes *PositionEvent*, *MotionEvent*, *AccelerationEvent*, *ProximityEvent* and *RotationEvent* extend the *SensorEvent* to represent the events related to the specific sensors.

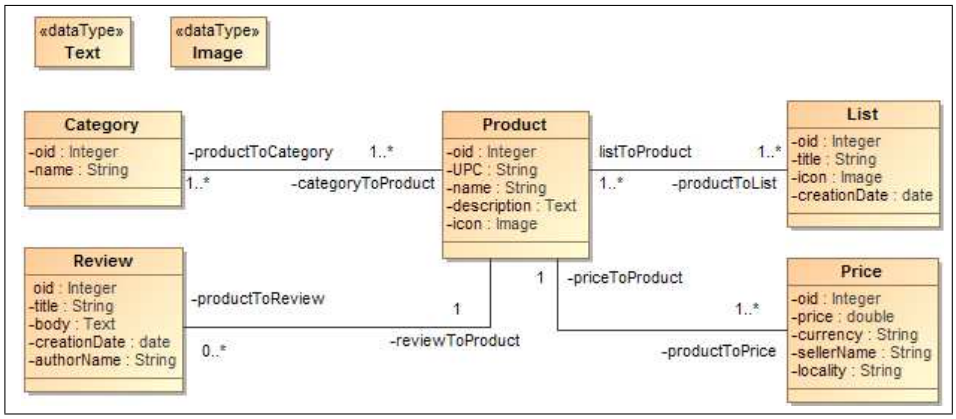

Fig. 8. Domain model of the RedLaser mobile app

MobileActionEvent class has been defined to model the events triggered by a mobile action. Among mobile actions, we have actions related to the photo camera such as the *Shoot* action and actions related to microphone as reported in Figure 6. Figure 7 shows example of such events. A user takes a photo with the device's photo camera and the application displays the product corresponding to the taken photo if any. Once the photo is available, a screen asking the user if he wants to use or retake the photo is displayed. The *photo available* event is associated to the action *shoot*.

5 Case study: the RedLaser example

To demonstrate the applicability and the expressive power of the proposed extensions, this section exemplifies their use by modeling some of the functions of RedLaser[3], a shopping app available for iPhone, Windows Phone and Android. Figure 8 shows the data model of the RedLaser application. In RedLaser the *products* are organized in *categories*. Each product has one or more *prices*. A user can create a *list* of the products he likes. The application allows the user to *review* a product.

The RedLaser user interface consists of a top level container, which is logically divided into three alternative sub containers one containing the recent and popular products, the second one containing the history of the application and the last one allowing the user to save its favorites(as shown in Figure 9).

The IFML model of the top screen comprises the *ToolBar* of the app and three screens in alternative. (1) *Home*. Is the default screen accessed when the user starts the app; (2) *History*. It contains the log of the application; and (3) *Lists*. It allows the user to save its favorite products.

[3] http://redlaser.com/

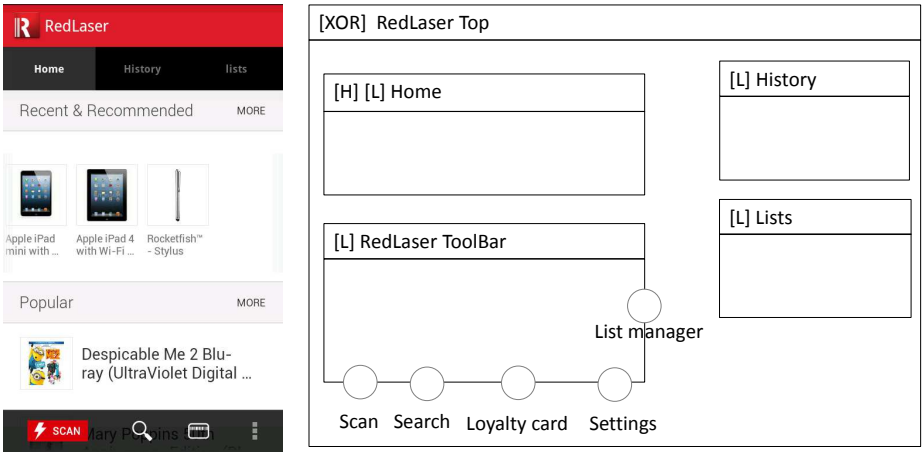

Fig. 9. Case study: *(a)* the home screen of RedLaser application; *(b)*IFML model showing logical organization of RedLaser application.

Product searching. The app allows the user to search a product in four different ways: (1) keyword search; (2) Photo based searching. The input of the searching system is a picture taken through the camera of the device; (3) Voice based searching; and (4) Barcode search.Figure 10 shows the screens allowing the searching of a product. Figure 11 shows a piece of IFML model for Keyword-Search and a voice based searching. The user can use the *product selection* event associated to each product on the lists of retrieved products to see its details as shown on the Figure 12.

Figure13 shows a piece of IFML model of the product overview. This model shows also the usage of the *swipe* user event to navigate among the sub screens of overview.

6 Implementation experience

Besides the formal definition of the mobile extensions to the IFML language, our research included the implementation of a complete prototype model editor in Eclipse, a code generator tailored to cross-platform mobile development, a few exemplary models built by reverse-engineering existing mobile applications, and some industrial developments for actual customers.

6.1 Modeling Tool

A snapshot of the modeling interface is shown in Figure 15. The tool has been implemented using the Obeo Sirius framework (`http://www.eclipse.org/`

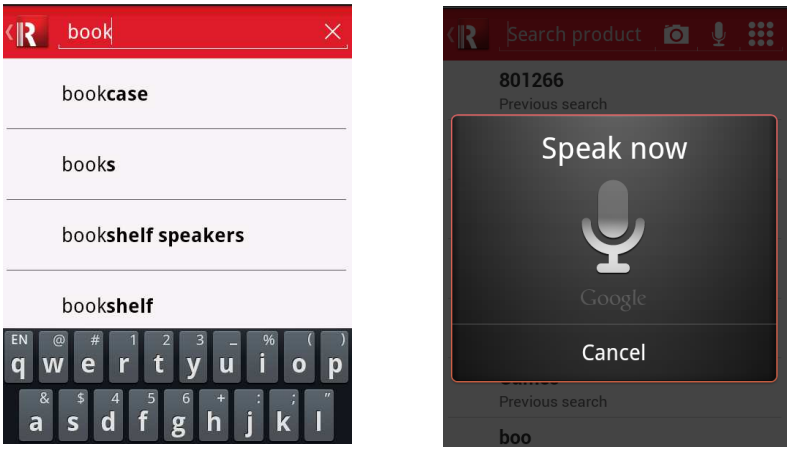

Fig. 10. Case study:UI for product searching. *(a)* Shows the user interface for *keyword based* search; *(b)* Shows the interface allowing the user to interact with the device's microphone in *voice based* search.

`sirius/`) and will be released as opensource Eclipse Project, and will benefit from the contribution of the community of MDE developers.

6.2 Code Generator

Regarding the mobile applications generation, our architecture is based on Apache Cordova (an open-source cross-platform framework for mobile development), specifically on the Adobe PhoneGap distribution. The idea is to generate HTML5, CSS3 and JavaScript code out of mobile-specific IFML models, wrap it in the Cordova container and then send the zip folder containing the whole code to the Build PhoneGap online platform in order to get the Android application file (apk) and the iOS application file (ipa). The architecture is shown in the figure 14. The mobile application generation is available at two levels: (i) starting only from the domain model (a ER diagram describing the data model underlying the app); or (ii) starting from the IFML model. So the generator analyses the input model of the application, serialized as an XML file (XMI) with all the information about the application data and interactionand produces all the JavaScript and html files needed to run the mobile application. Our attention has been focused on the client side, since the server side consists of a traditional REST service, for which code generators are already available within WebRatio starting from IFML diagrams. The look and feel of the generated applications is based on a basic CSS3 file, which includes general rules for mobile interaction patterns. Eventually, if needed, it is possible to add other custom generation rules. In order to implement our prototype we used Groovy (a scripting language consisting of a simplified Java syntax, eventually converted in actual Java) and

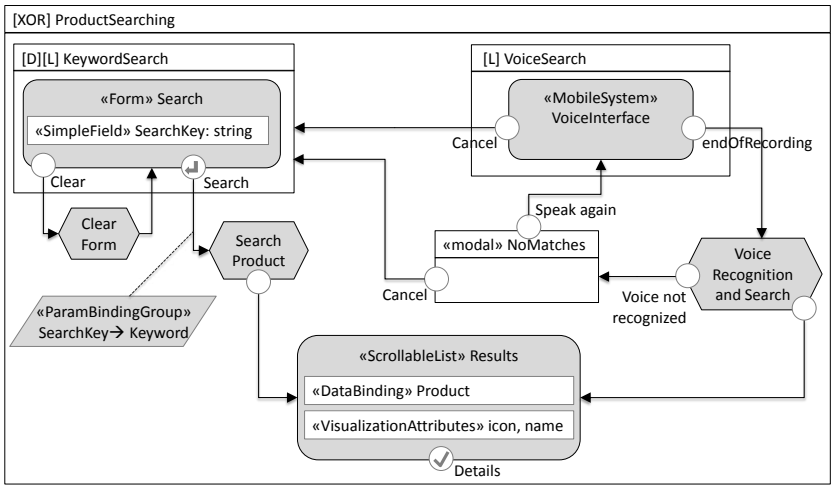

Fig. 11. Case study: IFML model corresponding to product searching shown in Figure 10. The model consists of input forms allowing the user to enter the keyword, Search Product actions and scrollable lists displaying the retrieved products.

ANT as generation engine. The ANT file provides all the required libraries for the generation, including Groovy and Dom4j for the model analysis and Jaxen, an open source XPath library. Moreover it provides some libraries required for the generated applications, like Backbone.js, Require.js, jQuery, Handlebars, Underscore.js and Spin.js. The ANT engine launches a main Groovy file, which in turn starts some Groovy templates in order to generate JavaScript and Html files.

Within this implementation work, we have also designed the metamodel of a few platforms (HTML 5, Java Swing, and the new Microsoft .Net WPF) and we have defined the mappings from the IFML abstract concepts to the platform-specific concepts of the three cases mentioned above.

6.3 Industrial Experience

The approach has been put at work on some industrial cases already. We report here on one of those cases, for which we also have some quantitative effort comparison with traditional development approaches. In the discussed scenario, the customer is a wholesale distributor of products for mechanical work-shops and tire repairers, with tens of thousands of different products in stock, for the whole European market. Products span from tires, to bolts, to large mechanical appliances for workshops such as gantry cranes.

The customer requested a mobile application to be deployed on tablets and cell phones (but to be used mainly on tablets) for its field agents, i.e., salesman that are assigned some geographical areas and go to customers for selling the goods. Requirements included: the need of the app to run completely offline; the

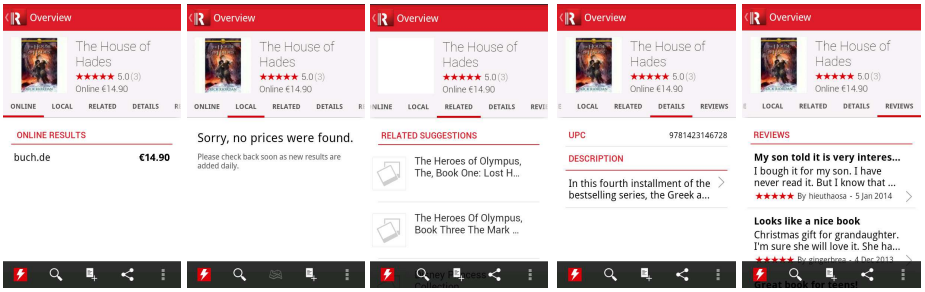

Fig. 12. *(a)*This screen shows the a list of the prices available *online* for the selected product. It is the default screen of *product overview*;*(b)*The *local* is accessed from the *online* and *related* screens by using *SwipeRight* and *SwipeLeft* gestures respectively; *(c)* Shows the list of other product *related* to the selected one. *(d)* Displays the details of the selected product;*(d)* Contains a list of *reviews* related to the selected product.

need of periodic or upon-request synchronization of the product catalog with the centralized copy of the data (including product information, photos, and technical sheets); dynamic calculation of prices based on the customer profile; and the possibility of defining custom versions of the catalog for specific types of customers.

The development of the application was performed by three teams of professional software developers, one focusing on the sever-side implementation of the REST services for managing the access to the centralized catalog data; one implementing the client-side of the mobile application manually; and one implementing it with our approach. Each team was composed by 2 developers, highly skilled for the respective technical needs. Both teams addressing the client side targeted cross-platform development through PhoneGap, with the same client-side architecture, JS libraries and functional and visual requirements.

The initial requirements specification implied 4 man-days of work (including discussions with the customer). The server-side development was performed using the Web Service modeling features of the WebRatio Platform. This resulted in a development effort of 9 man-days. On the client side, a common effort has been reported on the definition of the graphical styling of the app, which had to be extensively discussed and agreed upon with the customer. That part of the work accounted for 5 man-days. Finally, the client-side software development efforts have been:

– **manual development: 21 man-days**, organized as: 2 man-days on graphical style implementation and application to the code, 12 man-days on app structure and user interaction development, 7 man-days on client-server interaction and push notification implementation;

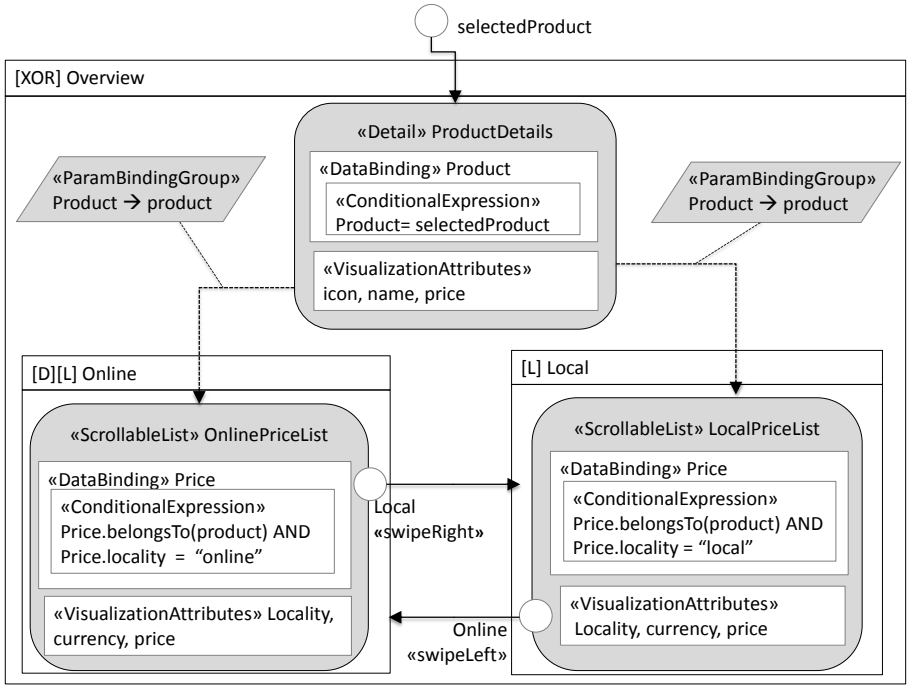

Fig. 13. Case study: IFML model corresponding to *Online* and *Local* tabs of product overview shown in figure 12. It consists of *ProductDetail* ViewComponent and two alternative lists of prices. The default one for online prices and the second displaying the local prices of the selected product.

– **model-driven development: 11 man days**, organized as: 1 man-day on graphical styling, 7 man-days on app structure design, and 3 man-days on client-server interaction design.

Furthermore, both client-side teams reported a testing period of 2 weeks. Thus, on the core mobile app development the model driven approach allowed to save 48% of the effort. Considering the total cost of analysis and development, this reduces the development cost of 21% overall.

A thorough comparison hasn't been possible on the other industrial cases because of the high cost of manual development of the applications. However, productivity in the other cases has been at the same levels.

7 Conclusions

In this paper we presented a mobile extension of OMG's standard IFML (Interaction Flow Modeling Language) for mobile application development. Our modeling of exemplary existing apps shows that the language is expressive enough to

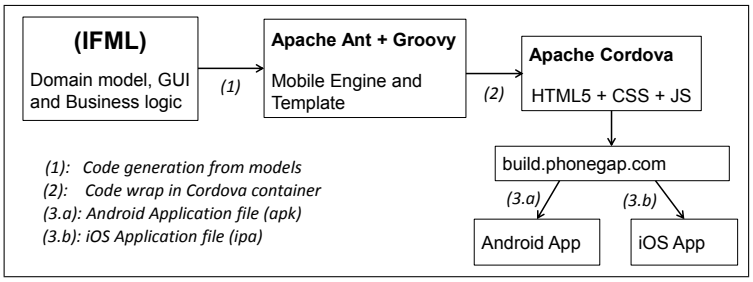

Fig. 14. Architecture for cross-platform mobile development followed in this research.

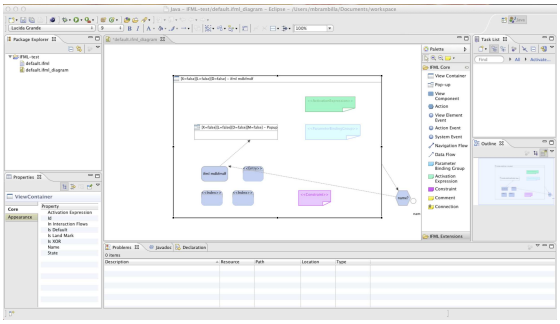

Fig. 15. Snapshot of the opensource IFML modeling tool developed in this research.

cover all the typical development needs on mobile. Furthermore, industrial experiences gave positive feedback on the applicability, effectiveness and efficiency of the approach. Future works will cover the implementation of more refined code generators and the study of design patterns for model-driven mobile applications design.

Acknowledgement. This work was partially funded by the AutoMobile EU 7th FP SME Research project (http://automobile.webratio.com).

References

1. Roberto Acerbis, Aldo Bongio, Marco Brambilla, Stefano Butti, Stefano Ceri, and Piero Fraternali. Web applications design and development with webml and webratio 5.0. In Richard F. Paige and Bertrand Meyer, editors, *TOOLS (46)*, volume 11 of *LNBIP*, pages 392–411. Springer, 2008.
2. Luciano Baresi, Franca Garzotto, Paolo Paolini, and Paolo Paolini. From web sites to web applications: New issues for conceptual modeling. In *ER (Workshops)*, pages 89–100, 2000.
3. Silvia Berti, Francesco Correani, Giulio Mori, Fabio Paternò, and Carmen Santoro. Teresa: a transformation-based environment for designing and developing multi-device interfaces. In *CHI Extended Abstracts*, pages 793–794, 2004.
4. Marco Brambilla, Piero Fraternali, and et al. The interaction flow modeling language (ifml), version 1.0. Technical report, Object Management Group (OMG), http://www.ifml.org, 2014.

5. Stefano Ceri, Piero Fraternali, Aldo Bongio, Marco Brambilla, Sara Comai, and Maristella Matera. *Designing Data-Intensive Web Applications*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers Inc., 2002.
6. J. Conallen. *Building Web applications with UML*. Addison Wesley, 2002.
7. Danny M. Groenewegen, Zef Hemel, Lennart C. L. Kats, and Eelco Visser. Webdsl: a domain-specific language for dynamic web applications. In Gail E. Harris, editor, *OOPSLA Companion*, pages 779–780. ACM, 2008.
8. Jaime Gmez, Cristina Cachero, Oscar Pastor, and Oscar Pastor. Conceptual modeling of device-independent web applications. pages 26–39, 2001.
9. Robert J. K. Jacob. A Specification Language for Direct-Manipulation User Interfaces. *ACM Trans. Graph.*, 5(4):283–317, 1986.
10. Karl R.P.H. Leung, Lucas C.K. Hui, S.M. Hui, and Ricky W.M. Tang. Modeling Navigation by Statechart. In *Proc. COMPSAC'00*, pages 41–47, 2000.
11. Marino Linaje, Juan Carlos Preciado, and Fernando Sánchez-Figueroa. A Method for Model Based Design of Rich Internet Application Interactive User Interfaces. In *Proceedings of International Conference on Web Engineering, July 16-20, 2007, Como, Italy*, pages 226–241, 2007.
12. Giansalvatore Mecca, Paolo Merialdo, Paolo Atzeni, Valter Crescenzi, and Valter Crescenzi. The (short) araneus guide to web-site development. In *WebDB (Informal Proceedings)*, pages 13–18, 1999.
13. Gerrit Meixner, Kai Breiner, and Marc Seissler. Model-driven useware engineering. In Heinrich Hussmann, Gerrit Meixner, and Detlef Zuehlke, editors, *Model-Driven Development of Advanced User Interfaces*, volume 340 of *Studies in Computational Intelligence*, pages 1–26. Springer, Heidelberg, 2011.
14. Fabio Paternò, Carmen Santoro, and Lucio Davide Spano. Maria: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. *ACM Trans. Comput.-Hum. Interact.*, 16(4), 2009.
15. David Raneburger, Roman Popp, Sevan Kavaldjian, Hermann Kaindl, and Jürgen Falb. Optimized GUI generation for small screens. In Heinrich Hussmann, Gerrit Meixner, and Detlef Zuehlke, editors, *Model-Driven Development of Advanced User Interfaces*, volume 340 of *Studies in Computational Intelligence*, pages 107–122. Springer Berlin / Heidelberg, 2011.
16. Daniel Schwabe, Gustavo Rossi, and Simone D. J. Barbosa. Systematic Hypermedia Application Design with OOHDM. In *Proc. Hypertext'96*, pages 116–128, 1996.
17. Daniel Schwabe, Gustavo Rossi, and Gustavo Rossi. The object-oriented hypermedia design model. pages 45–46, 1995.
18. Mathias Urbieta, Gustavo Rossi, Jeronimo Ginzburg, and Daniel Schwabe. Designing the Interface of Rich Internet Applications. In *Proc. LA-WEB'07*, pages 144–153, 2007.
19. Jean Vanderdonckt. A MDA-compliant environment for developing user interfaces of information systems. In *CAiSE*, pages 16–31, 2005.
20. Richard Vdovják, Flavius Frăsincar, Geert-Jan Houben, and Peter Barna. Engineering Semantic Web Information Systems in Hera. *Journal of Web Engineering*, 1(1-2):3–26, 2003.
21. Marco Winckler and Philippe Palanque. StateWebCharts: A Formal Description Technique Dedicated to Navigation Modelling of Web Applications. In *Proc. Intl. Workshop on Design, Specification and Verification of Interactive Systems*, pages 279–288, 2003.