Cross-platform Functional Consistency Validation for the Event-Driven Systems: An Ontology-Based Approach

Fahim T. IMAM [1]
School of Computing, Queen's University, ON K7K 3Z5, Canada
fahim.imam@queensu.ca

Introduction. Supporting multiple, heterogeneous platforms for the modern event-driven systems, e.g., mobile computing based applications, is a common requirement for any large-scale software engineering projects. When dealing with the functional behavior of these systems, an effective software engineering approach must tackle the following key challenges: (a) validating the level of functional consistencies between the cross-platform application systems; (b) maintaining a consistent co-evolution of the cross-platform functionalities based on the evolving requirements of the system's domain; and, (c) comprehending the functional changes that can be both human comprehensible and machine processable. The ultimate goal of this work is to develop an approach that can promote a machine-processable, intelligent decision-support mechanism for the kinds of functional reasoning that is required in order to mitigate these latter challenges for the Software Engineers. In order to achieve our goal, we consider the following research questions to be the critical driving force.

- How to effectively represent the software systems in terms of their functional behavior that can be both human comprehensible and machine processable?
- Can we incorporate the AI-based commonsense reasoning mechanisms for the functional behavior reasoning? What are the exact characteristics of the functional reasoning that is suitable for our required consistency validation mechanism?

Key Motivation. Despite the unavoidable heterogeneity issues with multiple platforms, the systems developed for those platforms must confirm that their intended set of functionalities are consistently implemented across the platforms. The tasks of maintenance in these systems can be quite expensive and challenging. An effective maintenance for the large-scale evolving systems must require a rigorous strategy for consistency management and change propagation across the overall system components. While the importance of software testing cannot be denied as part of the maintenance process, generating all possible test cases, and tracing the solution fragment for negative results, can be overwhelming. Contrary to software testing, we considered a more systematic, formal approach for the consistency management. However, our approach can help categorizing the set of test cases that must be executed in order to validate the functional consistencies between the cross-platform and evolving systems.

[1]The author of this research is supervised by Dr. Thomas R. Dean from the Department of Electrical and Computer Engineering at Queen's University, Canada. This research is supported by the NSERC, Canada.

Methodology. The idea is to utilize the powerful notion of ontology in order to mitigate the heterogeneity issues among the systems that are intended to have a set of identical functional goals. By representing the functional behavior of the systems into an ontology, we can have an effective mechanism to validate the levels of functional consistency among the systems. Since we are dealing with the event-based systems like mobile apps, the focus of the representation should be the functionalities that can be observed from its User Interface (UI) elements. Since the events associated with the UI elements are indicative of the system's behavior [1], these elements should provide the key source of knowledge regarding the functional model of the system. A set of such functional models should be captured at a level of abstraction that can be useful for our purpose. Once we have an ontology that can represent the functional models of the systems, different source code components of the involving systems can be annotated and mapped with that ontology. The ontological models will then represent a set of language-independent, comparable functionality models for each of the source systems. Next, the functionality models should be mapped and merged through the ontology. The ontology reasoner can then derive the implicit classification of functionalities, i.e., the logical functional decomposition of the source systems. Finally, as part of the reasoning process, the ontology reasoner would be able to detect the inconsistent set of functional features, along with the logical reasoning. To our best knowledge, there exist no direct, adequate research contributions that can be considered comparable to our approach.

Results. Based on the theory of *action*, *events*, and *change*, as understood from the literature of commonsense reasoning [2], event calculus, and functional reasoning [3], we have developed an ontology called the Event-based Functional Behavior Ontology (EFBO, `http://cs.queensu.ca/~imam/uifo.html`) for our purpose. The core classes of the EFBO are the *Event*, *Agent*, and the *Interface* classes. All the other entities within the EFBO are logically based on these three classes. The concept of *Event* within the EFBO essentially corresponds to the theory of SPAN [4], as endorsed by the Basic Formal Ontology (BFO). The concept of *Agent* and *Interface*, on the other hand, correspond to the theory of SNAP [4] within the BFO. The EFBO provides an effective representational facility to model the functional behavior of any event-based systems that can be comprehensible for both humans and machines. We have some preliminary results on modelling different systems using the EFBO. We are currently in the stage of annotating and mapping a number of cross-platform mobile apps that can be used to validate their functional consistencies based on the EFBO. We are also in the process of implementing a prototype application that can utilize our cross-platform functional validation approach in a semi-automated manner. The next step would be to develop a comparison metrics to verify the effectiveness of our approach.

References

[1] D. M. Hilbert and D. F. Redmiles. Extracting usability information from user interface events. *ACM Computing Surveys (CSUR)*, 32(4):384–421, 2000.

[2] E. Davis and G. Marcus. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, 58(9):92–103, 2015.

[3] B. H. Far and A. H. Elamy. Functional reasoning theories: Problems and perspectives. *AIE EDAM*, 19(02):75–88, 2005.

[4] P. Grenon and B. Smith. SNAP and SPAN: Towards dynamic spatial ontology. *Spatial cognition and computation*, 4(1):69–104, 2004.