Mobile applications testing based on Bigraphs and Dynamic feature Petri nets

Nguyen Thanh Binh[1], Le Thi Thanh Binh[2], Oum-El-Kheir Aktouf[3] and Ioannis Parissis[3]

[1]The University of Danang - Vietnam Korea University of Information and Communication Technology (VKU), Da Nang, Viet Nam,
ntbinh@vku.udn.vn
[2]The University of Danang - University of Science and Education, Da Nang, Viet Nam,
lttbinh@ued.udn.vn
[3]Univ. Grenoble Alpes, Grenoble INP, LCIS, Valence, France,
oum-el-kheir.aktouf@lcis.grenoble-inp.fr
ioannis.parissis@grenoble-inp.fr

Abstract. Nowadays, mobile smartphones are being widely used. They allow users to access a variety of services provided by mobile applications (mobile apps). These services are location-based services, meaning that a user's location is taken into consideration for service provision. Testing these mobile apps is challenging due to the complexity of context variability (i.e., a user's location). Current testing approaches cannot efficiently handle dynamic variability of mobile apps. To solve this problem, this paper introduces a model-based testing approach of mobile apps that uses a combination of a Bigraphical Reaction System (BRS) model and a Dynamic feature Petri net (DFPN) for automatic generation of test cases. Our model addresses the mobile app testing challenges related to the context of mobile apps, and especially to changes in the context location.

Keywords: Mobile applications testing, Bigraphs, Dynamic feature Petri nets.

1 Introduction

In recent years, more and more mobile apps have been developed to support different needs in news, social, tourism, business, health, and other domains. Hundreds of new apps are released and downloaded daily. By the end of year 2023, it is expected that the global revenue from the mobile app market will be $935 billion whereas almost 299 billion downloads of mobile apps will be performed [1].

Due to the increasing impact of mobile apps, there is an urgent need for a sound testing process. Indeed, testing mobile apps is a challenging research topic as testing a mobile app induces to validate it on many different contexts including different hardware platforms, operating systems, web browsers, many geographical locations for location-dependent apps, etc. All these varying features are referred to as the context of mobile apps.

This paper develops new testing techniques for automatic generation of test cases using models for context-aware mobile apps. We will mainly focus on addressing the context location variability.

The remaining parts of this paper are organized as follows: In section 2, we present main features of context-aware mobile apps. In section 3, we discuss related work. In section 4, we propose a test model for context-aware mobile apps. This model is illustrated on a case study in Section 5. Finally, section 6 concludes our approach and outlines future work.

2 Context-aware mobile apps

Modern mobile devices, with their mass of sensors, are capable of identifying many of the contexts, which surround the user. These sensors offer the opportunity to gain more information about the user's environment than before. Mobile apps that have these features are called context-aware.

Context awareness has become a trendy topic with mobile apps. When dealing with context, entities can be classified in three groups: people (groups, individuals), places (buildings, rooms, etc.) and things (computer components, physical objects, etc.). Each entity can be described by different attributes, which can be distinguished into four categories: location (an entity’s position, co-location, etc.), identity (each entity has a unique identifier), status (the properties of an entity, e.g., lighting or temperature for a room) and time (used to accurately define situation or ordering events, etc.) [2].

Mobile apps are increasingly incorporating functionalities which behaviors not only depend on explicit user input but also on the state of the surrounding environment. This type of mobile apps is known as context-aware mobile apps.

Context-aware mobile apps often consist of a middleware and a collection of services, and adapt their behavior according to the changing context [3]. They provide adaptive services responding to the dynamically changing contexts in the environment [4]. This means that, context-aware mobile apps use current context to provide proper services to the user.

3 Related work

Testing is extremely important and costly in the software development life cycle. To test context-aware apps, the tester must not only verify the system functions by providing user interactions, but he/she also has to provide these interactions according to different combinations of the system context.

For most test automation tools, test engineers develop test scenarios manually, only the execution of test cases is automated. Furthermore, model-based testing (MBT) is used to automatically generate test cases from the system according to a test model. Model based system engineering aims to focus efforts on modeling the solution instead of writing the code [3]. There are two key elements related to MBT approaches [5]: (i) Selecting proper models to describe the software behaviors (UML diagrams and finite state machines are often used to build models); and (ii) selecting criteria to generate test cases.

Currently, there are research works targeting context-aware applications and context-aware mobile apps testing. In particular, Siqueira et *al.* [6] made an exhaustive survey about this test field, where they describe the main papers on related techniques for automatic generation of test cases using model-based testing approaches. Achilleos et *al.* [7] defined three modelling components: Presentation model, Petri Net model, Context model. These components have been generated and integrated into the model-driven environment as Eclipse plug-ins. Seiger and Schlegel [8] created a test model based on DFPNs. Then, they defined context rules and integrated them into their model. Yu et *al.* [4] used two modelling notations, bigraphical labelled transition system and finite state machine. Test cases were generated by tracing interactions between the environment model and the middleware model. Griebe and Gruhn [9] defined a development model based on a meta model. Then, they transformed the development model into Petri net. Mirza and Khan [10] defined a development model as a UML activity diagram. Then, the development model is transformed into a function net.

On the basis of the above overview, we can see that test models based on adding context data directly to the behavioral model, as in the work of Griebe and Gruhn [9], Mirza and Khan [10] are complex and not suitable to model large mobile apps. Another test model where test cases are generated for changing environment and that uses data-flow-based coverage criteria, is adopted by Yu et *al.* [4] and seems to be an interesting idea. Indeed, separate models for the context and the behavior of the context-aware mobile apps, leads to models that are more manageable and simpler to modify and to extend. However, context aware mobile apps use the user's context to provide adaptive services. A service in service-oriented architecture may be an atomic service or a composite service [11]. Yu et *al.* [4] proposed a model-based testing approach based on bigraphical modeling for the context-aware mobile apps with an atomic service. The authors experiment on an airport application and an atomic illumination service. Nevertheless, this model is not implemented for context-aware mobile apps with a composite service. In fact, Yu et *al.* [4] used an extended finite state machine to model an atomic service. But, when the number of services in context-aware applications grows, then the number of states also increases and using such extended finite state machine model could be impractical. To overcome this limitation, we investigate in our work the use of DFPN instead of finite state machine, in combination with bigaph model.

4 A test model for context-aware mobile apps

Based on the above analysis, we propose a novel test model for context-aware mobile apps with both atomic and composite services. Our model comprises two separate diagrams: a bigraph for modeling the context and a DFPN for modeling the behavior of the context-aware mobile apps.

4.1 Bigraph

Bigraphs are graphs with nodes and edges. The nodes may be nested, representing locality and the edges link the nodes. Details about the notion of bigraph can be found in [12][13][14].

Example: The TripAdvisor application provides a set of services (Tourist Spots, Hotels, Restaurants) which fit to the user's location. Let's consider an example of a simulated environment, modeled as a bigraph in Fig. 1.

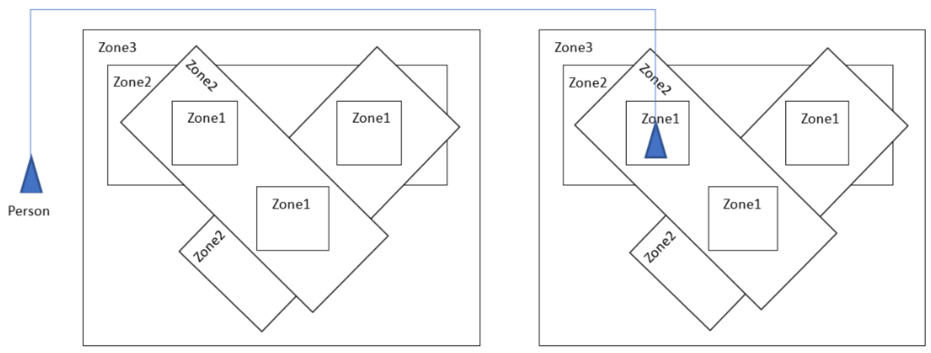


Fig. 1. A bigraph for modeling in a simulated environment of the TripAdvisor application

There are two structures on the nodes of a bigraph, they may be nested, and they may also be connected by links. The linkage is independent of the nesting, so links often cross node boundaries [12]. We can observe the particular bigraph in Fig. 1, as follows:

- Three regions (rectangles): Zone 3 is a city of a country. Zone 2 is a district belonging to the city. Zone 1 is a ward belonging to the district.

- A person: a human. A person may enter or leave another Zone.

Each model using bigraphs is represented by two parameters: a signature that makes the bigraph represent formal entities of the model and a set of reaction rules to represent their behaviors[14]. Reaction rules in a bigraph model are used to present the dynamics in the form r -> r', meaning that r can change its state to r' in suitable contexts. These changes may involve both placing and linking. The reaction rules of the TripAdvisor application are presented in Table 1.

4.2 Dynamic feature Petri net model

Dynamic feature Petri net is an extension to Petri nets. A basic Petri net consists of places which are represented by circles. Each place may be marked by tokens that are represented by dots. Transitions (rectangles) and arcs (arrows) connect transitions and places. A marking may be created by putting tokens into places. A transition is activated when all places are marked with tokens. The state of a Petri net can be changed by letting a transition consume tokens from input places and produce new ones in output places. Multiple places can be marked in the same state [15].

In DFPN, transitions are additionally annotated in the following way [15]:

Transitions = (application condition) / (update expression) (1)

An application condition is a Boolean logical formula on a set of features. An update expression describes how the feature selection evolves after the transition.

The execution of a DFPN based on the feature selection is called application condition and the mechanism to update the feature selection is called update expression.

DFPN extends Petri net by adding a feature selection to the state of the Petri net marking transitions with application conditions and update expressions.

4.3 Interaction among models

Middleware is in between the environment and a set of services. Get and Put are two interfaces which are defined between middleware and context environment. The middleware uses Get interface to get context information from the environment, and Put interface to take action on the environment. The middleware sends a request to a service and acquires return results if any through a call interface between the middleware and services. The interaction between the middleware and the environment is modeled as the Cartesian product of models.

5 Case study

To provide a proof of concept of our proposed approach for context-aware mobile apps testing, we applied it to test the TripAdvisor application which is introduced in section 4.1.

TripAdvisor provides location-based travel services, which helps travelers to search tourist Spots, Hotels and Restaurants based on their locations. We consider the following scenario: There is 1 service (e.g., Tourist Spots service or Hotels service or Restaurants service) in Zone 1, there are 2 services e.g., (Tourist Spots service and Hotels service) or (Hotels services and Restaurants service) or (Tourist Spots services and Restaurants service) in Zone 2 and there are 3 services (e.g., Tourist Spots service and Hotels service and Restaurants service) in Zone 3.

5.1 Modeling the environment with a Bigraph

There are 15 reaction rules shown in Table. 1, describing the behaviors of the user and interactions with the environment.

Table 1. Descriptions of 15 reaction rules of the TripAdvisor application.

No.	Description
1	User enters Zone 1 having one service (Tourist Spots service).
2	User enters Zone 1 having one service (Hotels service).
3	User enters Zone 1 having one service (Restaurants service).
4	User moves from Zone 1 having one service (Restaurants service) to Zone 2 having two services (Tourist Spots service and Hotels service).

5	User moves from Zone 1 having one service (Restaurants service) to Zone 2 having two services (Restaurants service and Hotels service).
6	User moves from Zone 1 having one service (Restaurants service) to Zone 2 having two services (Restaurants service and Tourist Spots service).
7	User moves from Zone 1 having one service (Tourist Spots service) to Zone 2 having two services (Restaurants service and Hotels service).
8	User moves from Zone 1 having one service (Tourist Spots service) to Zone 2 having two services (Tourist Spots service and Hotels service).
9	User moves from Zone 1 having one service (Tourist Spots service) to Zone 2 having two services (Tourist Spots service and Restaurants service).
10	User moves from Zone 1 having one service (Hotels service) moves to Zone 2 having two services (Tourist Spots service and Restaurants service).
11	User moves from Zone 1 having one service (Hotels service) moves to Zone 2 having two services (Hotesl service and Restaurants service).
12	User moves from Zone 1 having one service (Hotels service) moves to Zone 2 having two services (Hotels service and Tourist Spots service).
13	User moves from Zone 2 having two services (Hotels service and Tourist Spots service) to Zone 3 having 3 services (Tourist Spots service, Hotels service and Restaurants service).
14	User moves from Zone 2 having two services (Hotels service and Restaurants service) to Zone 3 having 3 services (Tourist Spots service, Hotels service and Tourist Spots service).
15	User moves from Zone 2 having two services (Tourist Spots service and Restaurants service) to Zone 3 having 3 services (Tourist Spots service, Hotels service and Restaurants service).

Reaction rule r0 (see Fig. 2): The user is outside of the Zone (C0) and moves into Zone 1 having one service {Tourist Spots service (C1) or Hotels service (C2) or Restaurants service (C3)}.

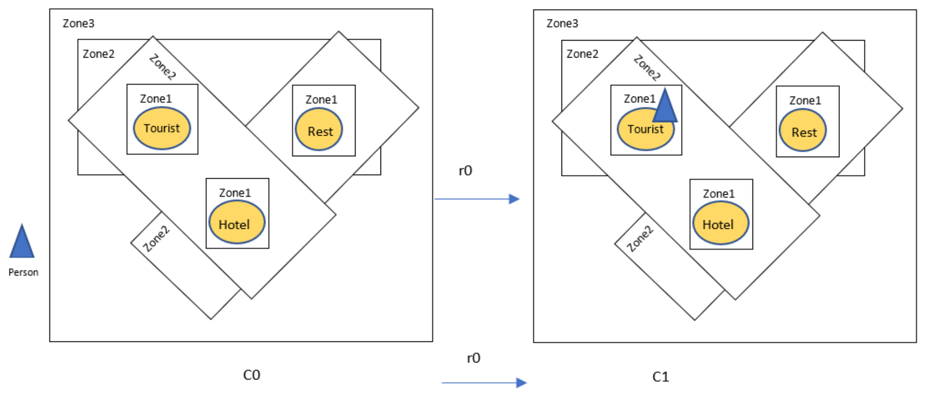


Fig. 2. Reaction rule r0

Reaction rule r1: User moves from Zone 1 having one service {Tourist Spots service (C1)} moves into Zone 2 having two services (including pre-existing service) {Tourist Spots service and Hotels service (C4)} or {Tourist Spots service and Restaurants service(C5)}.

Reaction rule r2: User moves from Zone 1 having one service {Tourist Spots service (C1)} moves into Zone 2 having two services (pre-existing services are not included) {Hotels service and Restaurants service (C6)}.

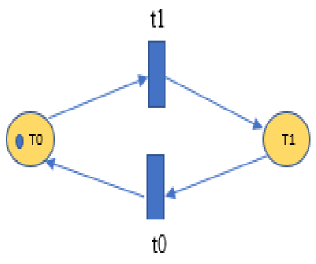
Reaction rules r3: User moves from Zone 2 having two services {Tourist Spots service and Hotels service (C4)} into Zone 3 having 3 services {Tourist Spots service, Hotels service and Restaurants service (C7)}.

5.2 Modeling the middleware with a DFPN

Context-aware mobile apps include the environment and the system under test (SUT). There are two components in the SUT: middleware and services. The environment obtains context information. The SUT obtains a set of services for the middleware to invoke. The interactions between the environment and the SUT are implemented through the get and put interfaces. We add these two interfaces to the transitions of DFPN as in [8]. We model the TripAdvisor application with a DFPN as follows:

We describe this application with a set of services: Tourist Spots service (T), Hotels service (H), Restaurants service (R) and the location changing of the user among Zones. Each service has 2 states and 2 transitions respectively: Tourist Spots service (T) has 2 states of no service (T0) and service (T1) and transitions t0 and t1. For instance, Fig.3 shows the DFPN for Tourist Spots service.

.



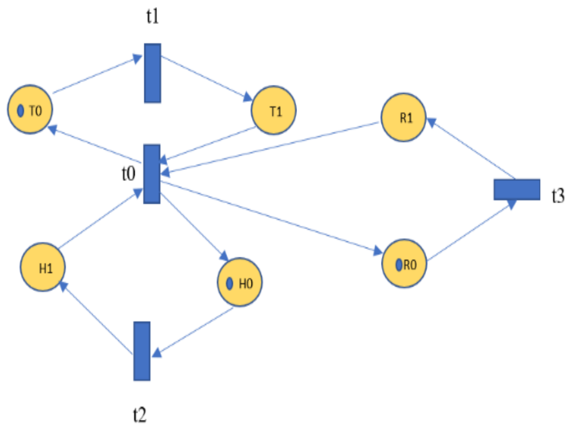


Fig. 3. DFPN for Tourist Spots service. **Fig. 4.** DFPN for the TripAdvisor application

Basically, a transition is activated if its input states are marked. In the beginning, the token may be consumed by two transitions t0 and t1, if transition t1 fires, the token moves from state T0 to state T1, and Tourist Spots service is enabled. Symmetrically, when token moves from state T1 to state T0 (t0 fires), Tourist Spots service is disabled. Similarly, Hotels service (H) has 2 states of no service (H0) and service (H1) and transitions t0 and t2. Restaurants service (R) has 2 states of no service (R0) and service (R1) and transitions t0 and t3. Three services in TripAdvisor application can be modeled with DFPN (see Fig. 4).

5.3 Assessments

In order to evaluate our approach on the TripAdvisor app, we use three different criteria (all paths, all-uses, and all-defs) as in [4].

We implement our test model with test situation 1: The user is outside of the Zone and moves into Zone 1 having one service (Tourist Spots service); then he/she moves into Zone 2 having two services (Tourist Spots service and Hotels service); then he/she moves into Zone 3 having 3 services (Tourist Spots service, Hotels service and Restaurants service).

An interesting bigraph in Fig. 5 and the reaction rules in Table 1 constitute an example of a bigraphical reactive system. Applying the matching of these reaction rules, a bigraph label transition system is generated.

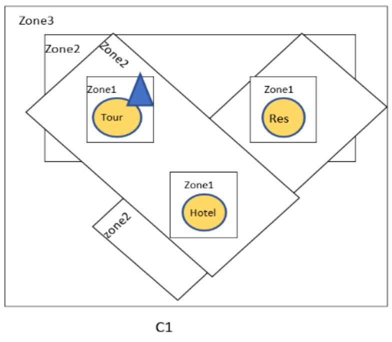


Fig. 5. A bigraph of interest.

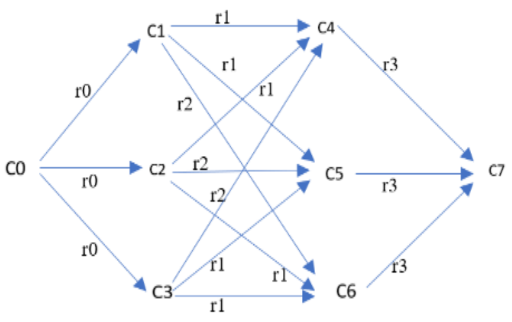


Fig. 6. Bigraph label transition system

We model this test situation 1 with bigraph as shown in Fig. 6 where reaction rules as labels indicated as r0 to r3 and bigraph as states indicated as C0 to C7.

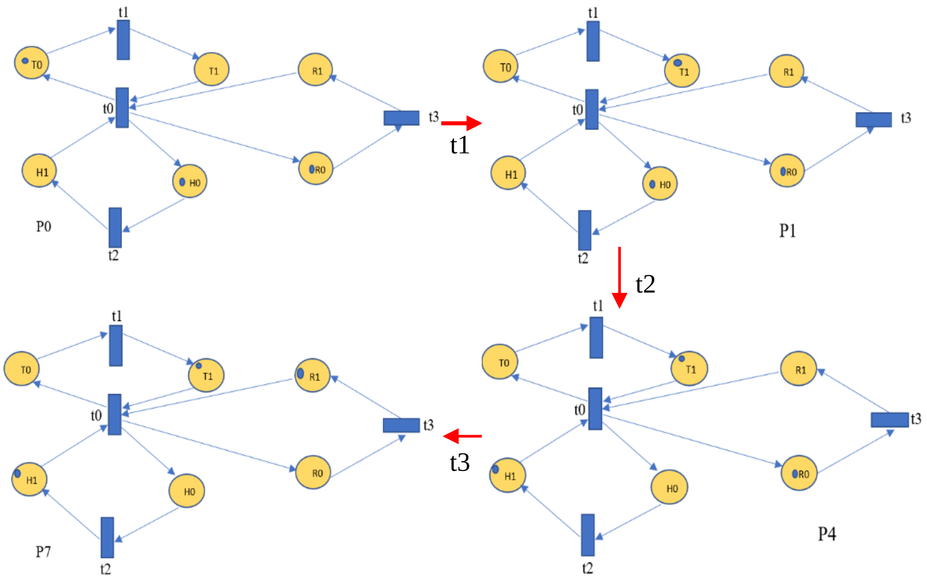


Fig. 7. DFPN for test situation 1

We continue to model this test situation 1 with DFPN as shown in Fig. 7, where transition as labels indicated as t0 to t3 and DFPN as states indicated as P0 to P7. In particular, the user is outside of the Zone (P0). The user is in Zone 1 having one service {Tourist Spots service (P1) or Hotels service (P2) or Restaurants service (P3)}. The user is in Zone 2 having two services {Tourist Spots service and Hotels service (P4)} or {Tourist Spots service and Restaurants service (P5)} or {Hotels service and Restaurants service (P6)}. The user is in Zone 3 having 3 services {Tourist Spots service, Hotels service and Restaurants service (P7)}.

Then, we provide the combination between BLTS and DFPN with test situation 1. This synchronization model is modeled as a Cartesian product between them. We have the following result:

$$C0P0 \xrightarrow{t1} C0P1 \xrightarrow{r0} C1P1 \xrightarrow{t2} C1P4 \xrightarrow{r1} C4P4 \xrightarrow{t3} C4P7 \xrightarrow{r3} C7P7$$

The BLTS in Fig. 6 may contain a large number of reaction paths. There are too many possible test-case sequences in the synchronized model. So, we use testing strategies based on pattern-flow based testing in [4] to select test cases by selecting a set of reaction paths such that the set of paths cover all-defs or all-uses.

Regarding pattern-flow based testing, r0 is an example of pattern def of the bigraph of interest as shown in Fig. 5 where the user enters Zone 1 with Tourist Spots service; r1 is a pattern def-use, r2 is a pattern use, r3 is a pattern def-use. Test cases are generated from test situation 1 as follows:

1. $C0 \xrightarrow{r0} C1 \xrightarrow{r1} C4$
2. $C0 \xrightarrow{r0} C1 \xrightarrow{r1} C4 \xrightarrow{r3} C7$
3. $C0P0 \xrightarrow{t1} C0P1 \xrightarrow{r0} C1P1 \xrightarrow{t2} C1P4 \xrightarrow{r1} C4P4 \xrightarrow{t3} C4P7 \xrightarrow{r3} C7P7$

Test situation 2: The user is outside of the Zones and moves into Zone 1 having one service (Tourist Spots service), then he/she moves into Zone 2 having two services (Restaurants service and Hotels service), then he/she moves into Zone 3 having 3 services (Tourist Spots service, Hotels service and Restaurants service).

Modeling this test situation 2 with bigraph is also shown in BLTS in Fig. 6.

We model this test situation 2 with DFPN as follows:

$$\begin{tikzcd}
C0P0 \arrow[r, "t1"] & C0P1 \arrow[r, "r0"] & C1P1 \arrow[r, "t0"] & C1P0 \arrow[r, "r2"] & C6P0 \arrow[r, "t2"] \arrow[dr, "t3"'] & C6P2 \arrow[r, "t3"] & C6P6 \arrow[r, "t1"] & C6P7 \arrow[r, "r3"] & C7P7 \\
& & & & & C6P3 \arrow[ur, "t2"'] & & &
\end{tikzcd}$$

Test cases are generated from test situation 2 as follows:

1. $C0 \xrightarrow{r0} C1 \xrightarrow{r2} C6$
2. $$\begin{tikzcd}
C0P0 \arrow[r, "t1"] & C0P1 \arrow[r, "r0"] & C1P1 \arrow[r, "t0"] & C1P0 \arrow[r, "r2"] & C6P0 \arrow[r, "t2"] \arrow[dr, "t3"'] & C6P2 \arrow[r, "t3"] & C6P6 \arrow[r, "t1"] & C6P7 \arrow[r, "r3"] & C7P7 \\
& & & & & C6P3 \arrow[ur, "t2"'] & & &
\end{tikzcd}$$

Context-aware mobile applications provide adaptive services in response to the dynamically changing contexts in the environment. In the TripAdvisor application, its environment comprises of a range of moving user and physical facilities. Physical facilities include Zone 1, Zone 2, Zone 3 and moving user may refer to the user who may move through Zones. Providing suitable services to end user according to change of the environment is hard. So, testing these applications is challenging. This paper handles this problem by using the proposed approach on the test situations of the TripAdvisor application. That means, when the user moves through Zones (Zone 1, Zone 2, Zone 3), the adaptive services (Tourist Spots service, Hotels service and Restaurants service) are provided according to the user's movement.

6 Conclusion and future work

Context-aware mobile applications interact with the environment in real time, which keeps on changing. Therefore, one challenge of context-aware mobile applications testing is to generate test cases for the changing environment. This paper proposes to use BRS as an environmental model and DFPN as a middleware model. By synchronizing the BRS and DFPN, test cases are generated to verify the interactions between the environment and the middleware. Our novel test model for context-aware mobile apps that uses a combination of BRS model and DFPN solves the challenge of changing context, especially location context. In the future, we will implement our test model with testing tools to assess our approach.

References

1. https://www.statista.com/statistics/269025/worldwide-mobile-app-revenue-forecast/, last accessed 2021/11/20.
2. Dey, A. K., Abowd, G. D.: A conceptual framework and a toolkit for supporting rapid prototyping of context-aware applications. Journal Human-Computer Interactions 16(2-4), 7–166 (2001).
3. Baldauf, M., Rosenberg, F., Dustdar, S.: A survey on context-aware systems. In: International Journal of Ad Hoc and Ubiquitous Computing, pp. 236-277. Geneva, Switzerland (2007).
4. Yu, L., Tsai, W.-T., Perrone, G.: Testing context-aware applications based on bigraphical modeling. pp. IEEE Transactions on Reliability 65, 1584–1611 (2016).
5. Mehmood, M. A., Khan, M. N. A. and Afzal, W.: Transforming context-aware application development model into a testing model. In: 8th International Conference on Software Engineering and Service Science, pp. 177-182. IEEE Press: Beijing, China (2017).
6. Siqueira, B. R., Ferrari, F. C., Souza, K. E., Camargo, V. V., Lemos, R. J. S. T.: Testing of adaptive and context-aware systems: approaches and challenges. Verification and Reliability 1772, 1-46 (2021).
7. Achilleos, A. P., Kapitsaki, G. M., and Papadopoulos, G. A.: A framework for dynamic validation of context-aware applications. In: 15th International Conference on Computational Science and Engineering, pp. 532-539. IEEE, Cyprus (2012).
8. Puschel, G., Seiger, R., Schlegel, T.: Test modeling for context-aware ubiquitous applications with feature petri nets. In: 2nd Workshop on Model-Based Interactive Ubiquitous Systems, pp. 37–40. ACM: Copenhagen, Denmark (2012).
9. Griebe, T., Gruhn, V.: A model-based approach to test automation for context-aware mobile applications. In: 29th Annual ACM Symposium on Applied Computing, pp. 420–427. ACM: New York, USA (2014).
10. Mirza, A. M., Khan, M. N. A.: An automated functional testing framework for context-aware applications. IEEE Computer 6(1), 46568–46583 (2018).

11. http://docs.oasisopen.org/wsbpel/2.0/CS01/wsbpel-v2.0-CS01.html, last accessed 2021/10/21.
12. Milner, F.: The space and motion of communicating agents. 2nd edn. Cambridge Univesity Press, UK (2009).
13. Yu, L., Gao, J.: Generating test cases for context-aware applications using bigraphs. In: 8th International Conference on Software Security and Reliability, pp. 137–146. IEEE Press: San Francisco, USA (2014).
14. Milner, R.: Pure Bigraphs: Structure and Dynamics. Information and Computation 204(1), 60–122 (2006).
15. Muschevici, R., Clarke, D., Proenca, J.: Feature petri nets. In: 14th International Conference, pp. 13-17. Jeju , Korea (2010).