Challenges in Model-Driven Development of Multi-Platform Augmented Reality Applications

Ivan Jovanovikj, Enes Yigitbas, Stefan Sauer, Gregor Engels
Paderborn University
Fürstenallee 11, 33102 Paderborn, Germany
{ivan.jovanovikj|enes.yigitbas|sauer|engels}@upb.de

Abstract

Augmented Reality (AR) is a technology that expands our physical world by adding layers of digital information onto it. Even though AR has been present for some time, its solutions have been widely used in high-end applications which are sometimes costly and usually not suitable for large scale deployments. Today, we have a lot of low budget mobile devices (smartphones and tablets) with high processing capacities to run AR applications, so it is important that we rethink how we develop AR applications for the newly introduced devices on the market. To address this problem, in this paper, we propose a model-driven development framework for multi-platform augmented reality applications.

1 Introduction

Augmented Reality (AR) is a user interface metaphor, which allows for interweaving digital data with physical spaces. According to [1], AR relies on the concept of overlaying digital data onto the physical world, typically in real-time in form of graphical augmentations. The recent technological advances and the rapid spreading and increased usage of Augmented Reality (AR) techniques in different domains (e.g., education, communications, medicine, entertainment etc.), rise the need of covering different platforms AR applications run on. However, due to the high number of different AR platforms (e.g., Android, iOS, HoloLens[1] etc.), different development framework (e.g., ARCore[2], ARKit[3], Vuforia[4] etc.) different AR-compatible devices (hand-held or head-mounted) makes the development of AR applications for different platforms a quite challenging task. Furthermore, the growing functional complexity of nowadays AR application regarding the complex structure (tasks, scenes) and composition (interrelations between real and virtual information objects) of AR applications, makes the development task even more challenging. Therefore, in this paper, we discuss the main challenges in developing multi-platform AR applications and propose a solution in terms of a model-driven development framework for multi-platform augmented reality applications.

[1] microsoft.com/hololens [2] developers.google.com/ar
[3] developer.apple.com/arkit [4] developer.vuforia.com

2 Challenges

Currently, there is not enough support by most AR device manufacturers for multi-platform interoperability between different AR enabled devices. Most of them only consider their own features and therefore provide support for libraries which can only be used to produce AR applications that run only on their products. Hence, we identified the following main challenges for development of multi-platform AR applications:

(C1) Selection and Implementation of Common AR Features. To develop a multi-platform AR application, it involves the use of different libraries as every AR device manufacturer has their own unique sets of libraries that propels their products. Fusing different predefined AR libraries into a single package solution presents a great challenge. This comes from the fact that most often these libraries are solely optimized for their targeted products with little or no support for other AR devices that could also perform similar functions as they do.

(C2) Hardware Components Initialization. Within a single session for different types of AR devices there is a specific hardware component initialization. Therefore, the framework needs to intuitively analyze its environment and initialize the appropriate hardware components which are necessary for the execution. However, components like the camera and button controls are completely different on handheld mobile devices and that of wearable devices like the HoloLens. There is always a problem if a wrong initialization takes place, it could result in a blank screen, frozen or a malfunctioned application on the respective AR device.

(C3) Code Generation. In general, writing different code to produce the same functionality for multiple AR enabled devices is not a pleasant experience. Most devices have varying architecture thus making the coded functionality incapable of running on multiple AR devices without some structural changes. Furthermore, the different platforms like Android, iOS, and Microsoft HoloLens, makes this even more challenging. So, creating multiple versions of an AR ap-

plication for multiple AR devices and platforms often leads to bad maintainability, sustainability, security etc. The developers, instead of focusing on their core strength, i.e., the functional requirements of their customers, they spend enormous amount of time making different versions for different AR enabled devices.

(C4) Task Execution. AR applications that are meant to support certain process as part of a training or maintenance scenarios, have to address all the tasks that are comprising that process. This means that these tasks have to be part of the application' s logic and at a given point of time, to be executed. However, doing this directly in the code, and not separating properly from the other aspects like data and scenes and other AR elements, could lead to a process difficult to manage. Furthermore, managing different data and managing different scenes that are somehow related logically could be quite complex if not managed from a central place.

3 Solution Idea

To address the previously introduced challenges, we propose a multi-platform development framework for augmented reality applications that relies on model-driven software development principles [2]. Based on our previous experience in the domain of model-driven development of AR applications [3], we propose the solution architecture shown in Figure 1. On the top, a model of an AR application is shown which is an instance of the Unified AR Modeling Language (UARML), a language we are still working on. As this language should cover different aspects of an AR application like processes, data, and scenes, it unifies three different languages namely, BPMN[5] (Business Process Modelling and Notation) for process, UCD (UML[6] class diagram) for data and IFML-AR (AR extension of the IFML[7] (Interaction Flow Modelling Language) for scenes. Hence, an UARML model consists of the following three models: *Process Model* specified in BPMN, a *Data Model* specified in UCD, and a *Scene Model* specified in IFML-AR. The central component, namely the *Multi-Platform Framework*, consists of the following components: *Common AR Capabilities*, *Hardware Initializer*, and *Code Generator*. In order to establish a set of common AR features, i.e., to address challenge *C1*, ARCore, ARKit, and Vuforia, are going to be analyzed in terms of their tracking capabilities (e.g., marker based tracking, model-based tracking, natural feature tracking etc.), their recognition capabilities (e.g., image or object recognition), etc. The collected features are going to be implemented as basic features of the framework as part of the *Common AR Capabilities* component. The *Hardware Initializer* component contains packages that are meant to serve as a codebase that

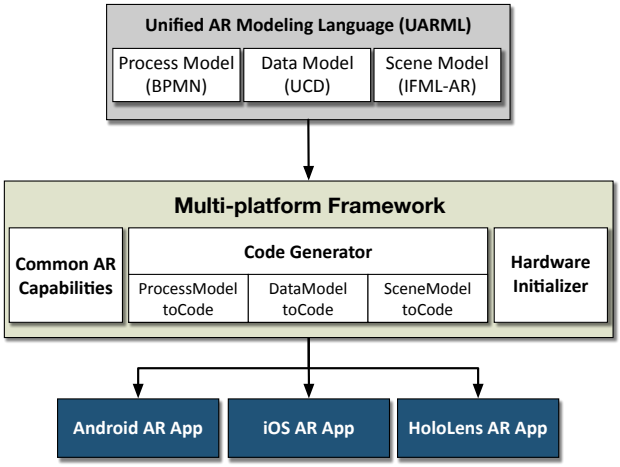

Figure 1: Solution Architecture for Multi-platform AR Application Development.

should reduce the work of developers regarding the initialization of the diverse hardware components on different devices (challenge *C2*). Technically, the *Hardware Initializer* consists of common Unity3D[8] components which provide support for camera initialization, options for text and button selection on different platforms, and also platform-specific code initialization which has been abstracted into a single container. The *Code Generator*, i.e., the corresponding sub-generators, take the input models to generate the final code for the three different platforms Android, iOS, and HoloLens, thus addressing challenge *C3*. The generated code contains process as well and therefore a task execution support is necessary, as defined in challenge *C4*. As most of existing workflow engine are quite powerful and require a lot of processing power, we build our own workflow engine which is capable of executing sequential activities as well as basic gateways like parallel or exclusive gateways.

References

[1] R. Azuma. A survey of augmented reality. *Presence*, 6(4):355–385, 1997.

[2] T. Stahl, M. Voelter, and K. Czarnecki. *Model-Driven Software Development: Technology, Engineering, Management*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2006.

[3] E. Yigitbas, I. Jovanovikj, S. Sauer, and G. Engels. A model-based framework for context-aware augmented reality applications. In *Handling Security, Usability, User Experience and Reliability in User-Centered Development Processes (IFIP WG 13.2 & WG 13.5 International Workshop @ INTERACT2019)*, 2019.

[5] bpmn.org [6] uml.org [7] ifml.org

[8] unity.com