Computer Languages, Systems & Structures 53 (2018) 43–58

Contents lists available at ScienceDirect

Computer Languages, Systems & Structures

journal homepage: www.elsevier.com/locate/cl

A process-oriented modeling approach for graphical development of mobile business apps

Christoph Rieger[*], Herbert Kuchen

ERCIS, University of Münster, Leonardo-Campus 3, Münster, 48149, Germany

ARTICLE INFO

Article history:
Received 12 August 2017
Revised 30 November 2017
Accepted 9 January 2018

Keywords:
Graphical DSL
Mobile application
Business app
Model-driven software development
Data model inference

ABSTRACT

Mobile app development is an activity predominantly performed by software developers. Domain experts and future users are merely considered in early development phases as source of requirements or consulted for evaluating the resulting product. In the domain of business apps, many cross-platform programming frameworks exist but approaches also targeted at non-technical users are rare. Existing graphical notations for describing apps either lack the simplicity to be understandable by domain experts or are not expressive enough to support automated processing. The MAML framework is proposed as model-driven approach for describing mobile apps in a platform-agnostic fashion not only for software developers but also for process modelers and domain experts. Data, views, business logic, and user interactions are jointly modeled from a process perspective using a graphical domain-specific language. To aggregate multiple use cases and provide advanced modeling support, an inference mechanism is utilized to deduce a global data model. Through model transformations, native apps are then automatically generated for multiple platforms without manual programming. Our approach is compared to the IFML notation in an observational study, with promising results regarding readability and usability.

1. Introduction

A decade after Apple triggered the trend towards smartphones with its first iPhone, mobile devices and apps have been widely adopted. Business apps usually cover small-scale tasks and support the digitalization of processes which benefit from the increased mobility and availability of ubiquitous devices. For example, salespersons can access company data from a customer's location, expenses can be followed up remotely, and employees can submit requests for vacations not only from their office desk.

Until now, app development remains a task predominantly executed by programmers, often considering other stakeholders and future users primarily in requirements engineering phases upfront implementation. However, the research institute Gartner predicted that within a few years, more than half of all company-internal business apps will be created using codeless tools [1]. Many frameworks for *programming* mobile apps have emerged over the past years and cross-platform approaches allow for a large user base with low development efforts (an overview is given in [2–4]). Several commercial platforms provide cross-platform capabilities but usually focus on source code transformations, partly supported by graphical editors for designing individual views (e.g., [5,6]).

[*] Corresponding author.
E-mail address: christoph.rieger@ercis.de (C. Rieger).

https://doi.org/10.1016/j.cl.2018.01.001
1477-8424/© 2018 Elsevier Ltd. All rights reserved.

Modeling approaches that focus on platform-agnostic representations of mobile apps are rarely used in practice. Model-driven software development using a domain-specific language (DSL) bears the advantage of transforming a concise specification of the target application into a software product (semi-) automatically [7]. DSLs are generally suited to cover a well-defined scope with sensible abstractions for inherent domain concepts and increase productivity of developers compared to general purpose languages (GPL) [8]. Several textual DSLs for mobile apps have been presented in literature (e.g., [9–11]), although not all of them fully automate code generation [12]. However, *textual* DSLs provide only minor benefits to *non-technical users* because they still feel like programming [13].

Graphical modeling is therefore particularly suitable to describe apps by stakeholders with strong domain knowledge in order to better match the software product with their tacit requirements [14,15]. A suitable notation mandates a platform-independent design to also consider emerging mobile devices such as wearables, smart home applications, and in-vehicle apps.

To model sequences of activities, a wide variety of general-purpose process modeling notations such as Business Process Model and Notation (BPMN) exists [16]. Usually, those are not detailed enough to cover mobile-specific aspects and can hardly be interpreted by code generators from a technical point of view. In contrast, technical notations such as the Interaction Flow Modeling Language (IFML) are too complex to be understood by domain experts and require software engineering knowledge [17,18].

Moreover, the editor component is of major importance for the usability of a graphical modeling approach. Comparisons for graphical notations such as the Unified Modeling Language (UML) show that editors for the same notation differ significantly in modeling effort, learnability, and memory load for the user [19]. Editor development is a challenge in itself [20,21] and even the presence of a metamodel can only support the syntax of the resulting notation [22].

This article aims to alleviate the aforementioned problems by presenting the Münster App Modeling Language (MAML; pronounced 'mammal'), a *graphical DSL* for describing business apps that tackles the trade-off between technical complexity and graphical oversimplification in order to be understandable not only for software developers but also for domain experts and process modelers. *Model transformations* allow for a fully automatic generation of native smartphone apps for the Android and iOS platform from the specified graphical model without manually writing code. Besides the technical necessity of such a global model for code generation, it enables *advanced modeling support* for the graphical editor.

Four main research questions are addressed to investigate the potential of data model inference in the context of model-driven code generation approaches for mobile business apps:

(RQ1) How can user-oriented specification of business app functionality be achieved using a graphical modeling notation?
(RQ2) Is the modular subdivision of mobile app functionality feasible from a technical perspective with regard to the recombination of partial data models?
(RQ3) What additional support can data model inference provide for users to create semantically correct models?
(RQ4) Does the process-oriented subdivision of functionality help non-technical users in understanding and creating mobile app models?

Extending on previous work presented at SAC 2017 [23], this article presents the data model inference approach using a the scenario of an inventory management app and focuses on the benefits of such a mechanism for modeling environments regarding semantic semantic and validation (RQ3). In addition, the evaluation of MAML is extended to investigate the perceived usefulness of data model inference specifically for non-technical users (RQ4).

After presenting related work in Section 2, the proposed DSL and framework are presented in Section 3. Section 4 discusses the approach and presents evaluation results from a usability study before concluding in Section 5.

2. Related work

Since model-driven and cross-platform development of apps has been a topic for a few years now, there is plenty of scientific work on the general topic. However, only few graphical modeling approaches with subsequent code generation of mobile apps exist. Especially with regard to a workflow-related level of abstraction, related work on business process modeling is also considered in the following.

2.1. Cross-platform mobile apps

Developing mobile apps that run on multiple platforms can be achieved using different approaches. El-Kassas et al. [24] distinguish three major categories: *compiling* existing source code from a legacy application or different platform such as in [25], *interpreting* a single code base through a runtime or virtual machine such as Apache Cordova for developing hybrid apps [26], and *model-driven* generation of native app source code from a common representation. With regard to the latter category, various academic and commercial frameworks exist [12]. Only few of them, such as Mobl [9], PIMAR [27], and AXIOM [10] cover the full spectrum of runtime behavior and structure of an app; often providing a custom textual DSL for this means. The work in this article is based on MD^2 which focuses on the generation of business apps (i.e. form-based, data-driven apps interacting with back-end systems [4]). The input model is likewise specified using a platform-independent, textual DSL [11]. After preprocessing the model, code generators transform it into native platform source code for the Android and iOS platform [28,29]. Despite reducing development effort and complexity through domain-specific