Test Criteria for Context-aware Mobile Applications

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

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

Abstract. Context-aware mobile apps provide adaptive services that depend on the changing environments. The challenge in testing context-aware mobile apps holds in that these apps adapt their behavior when context conditions are changing. According to an exhaustive survey of the testing context-aware mobile apps research area, current testing approaches take into consideration the apps functions and different contexts but do not consider testing coverage criteria nor evaluate the coverage of various situations. Our work is intended to fill this gap. In our previous research work, we presented a test model for context-aware mobile apps dealing with changing location context based on the combination of a Bigraph Reaction System and a Dynamic Feature Petri Net. In this paper, we propose a new test criterion for context-aware mobile apps. This criterion results from the combination of pattern-flow-based coverage criteria and boundary-based coverage criteria.

Keywords: Test Criteria, Context-Aware Mobile Apps Testing, Model-Based Testing.

1 Introduction

Today, mobile phones have a huge impact on people lifestyle. They allow users to run mobile applications (apps) and access mobile Internet services at anytime, anywhere. Thanks to their high portability, they are very useful to access information wherever the users go through GPS-enabled devices and mobile networks. Context-aware mobile apps provide adaptive services responding to the dynamically changing contexts in the environment [2]. As a result of the availability of mobile devices and mobile network infrastructures, context-aware mobile apps can position the mobile device on the Earth and use its current location to provide suitable services to the user.

Testing context-aware mobile apps is challenging due to the high interaction complexity between the applications and their environments [3]. In our previous paper [1], we proposed a test model for context-aware mobile apps dealing with changing location context. The model consists of: 1) describing the static structure of the environment by using place graphs and link graphs, called bigraphs and modeling the dynamic behaviors of the environment by reaction rules, which indicate the changes of the contexts in the environment, a collection of bigraphs enriched with reaction rules is called a Bigraph Reaction System (BRS); 2) using a Dynamic Feature Petri Net (DFPN) to model the middleware (the system is assumed to include a middleware and a set of services); 3) combining BRS and DFPN to describe the interactions between the environment and the system. Additionally, our work [1] introduced a pattern-flow testing criterion [2] to select test paths. However, while experimenting context-aware mobile apps testing with the proposed test model, we encountered some issues when selecting test data set. This paper proposes a new test criterion which eases the selection of test data sets and the coverage of test situations. It results from the combination of pattern-flow-based coverage criteria and boundary-based coverage criteria.

The remaining parts of this paper are structed as follows. Section 2 presents the background on test criteria. Section 3 introduces a case study used to illustrate the proposed approach. Section 4 briefly introduces the test model for context-aware mobile apps presented in [1]. Section 5 proposes a new test criterion for context-aware mobile apps. Section 6 provides an experimental analysis of the effectiveness of the proposed test criterion. Conclusion and future work are given in section 7.

2 Background

In this section, we introduce the definitions related to location contexts. These definitions are prerequisites to develop the selection of test data sets for context-aware mobile apps.

Definition 1. A location $l_i = (long_i; lat_i)$ is a pair of real numbers which refer to the longitude longi and the latitude lati of the location on the Earth surface [9].

Definition 2. A path $path_i(l_{i1}; l_{i2}; \ldots; l_{ik})$ is defined as a sequence of locations [9].

Definition 3. $P_x(long_x; lat_x)$ is the location information of a service $service(P_x)$ which is known as Point of Interest (POI) [7].

Definition 4. A test case is a sequence of input stimuli to be fed into a system and the corresponding expected behavior of the system. A test case comprises abstract test cases and concrete test cases.

Definition 5. An abstract test case consists of abstract information about the sequence of inputs and outputs. The missing information is often concrete parameter values or function names. Abstract test cases are often the first step in test case creation. They are used to get an idea of the test case structure or to get information about satisfied coverage criteria.

Definition 6. A concrete test case is an abstract test case plus all the concrete information that is missing to execute the test case. Concrete test cases comprise the complete test information and can be executed on the system under test (SUT).

Definition 7. Control-flow-based coverage criteria are defined on the basis of a control flow graph representing a program.

Definition 8. Data-flow-based coverage criteria are focused on the data flow of variables. Expressions can define and use variables. A def is a location where a value for a variable is stored into memory. A use is a location where a variable's value is accessed.

Definition 9. Boundary-based coverage criteria [4] are constraints that specify value partition of objects. Objects that satisfy a value partition are said to be inside/outside the partition.

3 Case study

The TripAdvisor application illustrated on Fig. 1 is an example of context-aware mobile app. This app provides Tourist Spots service, Hotels service, and Restaurants service, which help travelers to search suitable services based on their locations.

Fig. 1. The Tripadvisor application.

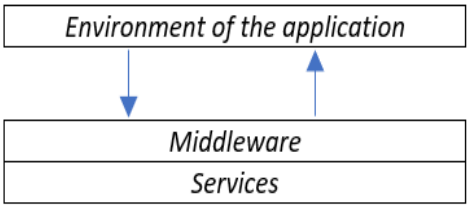


Fig. 2. Architecture of context-aware mobile apps.

This application can be used by every body in the world. App users can be in different countries, different cities, different districts, and so on. In order to ease the application testing, these areas are named with symbols: Zone3 is a city of a country; Zone2 is a district belonging to the city; Zone1 is a ward belonging to the district; and so on.

We consider the following scenario: There is 1 service (e.g., Tourist Spots service or Hotels service or Restaurants service) in Zone1, there are 2 services e.g., (Tourist Spots and Hotels service) or (Hotels and Restaurants service) or (Tourist Spots and Restaurants service) in Zone2 and there are 3 services (e.g., Tourist Spots and Hotels and Restaurants service) in Zone3.

The following test situation will be examined: The user is outside the Zones and moves into Zone1, which has one service (Tourist Spots service); then he/she moves into Zone2, which has two services (Tourist Spots and Hotels service); then he/she moves into Zone3, which has 3 services (Tourist Spots, Hotels and Restaurants service).

4 Test model for context-aware mobile apps

Context-aware mobile apps provide adaptive services responding to the dynamically changing contexts in the environment. Context-aware mobile apps often consist of a middleware and a collection of services.

Based on the architecture of context-aware mobile apps (Fig. 2), we proposed a test approach that consists of three phases: modeling the environment of the application; modeling the middleware and the services; and combining the two models above to verify the interactions between the environment and the application [1].

4.1 Modeling the environment of the context-aware mobile application

The environment of applications comprises a range of physical facilities, moving entities and sensors or wireless connections to backend systems. For example, in the TripAdvisor application, physical facilities include Zone1, Zone2, Zone3; moving entities may refer to the user who may move through Zones; and wireless connections to backend systems.We proposed to describe the static structure of the environment by bigraphs and to model the dynamics of the environment by reaction rules.

Describing the static structure of the environment. A bigraph consists of two graphs: a place graph that captures notions of locality or containment and a link hypergraph that models connectivity or associations. Therefore, bigraphs are a natural way to model the containments like physical facilities, connections and moving entities of context-aware mobile app.

Fig. 3 shows a bigraph, which describes physical facilities (static structure of the environment) of the TripAdvisor application. It consists of three regions (rectangles): Zone3 is a city of a country, Zone2 is a district belonging to the city, Zone1 is a ward belonging to the district. A user may enter or leave a Zone.

Modeling the dynamics of the environment. In bigraph models, the mobility of entities in the environment (the dynamics of the environment) is expressed as reactions, with sets of reaction rules that give possible ways in which a system might be reconfigured. The reaction rules describing the user's mobility through Zones of the Tripadvisor application are presented as follows.

Reaction rule r0 (see Fig. 4): The user is outside of the Zone (C0) and moves into Zone1 having one service {Tourist Spots service (C1) or Hotels service (C2) or Restaurants service (C3)}.

Reaction rule r1: User moves from Zone1 having one service {Tourist Spots service (C1)} into Zone2 having two services (including pre-existing service) {Tourist Spots and Hotels service (C4)} or {Tourist Spots and Restaurants service (C5)}.

Reaction rule r2: User moves from Zone1 having one service {Tourist Spots service (C1)} into Zone2 having two services (pre-existing services are not included) {Hotels service and Restaurants service (C6)}.

Reaction rules r3: User moves from Zone2 having two services {Tourist Spots service and Hotels service (C4)} into Zone3 having three services {Tourist Spots service, Hotels service and Restaurants service (C7)}.

We select a bigraph of interest to represent the initial state of the environment and the set of reaction rules as above. Then, by matching reaction rules against the current bigraph, a Bigraph Label Transition System (B-LTS) is created with bigraphs as states and reaction rules as labels.

We model the test situation presented in Section 3 by the bigraph and reaction rules as shown in Fig. 5, where reaction rules are labels from r0 to r3 and bigraphs are states from C0 to C7.

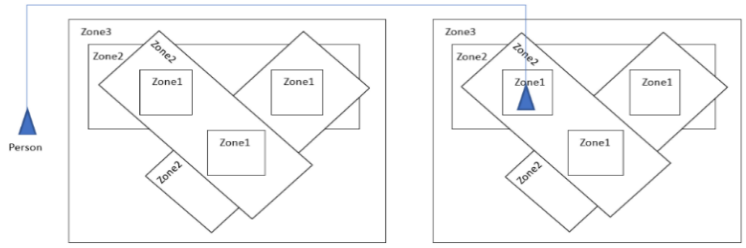


Fig. 3. A bigraph for modeling the TripAdvisor application in a simulated environment

4.2 Modeling middleware and services

Middleware is between the environment and a set of services. The middleware often provides three functions: 1) collecting context information from the environment; 2) analyzing and reasoning about the situation; and 3) selecting and invoking proper services to react, which can be atomic or composite services.

Context-aware mobile apps are based 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. [2] proposed a model-based testing approach based on bigraphical modeling for the context-aware mobile apps with an atomic service. They experimented their model on an airport application and an atomic illumination service. Nevertheless, this model is not implemented for context-aware mobile apps with composite services. In fact, Yu et al. [2] used an extended finite state machine to model an atomic service. However, when the number of services in context-aware mobile apps 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 [1] the use of Dynamic Feature Petri Nets (DFPN) instead of finite state machines, in combination with the bigraph model.

We describe the TripAdvisor application with the following 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. 6 shows the DFPN for Tourist Spots service.

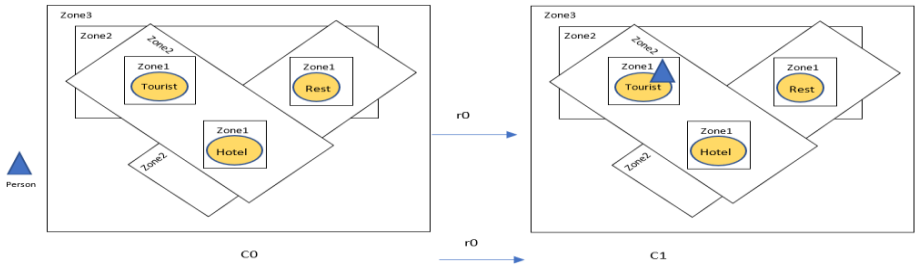


Fig. 4. Reaction rule *r*0.

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, then 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. These three services in the TripAdvisor application can be modeled with DFPN (see Fig. 7).

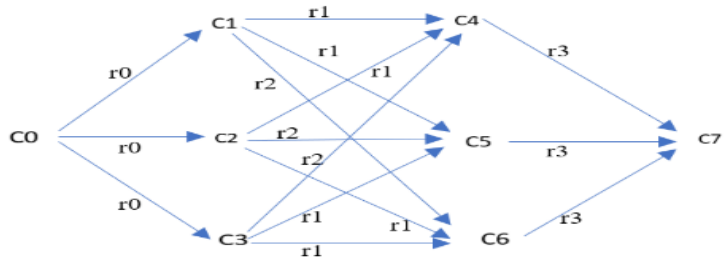


Fig. 5. B-LTS.

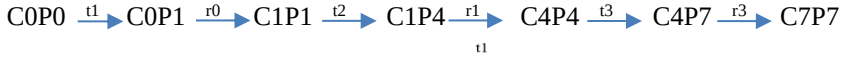
We model the test situation with DFPN as shown in Fig.12 where transition are labels from t0 to t3 and DFPN are states from P0 to P7.

4.3 Combining two models to verify the interaction between the environment and the application

The combination of BRS and DFPN as a Cartesian product result in a synchronized model. Let B be B-LTS model for the context-aware environment, and D be DFPN model for the context-aware middleware. The interaction between context-aware environment and middleware is modeled as B × D, such that

The vertex set S is the Cartesian product B × D. It is a finite and non-empty set of states.

Any two vertices (b, d) and (b', d') ∈ S are adjacent in S if and only if b = b' and d is adjacent with d' in D, or d = d' and b is adjacent with b' in B. The result below shows an example of the interactions between the B-LTS in Fig. 5 and the DFPN of the test situation.



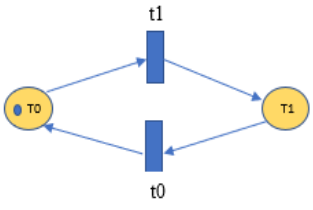


Fig. 6. DFPN for Tourist Spots service.

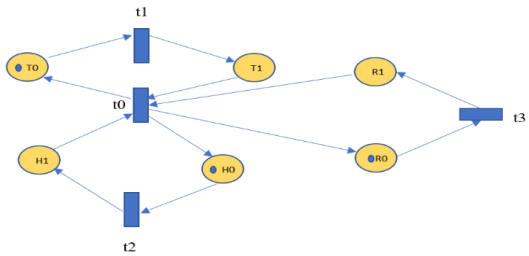


Fig. 7. DFPN for the TripAdvisor application.

5 Test criteria for context-aware mobile apps

5.1 Test strategy based on bigraphical pattern-flow testing

B-LTS may contain a high number of paths. In our previous work, we applied a bigraphical pattern flow testing strategy like in [4] to select a subset of paths. This strategy is similar to data-flow testing strategy as described in [6], but defined on reaction rules in terms of bigraphs instead of variables in data-flow approaches.

In bigraph terms, a reaction rule has the form R -> R′ where R is known as the redex and R′as the reactum. A pattern is a sub-structure in a redex or a reactum of a reaction rule. If a pattern does not appear in redex of reaction rule R, but appears in the reactum, it is called the pattern definition. If a pattern appears in redex of reaction rule R, but does not appear in the reactum, it is called the pattern use. If a pattern appears both in redex and reactum of reaction rule R, it is called the pattern def-use.

A path in a B-LTS is a sequence of $C_0r_0 \ldots Ciri$ from an initial state to a final state of B-LTS, where Ci stands for bigraph and ri for reaction rule. We select a set of reaction paths such that the set of paths meets all-defs, which ensures that each pattern definition reaches at least a pattern use, or all-uses, which ensures that each pattern definition reaches all pattern uses. Abstract test cases are paths which are generated from the B-LTS in Fig. 5 and the synchronized model with the test situation of TripAdvisor application as follows.

All-defs. Every def reaches a use.

1. *r0-r1* <=> *C0* $\xrightarrow{r0}$ *C1* $\xrightarrow{r1}$ *C4*
2. *r1-r3* <=> *C1* $\xrightarrow{r1}$ *C4* $\xrightarrow{r3}$ *C7*

All-uses. Every def reaches all possible uses.

1. *r0-r1* <=> *C0* $\xrightarrow{r0}$ *C1* $\xrightarrow{r1}$ *C4*
2. *r0-r2* <=> *C0* $\xrightarrow{r0}$ *C1* $\xrightarrow{r2}$ *C6*
3. *r0-r1-r3* <=> *C0* $\xrightarrow{r0}$ *C1* $\xrightarrow{r1}$ *C4* $\xrightarrow{r3}$ C7
4. *r0-r2-r4* <=> *C0* $\xrightarrow{r0}$ *C1* $\xrightarrow{r1}$ *C5* $\xrightarrow{r3}$ *C7*
5. *r1-r3* <=> *C1* $\xrightarrow{r1}$ *C4* $\xrightarrow{r3}$ *C7*

All-du-paths. All the paths between defs and uses are executed.

1. *r0-r1* <=> *C0* $\xrightarrow{r0}$ *C1* $\xrightarrow{r1}$ *C4*
2. *r0-r1* <=> *C0* $\xrightarrow{r0}$ *C1* $\xrightarrow{r1}$ *C5*
3. *r0-r2* <=> *C0* $\xrightarrow{r0}$ *C1* $\xrightarrow{r2}$ *C6*
4. *r0-r1-r3* <=> *C0* $\xrightarrow{r0}$ *C1* $\xrightarrow{r1}$ *C4* $\xrightarrow{r3}$ *C7*
5. *r0-r1-r3* <=> *C0* $\xrightarrow{r0}$ *C1* $\xrightarrow{r1}$ *C5* $\xrightarrow{r3}$ *C7*
6. *r0-r2-r4* <=> *C0* $\xrightarrow{r0}$ *C1* $\xrightarrow{r1}$ *C5* $\xrightarrow{r3}$ *C7*
7. *r1-r3* <=> *C1* $\xrightarrow{r1}$ *C4* $\xrightarrow{r3}$ *C7*
8. *r1-r3* <=> *C1* $\xrightarrow{r1}$ *C5* $\xrightarrow{r3}$ *C7*
9. *C0P0* $\xrightarrow{t1}$ *C0P1* $\xrightarrow{r0}$ *C1P1* $\xrightarrow{t2}$ *C1P4* $\xrightarrow{r1}$ *C4P4* $\xrightarrow{t3}$ *C4P7* $\xrightarrow{r3}$ *C7P7*

5.2 Test strategy based on boundary values testing

We explore here the opportunity of using boundary-based coverage criteria [10] as a means to build such partitions and, hence, to help narrowing the input data space.

Boundary values are used to select concrete values from partitions. In test process, faults are often detected around boundaries [13]. Therefore, boundary values should be considered to increase the faults detection. The concrete value selection for partitions are described in [14] as follows: For each partition boundary of a partition, a value should be selected inside the partition close to the boundary, outside the partition close to the boundary or on the boundary. Our approach focuses on values which are selected inside the partition to the boundary.

The location information (including latitude and longitude coordinates) of mobile end user or POIs can be obtained through the Global Navigation Satellite Systems (GNSS), the Geographic Information System (GIS) and the Wireless Communication (WC) technologies [8]. We propose the use of POIs to define partition boundaries for zones in the test model as below. For each POI in a zone, we consider its real-world coordinates $P_i = (long_i; lat_i)$ as provided from GIS, GNSS or WC [8].

Definition 10. Radius R_i (along the real-world coordinates) is maximum distance between the user and a POI. That means approximately Ri(km)/111(km) degree difference along the longitude and the latitude coordinates on the Earth surface [12].

For example, consider the POIs located of a service within 1km from the user's GPS location. That means approximately 1km/111km= 0.009-degree difference along the longitude and the latitude coordinates.

Definition 11. The circle $circle_i\ (P_i; R_i)$ denotes a circle, the center of the circle is Pi, and the radius of the circle is R_i (along the longitude and the latitude coordinates).

Definition 12. The partition boundaries of the zone having 1 service $service(P_i)$ is a circle $circle_i$.

Definition 13. The partition boundaries of the zone having n services $(service(P_1), service(P_2); \ldots; service(P_n))$ denotes the intersection between *n* circles $(circle_1; circle_2; \ldots; circle_n)$.

5.3 A novel coverage criterion based on pattern-flow and boundary values

We combine a boundary-based criterion and a pattern-flow-based coverage criterion to build a new improved coverage criterion for context-aware mobile apps testing. This criterion makes it possible to define input partitions based on the boundaries of Zones for abstract test cases that are generated from bigraphical pattern-flow-based coverage criteria of test model, as explained in section 4.

Abstract test cases are generated from the test model and bigraphical pattern-flow based testing in section 4, which comprise abstract information about input parameters (e.g., the sequence of positions of the user when he/she moves through Zones). We define partition boundaries for each zone. These partition boundaries are used to select specific information for each abstract test case (e.g. select a proper current coordinates position of user, which is inside partition boundaries of zones when he/she moves through zones). That means, the abstract test cases satisfy the chosen structural, e.g.,

bigraphical pattern-flow based, coverage criterion. For each abstract test case, proper input values can be selected to satisfy a chosen boundary-based coverage criterion. The result of this approach is a test suite that satisfies a combination of both kinds of coverage criteria.

6 Experiment

We consider context information as location context, and we select test Zones first. TripAdvisor application has been used in many countries and cities, we only test TripAdvisor application in an area of Valence, France as an experiment. We implement the proposed test criterion to the test model presented in section 4.

We test the TripAdvisor application at an area of Valence, France which has location coordinates including longitude from 43.38611 to 43.38736 and latitude from -1.66303 to -1.66113.

There are 38 services in the selected area with 9 services of interest (Tourist Spots, Hotels, Restaurants services). We define the partition boundaries of Zones corresponding to these services, which are proposed in the test model.

We call P_T the location coordinates of service Tourist Spots, P_H the location coordinates of service Hotels and P_R the location coordinates of service Restaurants. Location coordinates of services POIs include (P_T, P_H, P_R). From the area of Valence, we take the location coordinates of services (POIs):

P_T. $P_{Sas\ Luz\ Voyages}$(43.38645; -1.66146)}.

P_H. {$P_{Hotel\ Relais\ Saint}$(43.38629; -1.66158), $P_{Brit\ Hotel\ de\ Paris}$(43.38624; -1.66118), $P_{Saint\ Jean\ de\ Luz\ rentals}$(43.38662; -1.66075) }.

P_R. {$P_{Boulangerie\ Magri\ Christophe\ Ogi\text{-}labea}$(43.38635; -1.66169), $P_{Le\ Komptoir\ des\ Amis}$(43.38652; -1.66033), $P_{Le\ Bar\ a\ Foie\ Gras}$(43.38693; -1.66205), $P_{Kako\ Etxea}$(43.38716; -1.66097), $P_{Chez\ Pablo}$(43.38705; -1.66016)}.

Suppose that we use the radius of 50 meters. We calculate the partition boundaries of Zone1, Zone2, Zone3 as follows.

Zone1. Zone has 1 service, and is a circle (center is a POI's location, radius of 50 meters = 0.00045 degree longitude and latitue).

Tourist spot service (P_T). $P_{Sas\ Luz\ Voyages}$(43.38645; –1.66146). Zone1 has Tourist spot service in Zone having longitude from ($longP_T - R$) to ($longP_T + R$) and latitude from ($latP_T - R$) to ($latP_T + R$) equal to longitude from (43.38645 – 0.00045) = 43.38600 to (43.38645 + 0.00045) = 43.3869 and atitude from (-1.66146 - 0.00045) = -1.66191 to (-1.66146 + 0.00045) = -1.66101.

Hotel service (P_H). $P_{Hotel\ Relais\ Saint}$(43.38629; -1.66158). Zone1 has Hotel service in Zone having longitude from 43.38584 to 43.38674 and latitude from -1.66203 to -1.66113.

Restaurant service (P_R). $P_{Boulangerie\ Magri\ Christophe\ Ogi\text{-}labea}$(43.38635 ; -1.66169). Zone1 has Restaurant service in Zone having longitude from 43.3859 to 43.3868 and latitude from -1.66214 to -1.66124.

Zone2. Zone has 2 services, and is the intersection of 2 circles (2 Zone1 as in Fig. 8).

$C1$ is Zone1 has Tourist spot service $P_{Sas\ Luz\ Voyages}$. $C1$ is a circle (center is $P_{Sas\ Luz\ Voyages}$, radius is 50 meter) which has equation:

$$x^2 + y^2 - 86.7729\,x + 3.32292y + 1885.14449 = 0 \quad (1)$$

$C2$ is Zone1 has Hotel service $P_{Hotel\ Relais\ Saint}$. $C2$ is circle (center is $P_{Hotel\ Relais\ Saint}$, radius is 50 meter) which has equation:

$$x^2 + y^2 - 86.77258x + 3.32316y + 1885.13101 = 0 \quad (2)$$

Intersection of $C1$ and $C2$ is the results of two equations (1) and (2):

$$\begin{cases} x^2 + y^2 - 86.7729x + 3.32292y + 1885.14449 = 0 \\ x^2 + y^2 - 86.77258x + 3.32316y + 1885.13101 = 0 \end{cases}$$

$$\begin{cases} x_1 = 43.38662, y_1 = -1.66185 \\ x_2 = 43.38612, y_2 = -1.66119 \end{cases}$$

Intersection of $C1$ and $C2$ is area (x_1, x_2) (Fig. 8).

Zone3. Zone has 3 services, and is the intersection between 3 circles (3 Zone1 as in Fig. 9).

$C1$: is in Zone1 and has $P_{Sas\ Luz\ Voyages}$ service

$$x^2 + y^2 - 86.7729x + 3.32292y + 1885.14449 = 0 \quad (3)$$

$C2$: is in Zone1 and has $P_{Hotel\ Relais\ Saint}$ service

$$x^2 + y^2 - 86.77258\,x + 3.32316y + 1885.131001 = 0 \quad (4)$$

$C3$: is in Zone1 and has $P_{Boulangerie\ Magri\ Christophe\ Ogi\text{-}labea}$ service

$$x^2 + y^2 - 86.77270x + 3.32338y + 1885.13658 = 0 \quad (5)$$

Intersection of $C1$ and $C2$ is the results of two equations (3) and (4):

$$\begin{cases} x_{x1} = 43.38662, y_{x1} = -1.66185 \\ x_{x2} = 43.38612, y_{x2} = -1.66119 \end{cases}$$

Intersection of $C2$ and $C3$ is the results of two equations (4) and (5):

$$\begin{cases} x_{z1} = 43.41555, y_{z1} = -1.64589 \\ x_{z2} = 43.35713, y_{z2} = -1.67775 \end{cases}$$

Intersection of $C1$ and $C3$ is the results of two equations (3) and (5):

$$\begin{cases} x_{y1} = 43.62799, y_{y1} = -1.76661 \\ x_{y2} = 43.14745, y_{y2} = -1.55768 \end{cases}$$

Intersection of $C1$, $C2$ and $C3$ is area ($z1$, $x2$, $y2$) (Fig. 9).

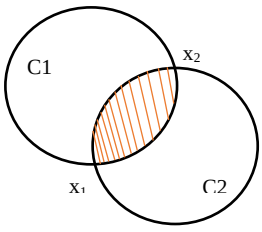


Fig. 8. Intersection between 2 circles

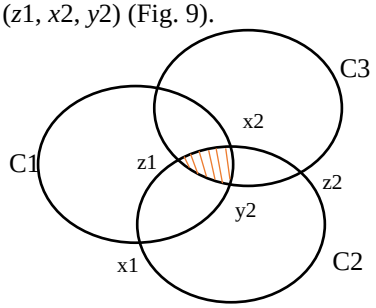


Fig. 9. Intersection between 3 circles

The abstract test cases are generated from t he test model in section 4. They can be expressed in another way as follows.

1. The user is outside the Zones and moves into Zone1.
2. The user is outside the Zones and moves into Zone1 and then moves into Zone2.
3. The user is outside the Zones and moves into Zone1 and then moves into Zone2 and after that moves into Zone3.
4. The user is in Zone1 and then moves into Zone2.
5. The user is in Zone1 and then moves into Zone2 and then moves into Zone3.

6. The user is in Zone2 and then moves into Zone3

Table 1. Partition boundaries of Zones.

Zones	Partition boundaries of Zones	
	Longitude	***Latitude***
Zone1 has Tourist spot service	From 43.38600 to 43.3869	From -1.66191 to -1.66101
Zone1 has Hotel service	From 43.38584 to 43.38674	From -1.66203 to -1.66113
Zone1 has Restaurant service	From 43.38590 to 43.38680	From -1.66214 to -1.66124
Zone2 has Tourist spot service and Hotel service	From 43.38612 to 43.38662	From -1.66185 to -1.66119
Zone3 has all three services	From 43.35713 to 43.62799	From -1.76661 to -1.66119

Based on Table 1, we can select a proper current coordinates position of the user who is inside the partition boundaries of Zones when he/she moves through Zones corresponding to the abstract test cases.

1. The user is outside the Zones and moves into Zone1.
   L_0(43.38611; -1.66303) -> L_1(43.38605; -1.66197)
   L_0(43.38611; -1.66303) -> L_1(43.38590; -1.66214)
2. The user is outside the Zones and moves into Zone1 and then moves into Zone2.
   L_0(43.38611; -1.66303) -> L_1(43.38605; -1.66197) -> L_2(43.38622, -1.66130)
   L_0(43.38611; -1.66303) -> L_1(43.3859; -1.66214) -> L_2(43.38652; -1.66170)
3. The user is outside the Zones and moves into Zone1 and then moves into Zone2 and after that moves into Zone3.
   L_0(43.38611; -1.66303) -> L_1(43.38605; -1.66197) -> L_2(43.38622; -1.66130) -> L_3(43.35805; -1.67890)
4. The user is in Zone1 and then moves into Zone2.
   L_1(43.38605; -1.66197) -> L_2(43.38622; -1.66130)
   L_1(43.38600; -1.66205) -> L_2(43.38632; -1.66165)
5. The user is in Zone1 and then moves into Zone2 and then moves into Zone3.
   L_1(43.38605; -1.66197) -> L_2(43.38622; -1.66130) -> L_3(43.35805; -1.67890)
   L_1(43.38600; -1.66205) -> L_2(43.38632; -1.66165) -> L_3(43.35895; -1.67990)
6. The user is in Zone2 and then moves into Zone3.
   L_2(43.38622; -1.66130) -> L_3(43.35805; -1.67890)
   L_2(43.38632; -1.66165) -> L_3(43.35895; -1.67990)

Instead of randomly selecting all location coordinates in the test execution process for context-aware mobile apps, we select the user's location coordinates inside partition boundaries. This saves time and cost in test execution process.

7 Conclusion and future work

Testing context-aware mobile apps is challenging due to the complexity of context variability. Current testing approaches cannot efficiently handle dynamic variability of context-aware mobile apps. To solve this problem, we present a model-based testing approach of mobile apps that uses a combination of a Bigraph Reaction System and a

Dynamic Feature Petri Net for automatic generation of test cases. This model addresses the mobile app testing challenges related to the context location of context-aware mobile apps. In this paper, we propose a test criterion combining pattern flow-based coverage criteria and boundary-based coverage criteria and illustrate its use on the TripAdvisor application. With this criterion, we select the user's location coordinates inside partition boundaries, which help narrowing the input data space and cover all test situations in test process. As future work, we plan to study the automation of the coverage assessment for any location dependent mobile application.

References

1. Nguyen, T. B., Le, T. T. B., Aktouf, O., and Parissis, I.: Mobile applications testing based on Bigraphs and Dynamic feature Petri nets. In: Nguyen, N. T., Dao, N. N., Pham, Q. D., Le, H. A. (eds.) Intelligence of Things: Technologies and Applications 2022, vol. 148, pp. 215-225, Springer, Heidelberg (2022).
2. Yu, L., Tsai, W.-T., Perrone, G.: Testing context-aware applications based on bigraphical modeling. IEEE Transactions on Reliability 65, pp. 1584–1611 (2016).
3. 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).
4. Ammann, P., Offutt, J., Huang, H.: Coverage criteria for logical expressions. In: 14th International Symposium on Software Reliability Engineering, pp. 99-107, (2003).
5. Muschevici, R., Clarke, D., Proenca, J.: Feature Petri nets. In: 14th International Conference, pp. 13-17. Jeju , Korea (2010).
6. Heng, L., Chan, W. K., Tse, T. H.: Testing context-aware middleware-centric programs: a data flow approach and an RFID-based experimentation. In: 14th International symposium on Foundations of software engineering, pp. 242–252, (2006).
7. Zhai, K., Jiang, B., Chan, W. K.: Prioritizing Test Cases for Regression Testing of Location-Based Services: Metrics, Techniques, and Case Study. IEEE Transactions on Services Computing, vol. 7, pp. 54-67, (2014).
8. Qun, R., Dunham, M. H.: Using semantic caching to manage location dependent data in mobile computing. In: 6th annual international conference on Mobile computing and networking, pp.210-222, (2000).
9. Zhang, T., Gao, J., Aktouf, O., Uehara, T.: Test Model and Coverage Analysis for Location-based Mobile Services, SEKE, pp. 80-86, (2015).
10. Kosmatov, N., Legeard, B., Peureux, F., Utting, M.: Boundary coverage criteria for test generation from formal models. In: 15th International Symposium on Software Reliability Engineering, pp. 139-150, (2004).
11. http://docs.oasisopen.org/wsbpel/2.0/CS01/wsbpel-v2.0-CS01.html, last accessed 2021/10/21.
12. https://gkchronicle.com/world-geography/Locating-points-on-earths-surface.php, last accessed 2023/1/21.
13. White, L. J., Cohen, E. I.: A Domain Strategy for Computer Program Testing. IEEE Transactions on Software Engineering, vol. 6, pp. 247-257, (1980).
14. Beizer, Boris,: Software Testing Techniques. John Wiley & Sons, USA, (1990).