Cross-Platform Development of Business Apps with MD^2

Henning Heitkötter and Tim A. Majchrzak

Department of Information Systems
University of Münster, Münster, Germany
{heitkoetter,tima}@ercis.de

Abstract. MD^2 is a framework for cross-platform model-driven mobile development. It consists of a domain-specific language for describing business apps concisely and of generators that automatically create complete iOS and Android apps from this specification.

Designers. MD^2 has been created as a research prototype at the Department of Information Systems, University of Münster. Henning Heitkötter and Tim A. Majchrzak, both interested in cross-platform approaches and mobile applications in general, supervised the implementation, which was mainly carried out by master students Sören Evers, Klaus Fleerkötter, Daniel Kemper, Sandro Mesterheide, and Jannis Strodtkötter.

Keywords: Business apps, mobile, cross-platform development, model-driven, domain-specific language, MDSD.

1 Introduction

Due to the rise of smartphones and tablets, more and more businesses are looking into mobile applications for their customers, employees, or business partners. The development of these so-called *apps* is a complex task of its own, as developers have to study a new environment and its unique capabilities. The heterogeneity of the mobile market further complicates app development. Most businesses target at least the two leading mobile platforms, i. e., iOS and Android (cf., e. g., [2]). Each platform increases development costs almost linearly due to profound differences in programming model, languages, and frameworks.

Cross-platform development approaches emerged to relieve developers from developing the same app repeatedly for different platforms. They allow implementing an app once and supplying several platforms from this common code base. A study by the authors [3] showed that existing approaches and solutions work well for certain situations. However, they do not achieve a truly native look & feel that resembles unique characteristics of each platform. This is often desired of high-quality apps or even a precondition for an app's functionality.

Our framework MD^2 employs a model-driven approach to cross-platform development of business apps. Developers specify an app using MD^2's domain-specific language (DSL). Generators transform this textual model to source code of iOS and Android apps. Our work makes several contributions to IS research:

J. vom Brocke et al. (Eds.): DESRIST 2013, LNCS 7939, pp. 405–411, 2013.

MD^2 is a novel, model-driven method for cross-platform app development. Its DSL represents an innovative set of constructs for specifying business apps.

Section 2 describes MD^2's design, i. e., the overall method and the set of constructs that forms the DSL. Its significance to research and to practice is analyzed in Section 3. Results from evaluating MD^2 are presented in Section 4, before Section 5 concludes with a short discussion and an outlook.

2 Design of the Artifact

MD^2 has been conceived and designed to address the lack of cross-platform solutions that produce apps with a native look & feel. App users and, consequently, businesses commissioning apps often desire a native appearance, as it matches the user interface (UI) of other applications on the device. Besides giving a more professional impression, such apps also integrate more closely with the respective platform. At the same time, MD^2 should considerably simplify the development of apps by increasing the abstraction level compared to programming languages such as Java or Objective-C. Section 3 illustrates the relevance of our goals.

MD^2 focuses on data-driven business apps. They are defined as mobile applications that serve a specific business purpose and mainly deal with form-based input, processing, and display of data. In contrast to, for example, mobile games, they mainly use standard UI elements and layouts. They retrieve, modify, and store structured data that may reside on the device or on remote servers. Their application logic needs to control the UI, validate input, and access device features such as GPS. Advanced calculations, however, are assumed to take place on servers. This set of requirements has been collected through close cooperation with industry partners and a previous industry survey [7] in order to ensure relevancy. It served as basis for deriving constructs needed to describe apps.

A model-driven approach fits well with the problem outlined above. Model-driven software development (MDSD) aims to make models a first-class artifact in the development process, from which lower-level representations, e. g., source code, can automatically be generated. These models can exhibit a higher level of abstraction, because subsequent transformations can introduce necessary detail when translating the constructs used in models into the target environment. Thanks to the specifically designed DSL, MD^2 models are concise, expressive, and easy to understand. Developers using MD^2 do not need in-depth knowledge of developing for iOS or Android. In particular, no knowledge of Objective-C or Java is required. They need to accustom themselves with MD^2's DSL, a task facilitated by its intuitive concepts and well-structured components.

Details on the implementation of MD^2 can be found in a separate paper [4]. Furthermore, MD^2's DSL has been described in detail as well [5]. In the following, we describe the architecture and features of MD^2 as they apply in an IS environment. They are the foundation for MD^2's contributions (see Section 3).

Development of an app with MD^2 (see Figure 1) may start with an initial idea of the app's software design. The app is described in a textual model using MD^2's DSL (phase 1). The DSL provides constructs for describing data model,

user interface, and control logic of an app in a declarative manner. These three parts are defined separately, but connected, as prescribed by the Model-View-Controller (MVC) pattern.

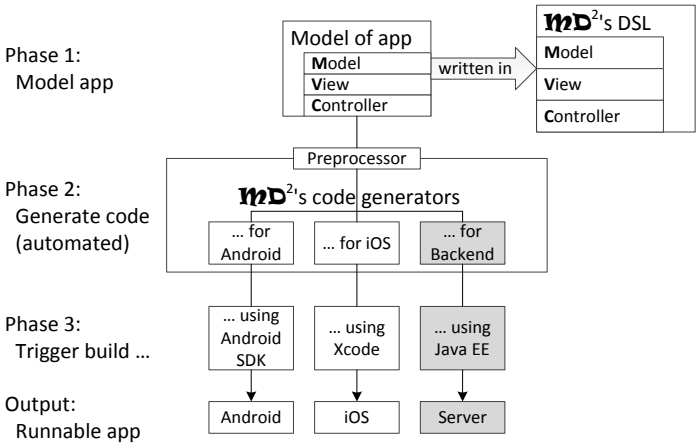

Fig. 1. Workflow and architecture of MD^2

The model part provides concepts such as entities with primitive-valued attributes and references to other entities. The modeler of an app instantiates these concepts to describe the app's data model. MD^2's view part includes standard UI elements such as text input fields, buttons, and labels. They can be arranged using nestable layouts, e. g., a tabbed navigation with several individual tabs. Modelers may define own styles and apply them to UI elements. The concept of an automatic generator inserts a default representation of an entity type and its properties.

The controller part of a model integrates model and view and specifies the behavior of the app. The integration is described using data mappings between UI elements and specific data stores. Essential behavioral constructs are events and actions, as apps built with MD^2 follow an event-based interaction model. In the controller model, custom actions are registered for certain events. Events range from simple user interactions, e. g., touching a button, to changes of the app's global state. For example, an action could be executed as soon as a certain condition involving data and UI state is true. Actions may consist of different kinds of predefined actions and also call other custom actions. There are predefined actions for event binding, data mapping, data operations, UI modification, and accessing device features. Additionally, MD^2 supports workflows for defining and restricting the navigation paths within the app. Implicit and explicit validators allow checking user input, for example, to ensure a particular format.

After the app – or a first version thereof – has been specified as a model, MD^2 invokes a preprocessor and, eventually, the code generators (phase 2). Each

generator inspects the model and creates native source code and additional files for its respective platform, i e., Android or iOS. The resulting project can directly be compiled without modifying the generated code (phase 3). The thus created native app can then be installed and executed on a suitable mobile device. Since each app directly uses the native source development kit of the respective mobile platform and native UI elements, generated apps are tightly integrated into their respective platform and inherently exhibit a native appearance.

Developers may change the model at any time and have the generators create a new version of the app following the same procedure. This enables a fast iterative development, as all phases except for the modeling are automated. Additionally, MD^2's backend generator creates a server application out of an app's data model, which provides apps with an interface for storing and retrieving data from a database.

MD^2 has been implemented as a set of plug-ins for the popular development environment Eclipse. Its DSL has been defined using Xtext, a language development environment [10] that also served to create powerful editors for MD^2 models. The code generators have been implemented in Xtend, a programming language with advanced code generation facilities [9].

3 Significance to Research and Practice

MD^2 is significant for both research and practice in that it is useful for practitioners yet addresses challenges in terms of fundamental and applied research.

While MDSD is well understood for developing desktop and server applications, its application to apps is rather new. In particular, understanding business needs for app development, identifying a set of requirements, and developing a corresponding DSL is no trivial task. Therefore, we provide not only a novel method for developing apps but also a set of constructs for describing apps as models. Developing such a DSL also fosters understanding of the domain. Our DSL highlights typical requirements and challenges of business apps.

The instantiation of our main artifact in form of a product connects the significance for research and practice. From a research perspective it proves the feasibility of our approach and the soundness of the underlying ideas. From a practical perspective it is the transfer of knowledge back from theory to practice.

Before work on our framework started, we participated in a project with regional companies that analyzed the status quo of app development and current challenges, especially regarding cross-platform development. Its results [7] showed the relevance of finding an adequate solution to tackle platform and device fragmentation and gave us insights into business needs. Additionally, we kept close contact with an IT consulting company that provided a pilot case for the first main prototype (see next section). In conclusion, MD^2 is highly significant and industry cooperation ensured its usefulness.

A multitude of cross-platform solutions, both from research and industry, also highlights the relevance of this topic. As demonstrated in a comparative study [3], existing approaches such as mobile web apps or hybrid apps – web apps packaged

with a native runtime environment – are useful in *some* scenarios. However, their very nature makes it difficult to develop apps with a native look & feel, because these approaches do not resort to native elements only. Instead, the look & feel of such Web-based apps resembles that of Web sites, which is undesirable if requirements demand a close native integration or a high-quality user experience.

As a model-driven approach promises to overcome these shortcomings, other solutions take this route as well. For example, *applause* [1] provides a DSL for apps and generates iOS, Android, and Windows Phone apps. However, in contrast to MD^2, applause is limited to displaying data. *AXIOM* [6] is another model-driven solution, but not fully automated, as the transformation requires manual intervention and mappings for each app. *Modagile Mobile* [8] uses a graphical DSL to describe data model and UI of apps and requires manual code for control logic.

4 Evaluation of the Artifact

We have evaluated MD^2 by applying it to real-world scenarios. Throughout development, an insurance tariff calculator served as a guiding example. It represents a typical form-based business app that implements a workflow. As the app uses a variety of UI elements and relies on a back end to provide data and perform calculations[1], it was well-suited as a pilot case. It even utilizes device features such as GPS to propose a location for a customer. Our industry partner, who supplied us with this case based on an actual project of theirs, also discussed requirements of business apps in general with us. To complete our set of apps for evaluating MD^2, we also used MD^2 to implement a number of smaller apps such as an e-commerce shop and a library front end.

For the current release, ease-of-use of MD^2 had to be assessed and resulting apps had to be evaluated. Due to the compactness and expressiveness of the language, it is relatively easy to implement apps using MD^2. The learning curve was found to be steep; no experience with model-driven approaches was necessary to use it. We believe that a good documentation will enable domain experts to understand and use our DSL. Regarding the actual apps, we got very positive feedback from our industry partner. Apps indeed have a native appeal and the desired functionality could be realized. This is particularly noticeable since some characteristics of apps have to be implemented quite differently on Android and iOS. We are confident that extending MD^2 to further platforms will not pose major technical difficulties.

We also evaluated MD^2 from a quantitative point of view. Lines of code (LOC) required for implementing apps are an adequate metric. 709 LOC in MD^2's DSL specified the tariff calculator. This lead to 10 110 LOC Java and 2 263 LOC XML for Android, 3 270 LOC Objective-C and 64 LOC XML for iOS, and 1 923 LOC Java and 57 LOC XML for the back end. Figures for other sample apps are

[1] Insurance companies typically do not want to expose their calculations and hence strive to keep as much logic on their (secured) servers as possible.

similar [4]. Since generated code resembles code as it would have to be written manually, the savings are more than notable.

The set of criteria expected from cross-platform approaches that has been established in a separate publication [3] can also be consulted to evaluate MD^2. MD^2 scores well with respect to most development criteria: It is accompanied by mature tooling such as an DSL editor; supports clearly structured, modularized, and concise models; and speeds up development in general. Regarding infrastructural criteria, it fulfills the requirements of a native look & feel, good performance, and access to platform-specific features. So far, only the two most popular mobile platforms are supported.

5 Conclusion

We have presented MD^2, our novel framework for cross-platform app development based on model-driven techniques. MD^2 is significant from a research perspective, as the way we apply a MDSD approach is new and the domain of business apps raises both technical and economic questions. At the same time, it is highly relevant. While first evaluation results are promising and most data-driven business apps can already be implemented, MD^2 is currently still limited with respect to certain functionality, for example advanced device features. Our work on MD^2 continues with refining it, extending it, and applying it to additional scenarios.

In parallel, we will continue to monitor the market of cross-platform development solutions. Future improvements of Web-based approaches, for example, in connection with the upcoming HTML 5 standard, might require a reevaluation of their potential regarding native integration. However, corresponding progress will most likely be limited to accessing more platform-specific functionality, but not extend to the look & feel.

Materials

MD^2's Web site (`http://wwu-pi.github.com/md2-web/`) provides access to the set of Eclipse plug-ins, MD^2's documentation and screencasts.

Acknowledgments. We gratefully acknowledge the work done by Sören Evers, Klaus Fleerkötter, Daniel Kemper, Sandro Mesterheide, and Jannis Strodtkötter, who implemented MD^2. Furthermore, we would like to thank viadee Unternehmensberatung GmbH for their support.

References

1. Behrens, H.: MDSD for the iPhone. In: Proc. of SPLASH (2010)
2. Gartner Press Release: Gartner Says Worldwide Mobile Phone Sales Declined 1.7 Percent in 2012 (February 2013), `http://www.gartner.com/newsroom/id/2335616`

3. Heitkötter, H., Hanschke, S., Majchrzak, T.A.: Evaluating cross-platform development approaches for mobile applications. In: Cordeiro, J., Krempels, K.-H. (eds.) WEBIST 2012. LNBIP, vol. 140, pp. 120–138. Springer, Heidelberg (2013)
4. Heitkötter, H., Majchrzak, T.A., Kuchen, H.: Cross-platform model-driven development of mobile applications with MD^2. In: Proc. 28th SAC, ACM (2013)
5. Heitkötter, H., Majchrzak, T.A., Kuchen, H.: MD^2-DSL – eine domänenspezifische Sprache zur Beschreibung und Generierung mobiler Anwendungen. In: Proc. Der 6. Arbeitstagung Programmiersprachen (ATPS), GI. LNI, vol. 215 (2013)
6. Jia, X., Jones, C.: AXIOM: A model-driven approach to cross-platform application development. In: Proc. 7th ICSOFT (2012)
7. Majchrzak, T.A., Heitkötter, H.: Development of mobile applications in regional companies: Status quo and best practices. In: Proc. 9th WEBIST (2013)
8. Modagile Mobile (2013), `http://www.modagile-mobile.de/`
9. Xtend (2013), `http://www.eclipse.org/xtend/`
10. Xtext (2013), `http://www.eclipse.org/Xtext/`