LCIS
Laboratoire de Conception et d'Intégration des Systèmes

La Région Auvergne-Rhône-Alpes

UGA Université Grenoble Alpes

GRENOBLE INP UGA

Modèles pour le test d'applications mobiles sensibles au contexte

Travaux en cours au LCIS

Abdallah Adwan, Oum-El-Kheir Aktouf, Thi Than Binh Le,
Than Binh Nguyen, Ioannis Parissis

6.4 Billion Users

- Social networking
- Shopping
- Banking
- Transportation
- Food
- Entertainment

Testing mobile apps is tough

- Diversity of device configurations
- Mobility
- Context-awareness

Context-awareness in mobile apps

- Modern mobile devices integrate advanced sensors.
- Enabled development of context-aware features.
- Implicit input from the environment.
- Simulating context is hard

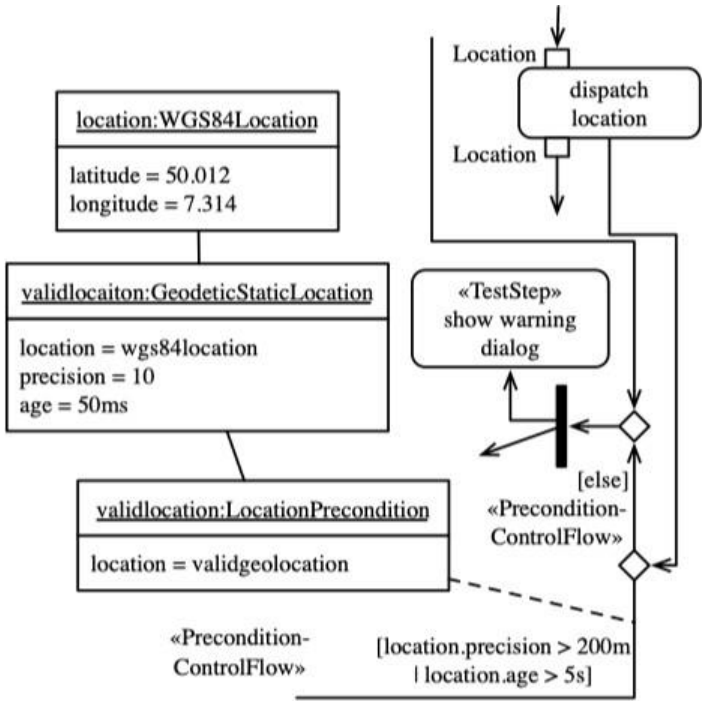
Examples of context related bugs

- Wikipedia Android App: app crashes when attempting to save a page without network connectivity.
- Traccar Android app: app fails to send location updates when GPS is turned off.
- A Twitter client app: app crashes when switching from WiFi to 3G while sending a tweet.

Related work

[Griebe and Gruhn, 2014]

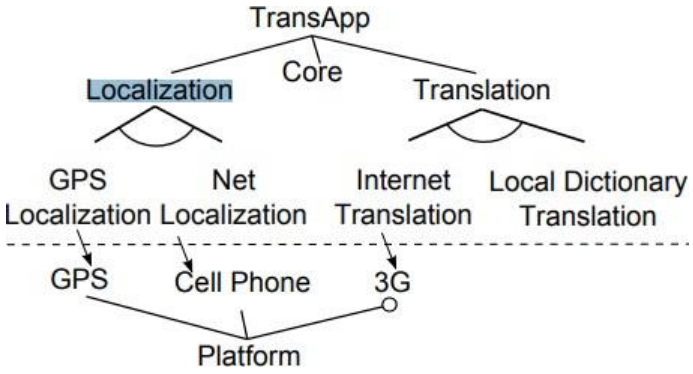
- Custom UML Activity Diagram.
- Static modeling of context data.

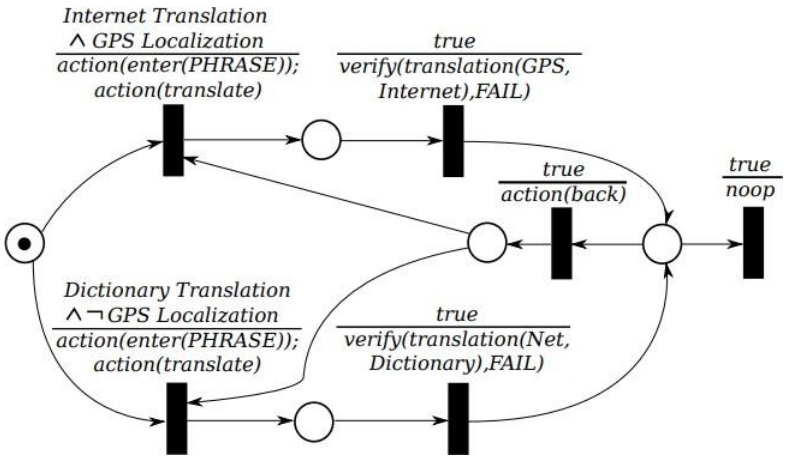


Related work

[Püschel et al., 2012]

- Context Model: Feature Model
- Behavioral Model: Feature Petri-net

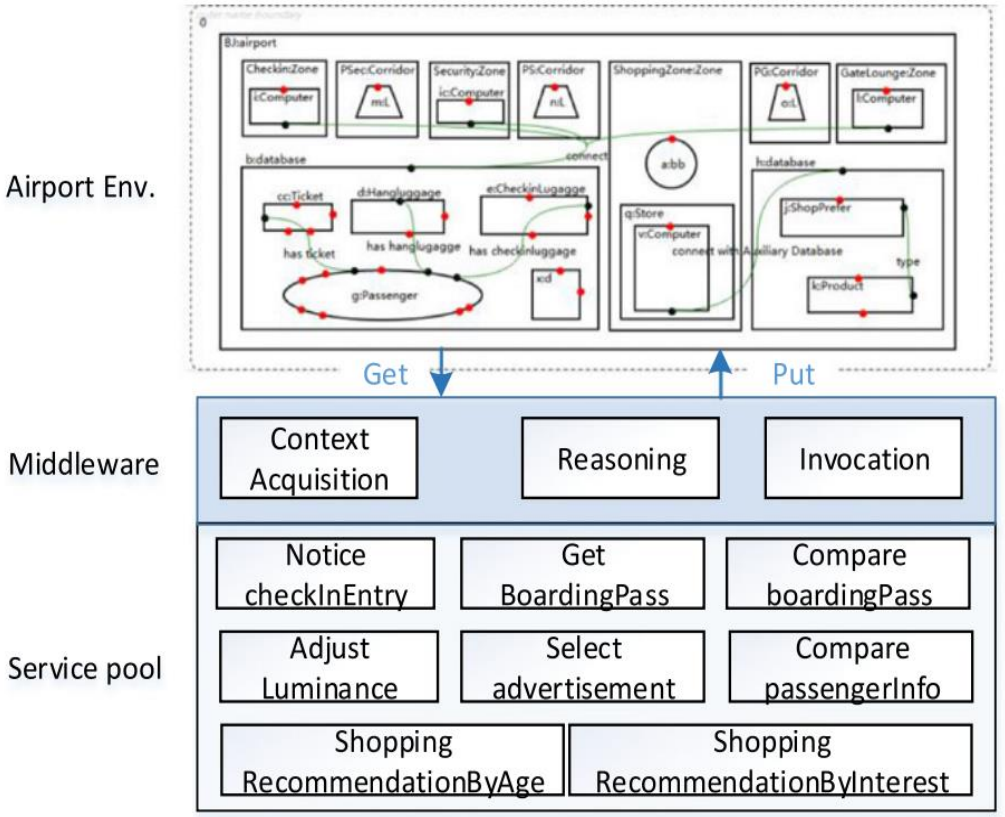




Related work

- [Lian Yu et al, 2016]

- Bigraph Reaction System to model the environment
- Extended Finite State Machine to model the middleware.
- Test cases can then be generated by synchronizing the BRSs and EFSM models.



Context-dependent Model-based Testing of Mobile Apps

Abdallah Adwan

Prof. Ioannis Parissis, co-supervisor

Assoc. Prof. Oum-El-Kheir Aktouf, co-supervisor

Master of Science in Informatics at Grenoble

Université Grenoble Alpes

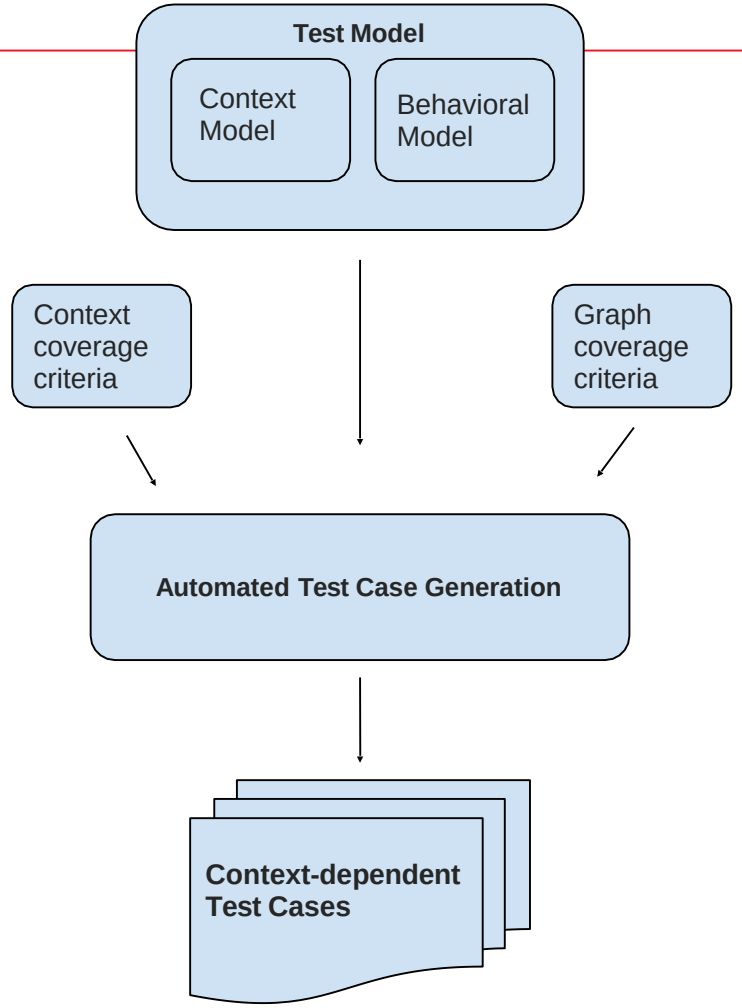
21/06/2021

Objectives of this work

- Model context-aware mobile apps for the purpose of testing
  - Test generation
  - Test coverage assessment

Solution outline

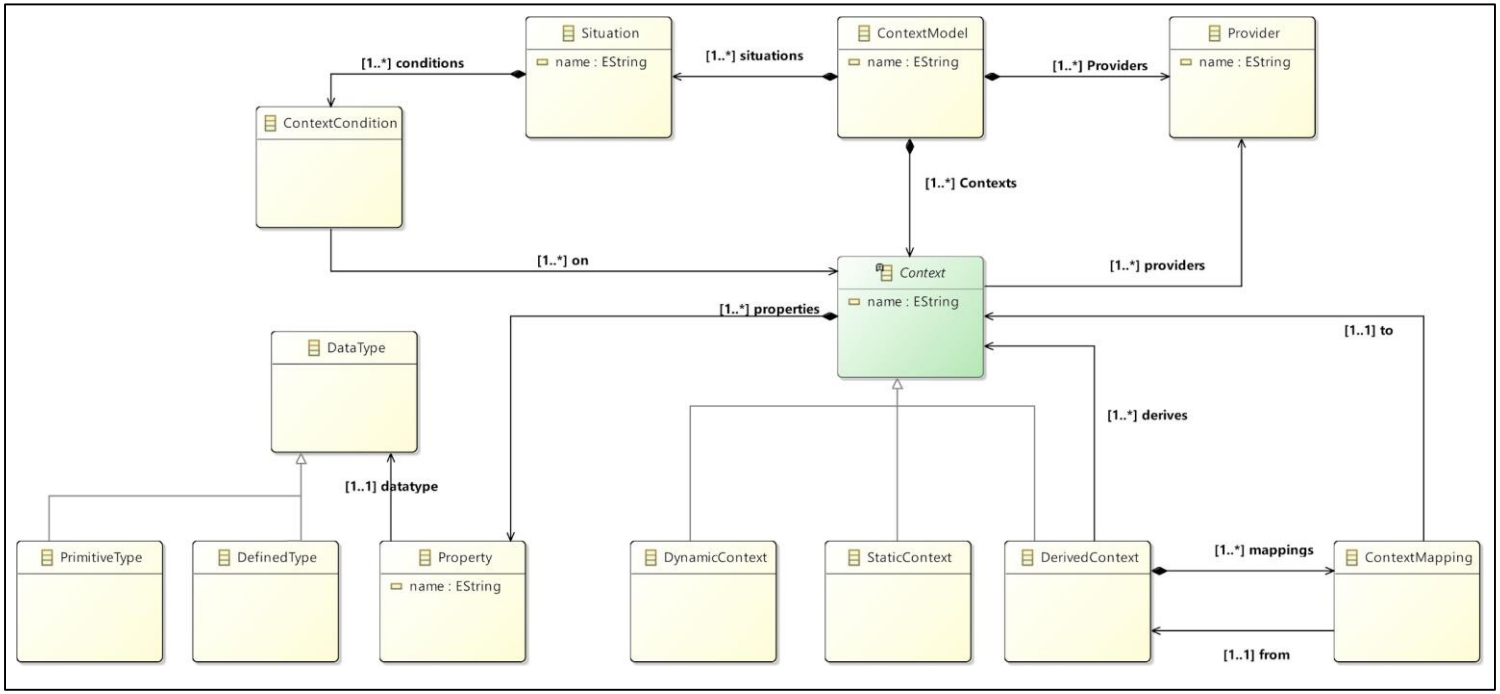
- Test model
  - Context and behavioral models
  - Model programs written in domain-specific languages
  - Finite state machine for behavioral modeling
- Test case generation
  - Input: Test model
  - Driven by test requirements expressed in terms of test coverage criteria
  - Specialized context-based coverage criteria
  - Output: context-dependent abstract test cases



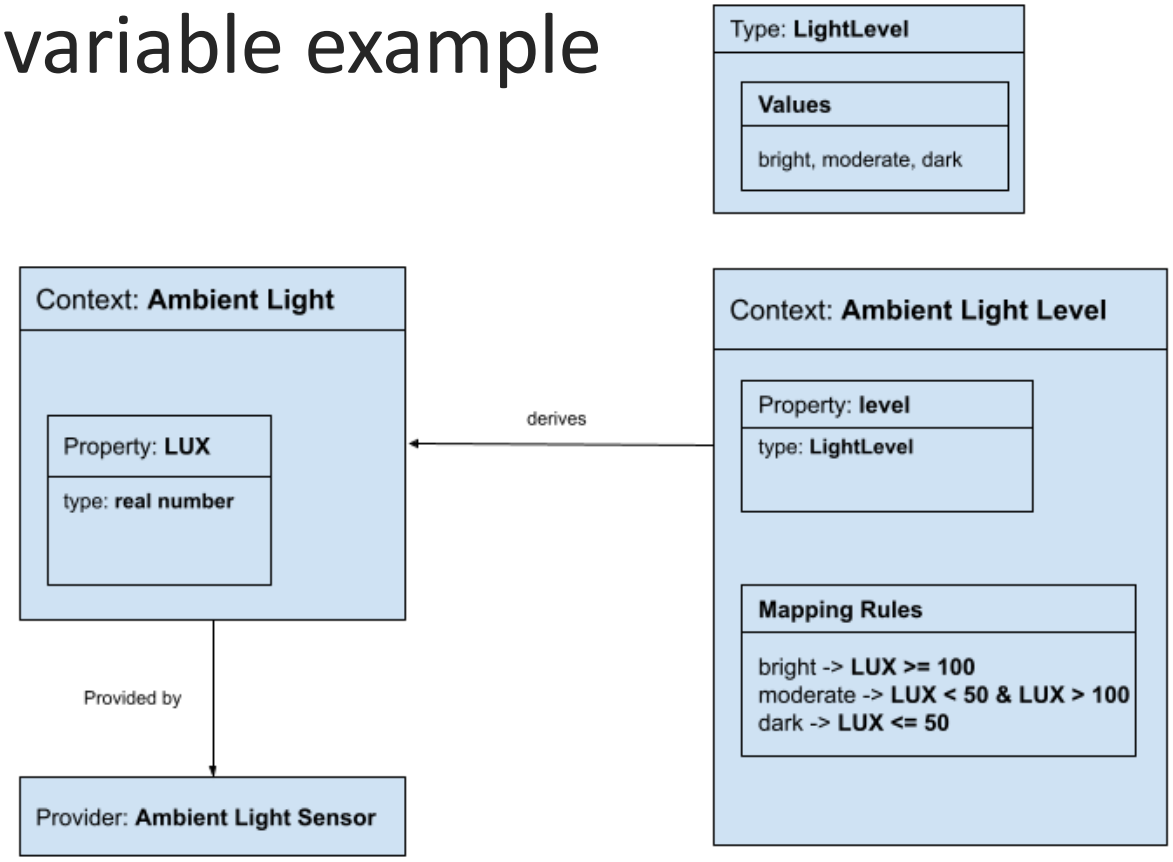
Context modeling

- Context variables used by the mobile app (e.g., user's location)
- Type of context variables (dynamic, static, derived)
- Source of context variable (e.g., physical sensor)
- Data properties of the context variable
- Derivation rules for derived context variables
- Interesting contextual situations that require the mobile app to adapt its behavior

Context metamodel



Context variable example



Context Definition Language (CDL)

```
context LOCATION {
  providers: [GPS_SENSOR, CELL_ADAPTER],
  properties: [longitude: double, latitude: double]
}

type Country {france, germany, spain}

context COUNTRY derives LOCATION {


 properties: [country: Country],
 mappings: {
   country.france -> LOCATION.longitude == 48.8534 and LOCATION.latitude == 2.3522,
   country.germany -> LOCATION.longitude == 52.5200 and LOCATION.latitude ==
   13.4050,  country.spain -> LOCATION.longitude == 40.4168 and LOCATION.latitude ==
}  3.7038
 }
```

```
type Connectivity {offline, wifi, slow3G, fast3G, 4g, high_latency}

context                  _CONNECTIVITY {
  INTERNET               FI_ADAPTER,
  providers : [WI        CELL_ADAPTER],
  properties : [co       nnectivity: Connectivity],
}
situation                NET_DISCONNECTED {
  INTER                  NNECTIVITY.connectivity == Connectivity.offline
  INTERNET_C
  O
}
```

Behavioral modeling

- Based on Hierarchical Finite State Machine (HFSM).
- Overcome the state space explosion problem inherit in FSM.
- Better modularity, scalability, and maintainability.

Modeling context-awareness

- Model app's context-independent (standard) behavior.
- Identify states where the context influence the app's behavior.
- Link context-aware states with their respective context variables.
- Extend the model with adaptive actions modeled as Adaptation FSMs that start with a context-aware state.

Context-driven Behavioral Modeling Language (CBML)

```
statemachine TRANSLATE_PHRASE_ACTIVITY_SM {

  state TRANSLATE_PHRASE_ACTIVITY  {
    transition on PHRASE_TEXTFIELED_SELECTED ->
    ENTER_PHRASE
    transition on TRANSLATE_BUTTON_CLICKED -> TRANSLATE
    external transition on TERMINATE_BUTTON_CLICKED -> EXIT
  }

 state ENTER_PHRASE {
   transition on PHRASE_ENTERED -> TRANSLATE_PHRASE_ACTIVITY
 }

 state TRANSLATE awareof LOCATION, INTERNET_CONNECTIVITY {
   external transition on TRANSLATION_SUCCEEDED -> VIEW_TRANSLATION_ACTIVITY
 }

}
```

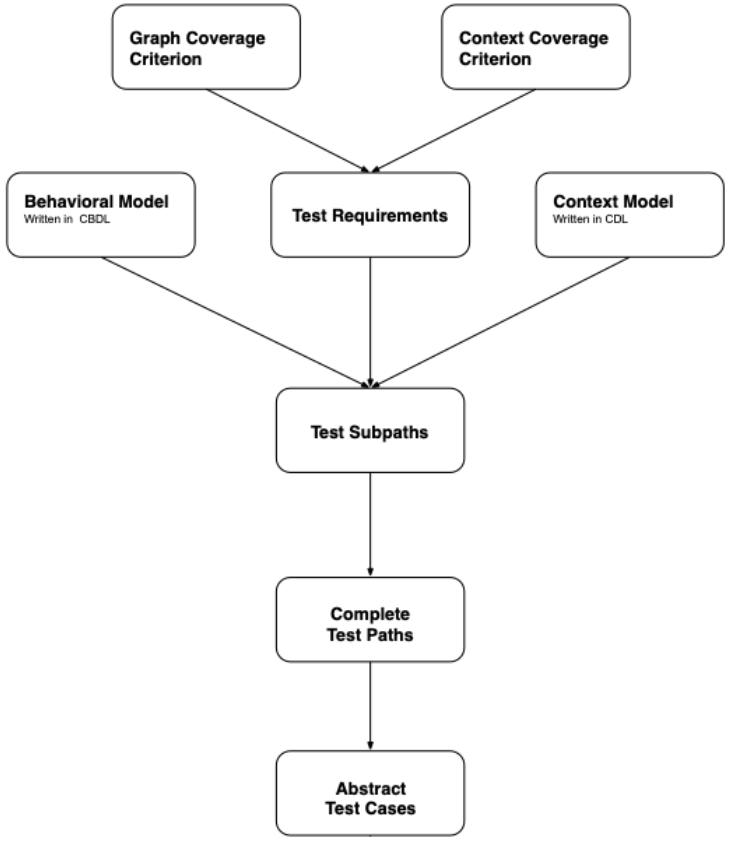
```
adaptation for INTERNET_DISCONNECTED at TRANSLATE {

  state TRANSLATE {
    transition on | ANSLATION_FAILED -> DISPLAY_WARNING
    TR
  }

  state | ARNING {
  DISPLAY_W | on on WARNIMG_DISMISSED -> TRANSLATE_PHRASE_ACTIVITY
    external
    transiti
  }
}
```

Test case generation

- **Input**: A behavioral model written in CBML, and its companion context model written in CDL.
- Driven by test coverage criteria.
- Test sub-paths are generated from hierarchy of FSMs in behavioral model.
- Sub-paths are aggregated to form complete test paths.
- Complete test paths are transformed to abstract test cases expressed as Cucumber scenarios.



Context-based test coverage criteria

- All-Situations Criterion
- All-Situation-Pairs Criterion

Test paths generation

- Test sub-paths generation
  - Generate test sub-paths for each FSM in the HFSM model.
  - Terminates when test coverage criteria are satisfied.
  - Paths consist of states and transitions.
  - Context constraints on context-aware states
- Test path aggregation
  - Starts with sub-paths generated from the top-level FSM of the HFSM
  - Recursively replace super states with their respective sub-paths

Test path aggregation

START ->
APP_STARTED ->
ENTER_PHRASE_ACTIVITY ->
PHRASE_TEXTFIELD_SELECTED ->
ENTER_PHRASE ->
PHRASE_ENTERED ->
ENTER_PHRASE_ACTIVITY ->
TRANSLATE_BUTTON_CLICKED ->
[INTERNET_CONNECTIVITY==offline] TRANSLATE ->
TRANSLATION_FAILED ->
DISPLAY_WARNING ->
WARNING_DISMISSED ->
ENTER_PHRASE_ACTIVITY ->
TERMINATE_BUTTON_CLICKED ->
EXIT

Abstract test cases

- Test paths are converted to abstract test cases expressed as Cucumber Scenarios.
- Cucumber is a widely used tool in test automation.
- Provides logical language to express excepted software behaviors.
- A good basis for transforming abstract test cases to executable ones.

Abstract test cases

START -> APP_STARTED ->
ENTER_PHRASE_ACTIVITY ->
PHRASE_TEXTFIELD_SELECTED ->
ENTER_PHRASE -> PHRASE_ENTERED ->
ENTER_PHRASE_ACTIVITY ->
TRANSLATE_BUTTON_CLICKED ->
[INTERNET_CONNECTIVITY==offline] TRANSLATE ->
TRANSLATION_FAILED ->
DISPLAY_WARNING -> WARNING_DISMISSED ->
ENTER_PHRASE_ACTIVITY ->
TERMINATE_BUTTON_CLICKED -> EXIT

Scenario: Attempt to translate when internet is disconnected.

Given APP_STARTED
And ENTER_PHRASE_ACTIVITY **When**
PHRASE_TEXTFIELD_SELECTED **Then**
ENTER_PHRASE
When PHRASE_ENTERED
Then ENTER_PHRASE_ACTIVITY
When TRANSLATE_BUTTON_CLICKED **And**
[INTERNET_CONNECTIVITY==offline] **Then**
TRANSLATE
When TRANSLATION_FAILED
Then DISPLAY_WARNING
When WARNING_DISMISSED
Then
ENTER_PHRASE_ACTIVITY
When TERMINATE_BUTTON_CLICKED
Then EXIT

Conclusion

- The proposed model-based testing approach has a great potential in the design and generation of effective test cases covering the context of a mobile app.
- The modeling approach based on FSM and using DSLs is simple and produces modular models that are maintainable and scalable.
- The context-based test coverage criteria we developed help testers assess context coverage in the generated test cases in a quantifiable manner.

LCIS
Laboratoire de Conception et d'Intégration des Systèmes

La Région Auvergne-Rhône-Alpes

UGA Université Grenoble Alpes

GRENOBLE INP UGA

A model-based testing approach based on bigraphical modeling

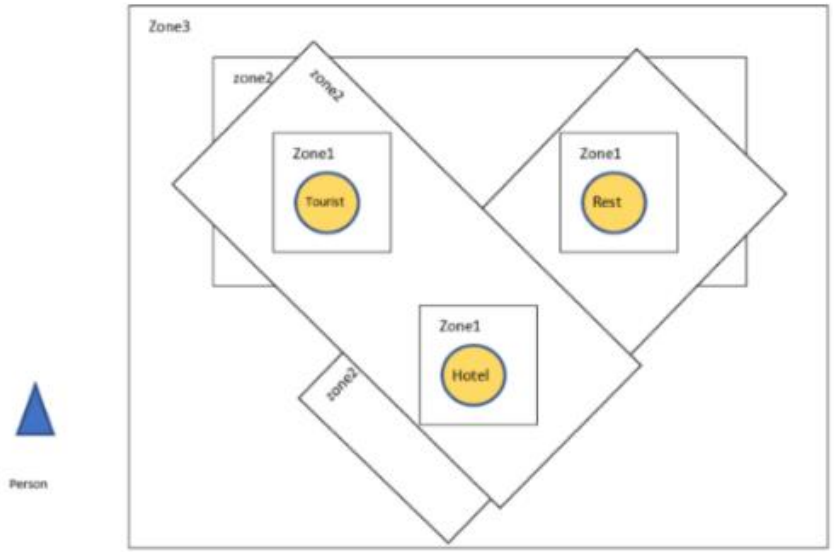
On going PhD work of Thi Than Binh Le
Université Grenoble Alpes and Danang University (Vietnam)

- Modelling a set of services using Extended Finite State Machines may be hard (large state space)
- So, we propose a model-based testing approach based on bigraphical modeling as follows:
- Using a (Bigraph Reaction System) to model the environment
- Using a Petri net to model the middleware.
- Synchronizing the BRS and Petri net to generate test cases.

The Tripadvisor example

- The Tripadvisor application provides a set of services (Tourist Spots, Hotel, Restaurants) which fit to the user's location.

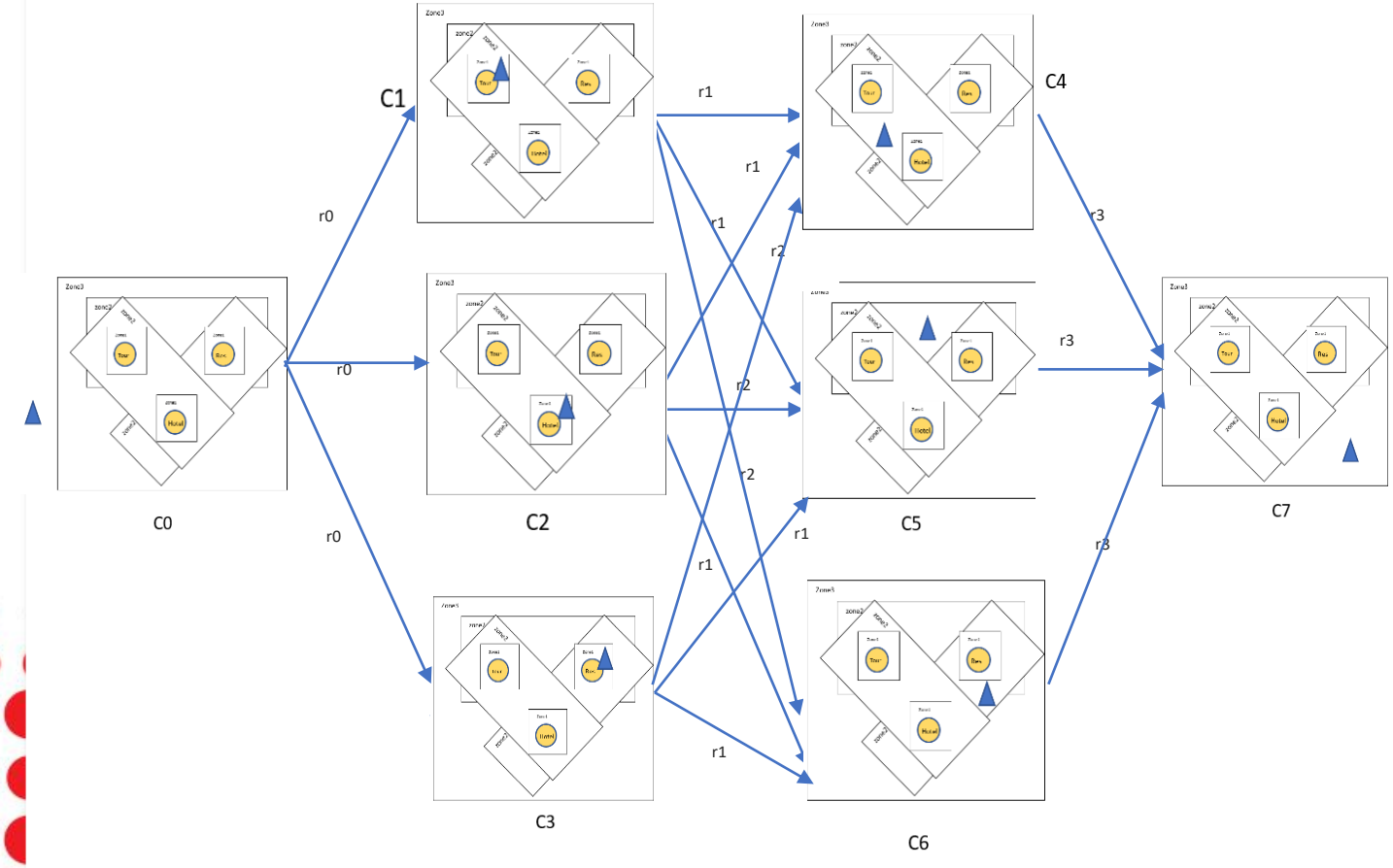
- Bigraph :



- Zone3 is a city of a country, zone2: a district belonging to the city, zone1: a ward belonging to the district.
- For example:
  - There is 1 service (e.g. Tourist Spots service or Hotel service or Restaurants service) in zone 1,
  - there are 2 services (e.g. Tourist Spots service and Hotel service or Hotel 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 Hotel service and Restaurants service) in zone 3.

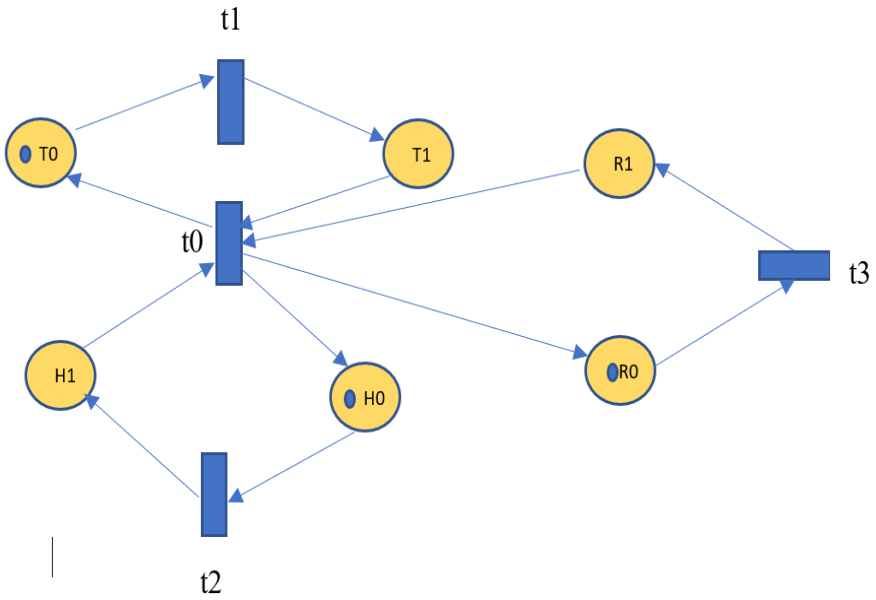
- *Reaction rules* to represent user moves

- C0, C1 ….. C7 bigraphs
- r0….r3 : reaction rules

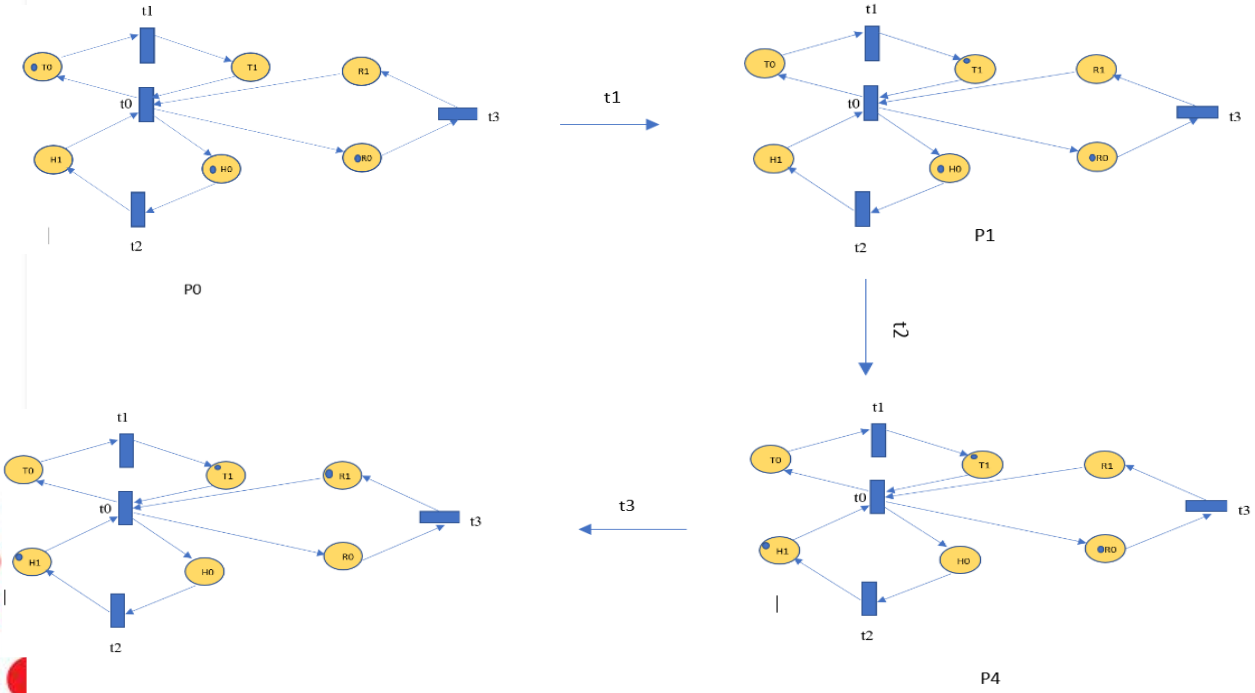


Dynamic feature Petri net model

- We describe this application with a set of services. Each service has 2 states and 2 transitions. For instance, Tourist Spots service (T) has 2 states: no service (T0) and service (T1); and transitions t0 and t1.



Example of Dynamic feature Petri net with test situation: The user is outside the zone and moves into zone 1 with Tourist Spots service, then moves into zone 2 where there are two services (Tourist Spots service and Hotel service), and finally moves into zone 3 with 3 services (Tourist Spots service, Hotel service and Restaurants service).



- P0 …. P7 : states of Petri net.
- t0….t3 : transitions

Selecting test cases

- Bigraph tree and the combination of Bigraph tree and Dynamic Feature Petri Net may result in a large number of paths.

- Selecting test cases requires defining adequacy criteria
  - Pattern-flow criteria
  - Data flow criteria
  - …

Conclusion & future work

- We study several modeling techniques for context-aware mobile apps
  - Hierarchical FSM combined with a DSL
  - Bigraphs combined with Petri nets
- Test selection methods are under investigation