Formalizing User Interaction Requirements of Mobile Applications

Shah Rukh Humayoun¹, Ragaad AlTarawneh¹, Yael Dubinsky²
¹Computer Graphics and HCI Group, University of Kaiserslautern, Germany
²IBM Research – Haifa, Mount Carmel, Haifa 31905, Israel
¹{humayoun, tarawneh}@cs.uni-kl.de, ²dubinsky@il.ibm.com

Provision of mobile user interaction, which consists of multi-touch gestures interaction and other direct interaction operations with the mobile device, makes the current mobile paradigm quite different from the conventional desktop paradigm. Writing the required mobile user interaction in an accurate and unambiguous form is one the main challenges in mobile application development. In this work, we propose to use MobiGolog language for writing formally the user interaction requirements of mobile applications (mobile apps). Moreover, we propose the idea of using layered graph representation for creating graph models of these formal written requirements in order to provide visual means to analyse the specified requirements.

Mobile environments. Mobile interaction. Multi-touch gestures. Graph models. MobiGolog

1. INTRODUCTION

The recent advancements in the mobile domain (such as the addition of multi-touch gestures, usage of sensors, multiple device sizes and platforms, single-task focused model, etc.) bring several new challenges for the development team in the different phases of building mobile apps.

Alongside others, one of the critical challenges that requirement engineers face nowadays is how to write accurately and unambiguously the user interaction with mobile devices and mobile apps. The current mobile user interaction paradigm is fundamentally different from the conventional desktop interaction paradigm, as it consists of multi-touch gesture interaction (e.g., tap, flick, swipe, pinching, etc.) and other direct interaction operation with the mobile device (e.g., through the accelerometer facility or through sensors) compared to the classic approach of mouse and keyboard selection in the desktop environment.

Targeting the above-mentioned concern, we propose to use MobiGolog (Mobile Task Modelling Golog) [Humayoun and Dubinsky 2014] for writing formally the user interaction requirements of mobile apps. MobiGolog is an extension to the TaMoGolog [Humayoun et al. 2014] formal task modelling language that was built on the top of the Golog family (e.g., Golog, ConGolog, IndiGolog, etc.) of language. One of the main capabilities of MobiGolog is that it enables to formalize the user gestures interaction with a specific mobile app or mobile device UI and the resulting effects based on this interaction. Few MobiGolog predicates relevant to user gestures interaction are: UserInteraction(𝑢) for defining a particular type of user interaction, PostconditionUserInt(𝑢, 𝑢, 𝜓(𝑢, 𝑢)) for defining the effects on variables due to a particular user interaction, UI-Element(𝑒) for defining a UI element, and mInteractionTask(𝑢, 𝑒, 𝑇) for defining which task(s) would execute based on a specific user interaction a particular UI element.

In this paper, first we propose to use MobiGolog for writing mobile user interaction requirements in an accurate and unambiguous due to the language’s preciseness and clearness. These written formal requirements in MobiGolog can also be used during the automated testing, which we proposed in [Humayoun and Dubinsky 2014], as a means to validate automatically the requirements. Second, we suggest graph models to visualize the requirements written in MobiGolog as a layered graph. This enables software teams not only to analyse the requirements effectively, but also they can use these graph models to compare with the developed mobile app graph models in order to validate the specified requirements.

2. WRITING MOBILE USER INTERACTION REQUIREMENTS

We provide a simple mobile app example to show formal modelling of user interaction requirements using MobiGolog.
Map-Viewing App: This app aims at providing the map with the zoom-in & -out functionality through two interaction types: direct-manipulating the map area (e.g., through pinching gesture) or through buttons. For each user interaction, the map scales up or down one size between 1 to 10 ranges.

Figure 1: Requirements specification of map-viewing app using MobiGolog

Figure 1 provides the user interaction requirements of the “map-viewing” using MobiGolog. It lists that the possible user interaction would be through three UI elements (i.e., a map area and two buttons “zoom+” & “zoom-”) using three types of interaction mode (i.e., pinchOpen, pinchClose, and tap). It lists two main required functionalities through unit tasks (i.e., zoom-in and zoom-out). It also explains that pinchOpen and pinchClose should work only on the map area, while the tap should work only with the zoom+ and zoom- buttons. The precondition axioms for both unit tasks tell that map should be zoomed-out and -in only, if it already has not reached to its minimum or maximum zoomed ratio. The four mInteractionTask predicates describe that which user interaction on a specific UI element will trigger a particular task execution. Finally, the proc definition of map-viewing task tells that user should be able to zoom-in or -out zero or more times.

3. THE GRAPH MODEL REPRESENTATION

In order to analyse the specified user interaction requirements throughout the development phase, we propose to model them using layered graph [Taentzer 2004, Kraft and Retkowitz 2006] representation (see Figure 2).

We propose to generate the graph representation based on the following: graph vertices are specific predicates useful for modelling user interaction, while the relations between these vertices are the graph edges. We classify these vertices into three layers (i.e., user interaction, UI elements, and unit tasks) based on the underlying predicate type. The edges are extracted using the predicate mInteraction from the formal specification. The precondition and postcondition nodes are represented as attributes attached to unit task nodes. In the current mobile domain, the outcome of interacting with a particular UI element depends on how a user interacts with it and that different interaction types with the same UI element can provide different results. This is the reason to have in/out ports in the nodes. In our model, the composite task is the subgraph that connects elements from these layers to achieve a specific goal. The resulted graph is represented as a Directed Acyclic Graph consists of three main layers, where each represents one vertices-type.

Figure 2: Layered graph model representation of the user interaction requirements of the map-viewing app

Our graph representation helps in tracing the path between the different layers of the graph elements. It can be used in analysing and comparing between the requirements graph model and the implemented code graph model for an automated validation. It can also be used as a visual communication tool between different groups. However, this graph representation model is still work in progress. In the future, we aim to develop an environment for automatically drawing of the graph model based on the formal requirement specifications written in MobiGolog, and vice versa.

5. REFERENCES


