Chapter 80
Migrating Software Towards Mobile Technologies

Liliana Maria Favre

https://orcid.org/0000-0003-1370-1861

Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina & Comisión de Investigaciones Científicas de la Provincia de Buenos Aires, Argentina

ABSTRACT

New paradigms such as pervasive computing, cloud computing, and the internet of things (IoT) are impacting the business world. Smartphones are at the core of these paradigms by allowing us interaction with the world around us. In light of this, it is imperative to migrate a lot of existing non-mobile software to adapt it to the new technological reality. The main challenge to achieve this goal is the proliferation of mobile platforms. An integration of ADM (Architecture Driven Modernization), cross-platform development and formal metamodeling to face this kind of migration is described. The proposal was validated with the migration of object-oriented software to different mobile platforms through the multiplatform language Haxe. A comparison of the approach with traditional migration processes and the description of existing challenges in real projects of the scientific and industrial field are included.

INTRODUCTION

Today, the use of technology is central to the business world. Organizations need to redesign their processes and models so as not to be left out of the market in the near future. The adoption of new digital technologies offers a competitive advantage and performs as a market differentiator. In particular, new paradigms such as Pervasive Computing, Cloud Computing and the Internet of Things (IoT) are impacting the business world. Smartphones are at the core of these paradigms, letting us locate and easily interact with the world around us. With the Smartphone through Wi-Fi and the 5G, we will handle all kinds of objects connected to the network. Just as smartphones have already displaced the camera, the GPS, the music player, and the wallet, they will be on-ramp for a new IoT revolution. In this scenario, humans and things act synergistically as a whole and smartphones will continue to play a crucial role due to they are the main interface connecting people to the Internet (Stancovik, 2014) (Islam and Want, 2015).

DOI: 10.4018/978-1-6684-3702-5.ch080

Smartphones are being transformed into a service center for different platforms in science, medicine, and education. Hossain et al. (2019) analyze the effects of variety-seeking intention by mobile phone usage on university students' academic performance.

In this context, most challenges for the competitive software industry are related to the problems caused by the proliferation of mobile platforms. New applications must support them as possible to remain profitable. Software applications can take full advantage of platforms only when they are built using native codebase. To address this problem a possible solution is to have different teams of developers who are fluent in a specific programming language to port an application to a specific platform. Instead of this traditional approach, organizations can use multiplatform or multi-paradigm cross-compiler based languages. The term "multiplatform" is used to refer source-source compilation, that is to say, the source code of these languages can be compiled into the source code of another programming language. Haxe is a good example of multiplatform languages. It allows using the same code to deploy an application on multiple platforms such as iOS, BlackBerry or Android. Specifically, it is an open-source high-level multiplatform programming language and compiler that can produce applications and source code for many different platforms from a single code-base (Haxe, 2019).

Frequently, the development of software component and applications aligned to mobile technologies requires adapting existing non-mobile software to different mobile platforms. For instance, there exist valuable software components and libraries implemented in C/C++or Java that need to be adapted for mobile developments. There is the need to define systematic, reusable migration processes with a high degree of automation that reduce risks, time and costs of the cross-platform development. Novel technical frameworks for information integration, tool interoperability, and reuse can help to achieve these goals. Specifically, Model- Driven Engineering (MDE) is a software engineering discipline which emphasizes the use of models and model transformations to raise the abstraction level and the degree of automation in software development (Brambilla et al., 2017). Productivity and some aspects of software quality such as maintainability or interoperability are goals of MDE.

Model Driven Developments (MDD) refer to forward engineering processes that use models as primary development artifacts. A specific realization of MDD is the Model-Driven Architecture (MDA) proposed by the Object Management Group (OMG) (MDA, 2019). Models, metamodels and model transformations play a major role in MDA. The MDA processes can be seen through a sequence of model transformations at different abstraction levels. A transformation is a process of converting a source model that conforms to a source metamodel in a target model that conforms to a target metamodel. The essence of MDA is Meta Object Facility (MOF), an OMG standard for defining metamodels that provides the ability to design and integrate semantically different artifacts in a unified way (MOF, 2016).

The OMG Architecture-Driven Modernization Task Force (ADMTF) is developing a set of specifications and promoting industry consensus on modernization. ADM is defined as "*the process of understand and evolve existing software assets for the purpose of software improvement, modifications, interoperability, refactoring, restructuring, reuse, porting, migration, translation, integration, service-oriented architecture deployment*" (ADM, 2019).

MDE progresses in the direction of achieving to develop software as reliable as any other engineering product, however, its impact on the software industrialization is limited. The main reasons are due, on the one hand, to a lack of repositories that contain reliable metamodels of the programming languages, and transformations between the different metamodels. It is worth considering that, MDE developers, instead of needing CASE tools to support the development on specific platforms, need to reuse metamodels and transformations. The incipient degree of progress in the specification of them requires that the developer,