تهیه و تنظیم: علی آخی
دکتر محمد تنهایی
دانشگاه ایلام

آزمون نرم افزار

GRAPH COVERAGE

FOR

USE CASE

ALI.AKHI-9811810007
ali.akhi.1998@gmail.com

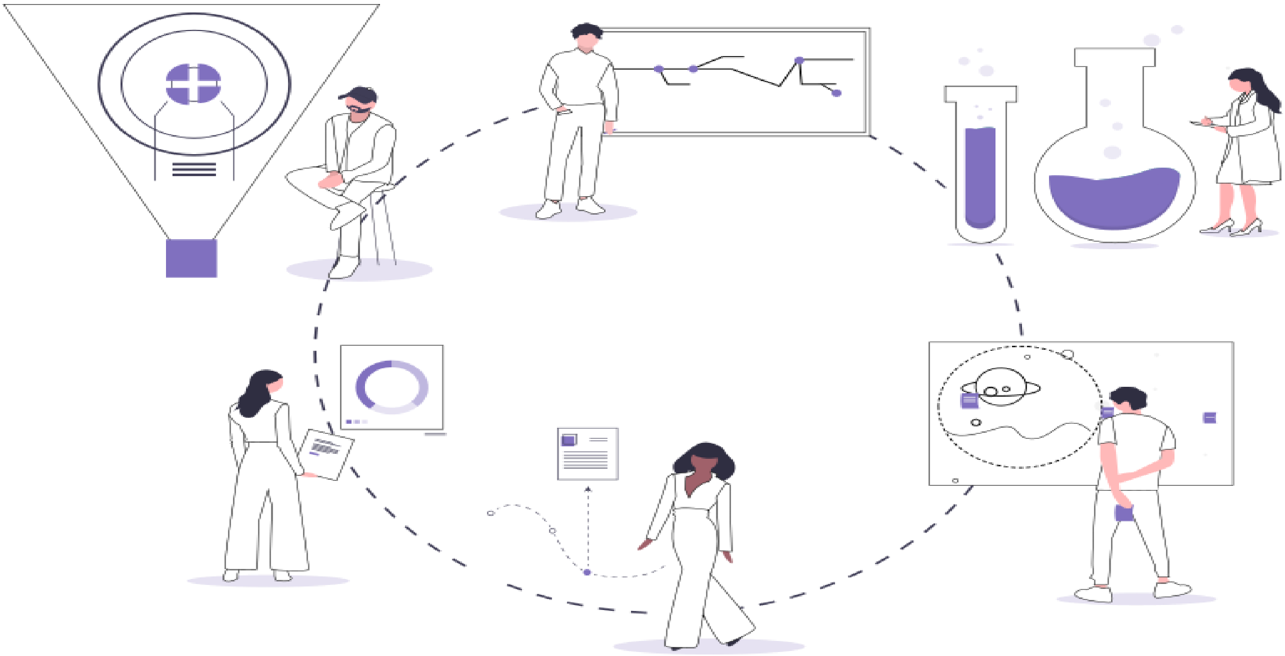
CONTENTS

UML Use Case

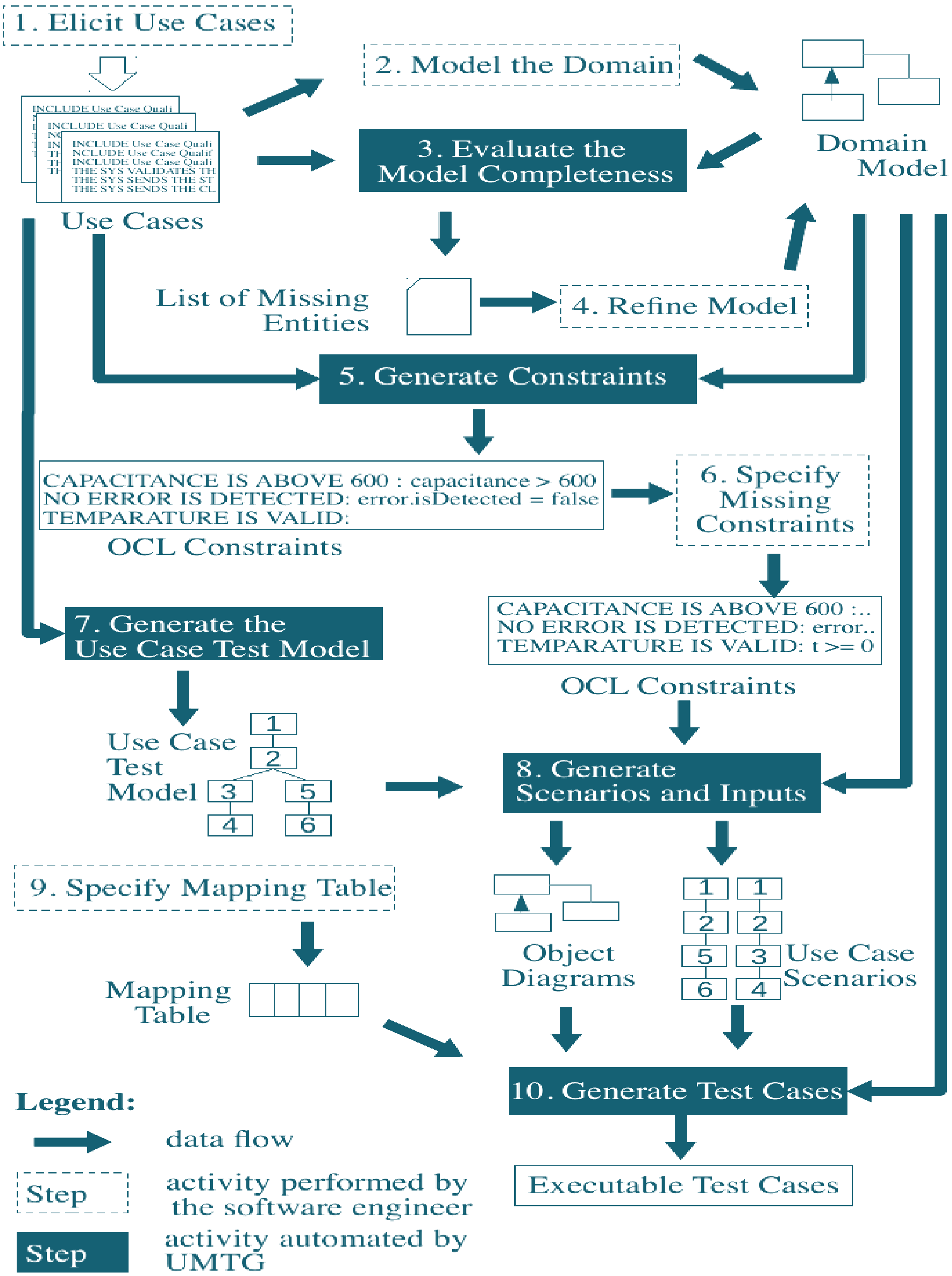
- UML use cases are often used to express software requirements(functional)
- They help express computer application workflow
- We won’t teach use cases, but show example

- Use Cases can help the tester to start testing activities early.
- Many book and papers can help the reader develop use cases But now we want to show you how to make activity graph and cover your graph.

The technique for using graph coverage criteria to develop tests from use cases in expressed through an example.



Significance of Use Case 1



1. Elicit Use Cases

Use Cases

2. Model the Domain

Domain Model

3. Evaluate the Model Completeness

List of Missing Entities

4. Refine Model

5. Generate Constraints

CAPACITANCE IS ABOVE 600 : capacitance > 600
NO ERROR IS DETECTED: error.isDetected = false
TEMPARATURE IS VALID:

OCL Constraints

6. Specify Missing Constraints

CAPACITANCE IS ABOVE 600 :..
NO ERROR IS DETECTED: error..
TEMPARATURE IS VALID: t >= 0

OCL Constraints

7. Generate the Use Case Test Model

Use Case Test Model

8. Generate Scenarios and Inputs

Object Diagrams

Use Case Scenarios

9. Specify Mapping Table

Mapping Table

10. Generate Test Cases

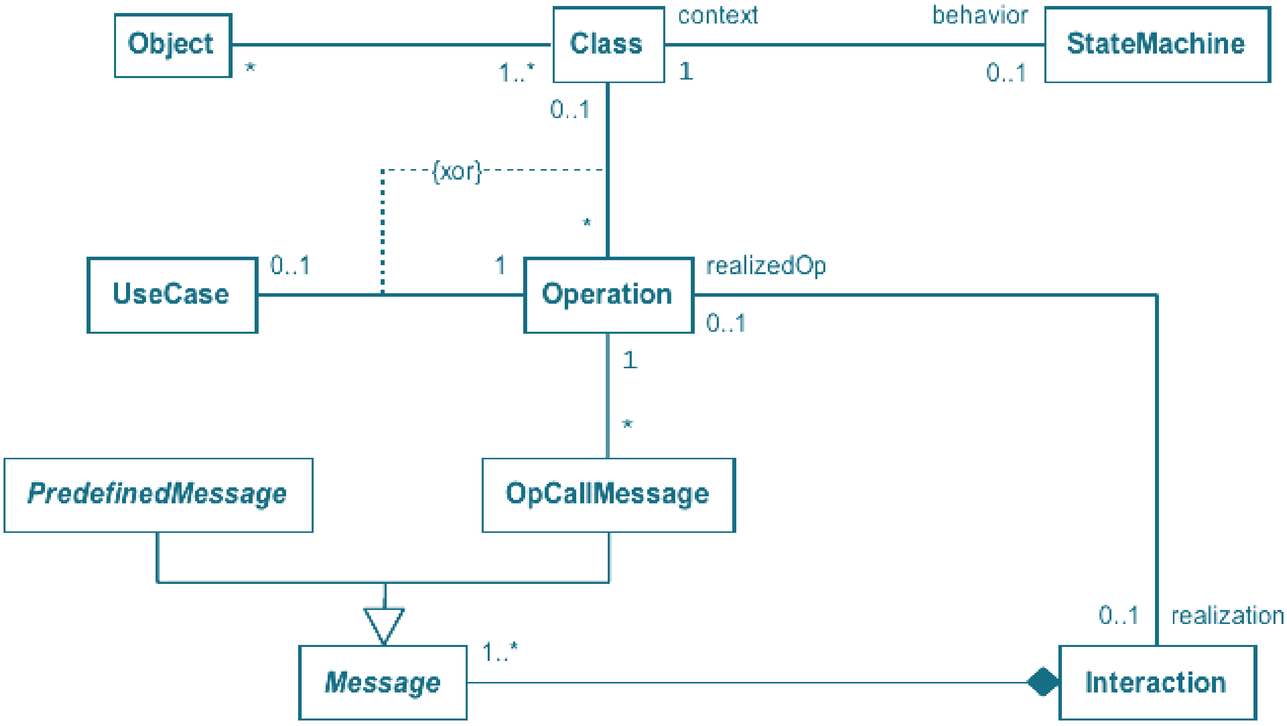
Executable Test Cases

Legend:

data flow

Step: activity performed by the software engineer

Step: activity automated by UMTG



Connection between central modeling concepts.

We cover the following UML features: use case, class, object, state, and interaction diagrams (collaboration and sequence diagrams) and last but not least full OCL.
We support class diagrams for defining the structure, and interaction diagrams for realizing operations declared in the class diagram. An interaction diagram contains a sequence of messages calling either an operation of a class that in turn is realized by an interaction diagram or calling a predefined functionality like creating an object or setting an attribute value.

Use cases are likewise realized by interaction diagrams. A use case resp. its realization states which operations are called by an actor and in which order this is done. State diagrams specify the order in which operations on an object may be executed.

Use Case Example

Fig. 2

- shows a use case diagram containing three use cases. The actor can get the time, or set the hours or minutes of the clock. The two use cases for setting the time have a parameter for the hour resp. minute the actor wants to set.

Fig. 3

declares the properties of our clock. It has one attribute holding the hours and one holding the minutes. There are two operations on the clock: pressA() for pressing the A button and pressB() for pressing the B button. In this simple example we have only this single class, however multiple classes with associations and inheritance are supported in our approach.

Use Case Specification

The use case specification is typically created in analysis and design phase in an iterative manner. At first, only a brief description of the steps needed to carry out the normal flow of the use case (i.e., what functionality is provided by the use case) is written.

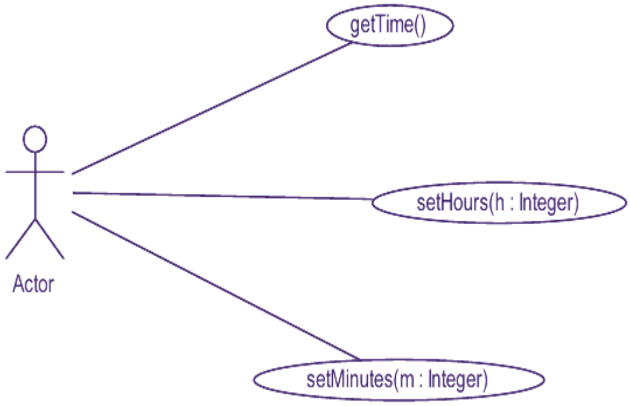


Figure 2. A use case diagram

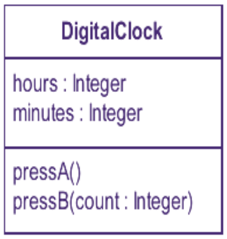


Figure 3. A class diagram

Elaboration of Use Case

- Use Case Name: Clock Setting
- Summary: The actor can get the time, or set the hours or minutes of the clock.
- Actor: MyActor
- Precondition: clock show the time
- Description:
  1. Show time of the clock
  2. If Time not be set,actor set the time of clock
  3. Fist check the hours and set hours of clock
  4. Second ckeck the minutes and set minutes of clock
  5. If Time is set
  6. Show the time

- Alternatives:
  - If clock don't show time show the 00:00
  - If hour is set ealier than minutes, show the current time
  - If time is not shown, show current time
- Postcondition: Time is set

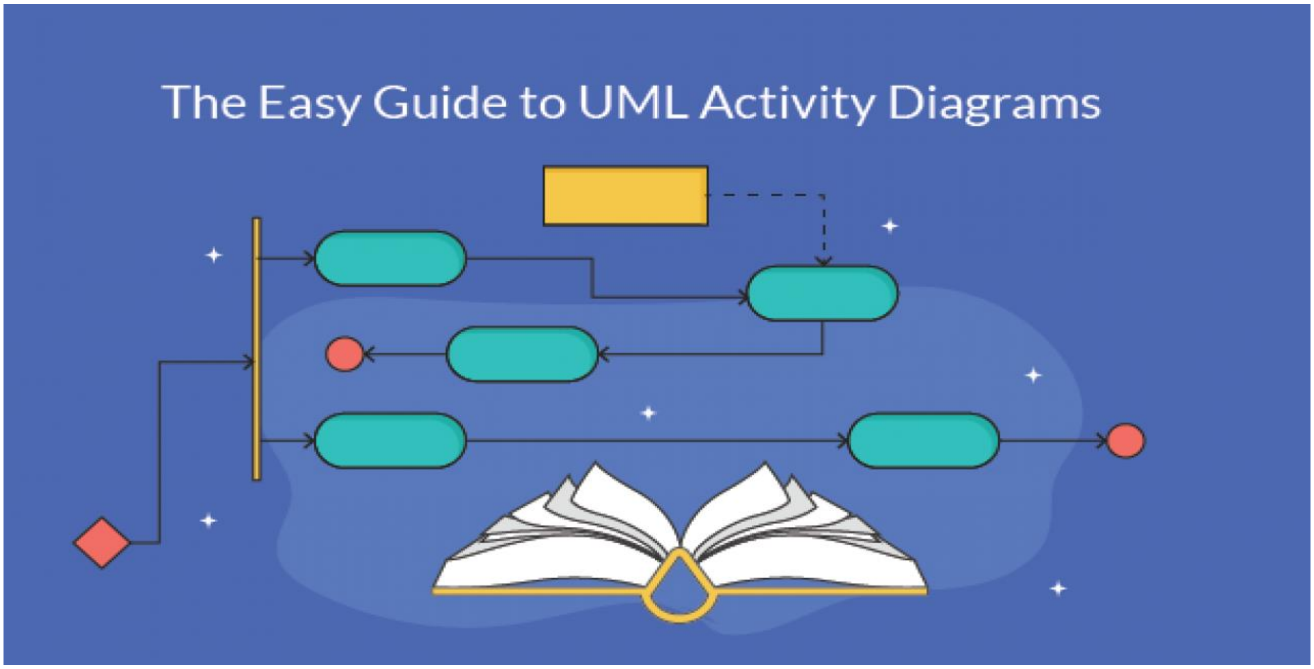
Wait A Minute ...

- What does this have to do with testing ?
- Specifically, what does this have to do with graphs ???
- Remember our admonition: Find a graph, than cover it !
- Beizer suggested "Transcation Flow Graph" in his book
- UML has something very similar: Activity Diagrams

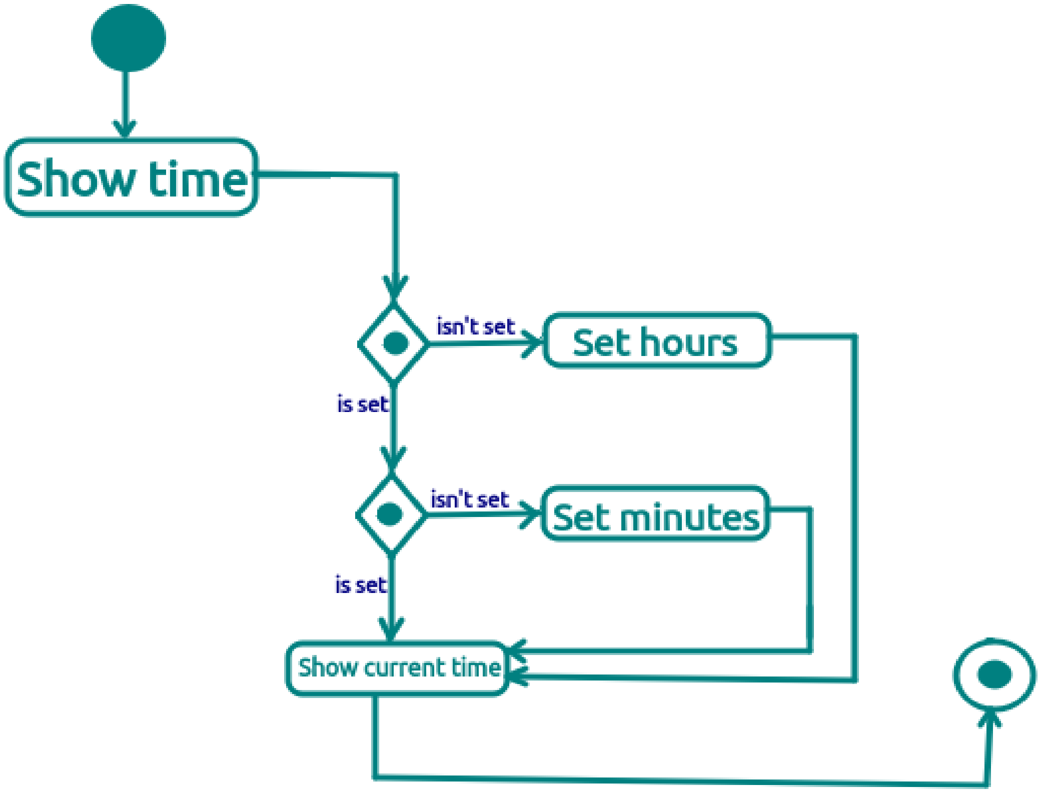
Use Cases to Activity Diagrams

- **Activity diagrams indicate flow among activities**
- **Activities should model user level steps**
- **Two kinds of nodes:**
  - **Action state**
  - **Sequential branches**

- **Use case descriptions become action state node in the activity diagram**
- **Alternatives are sequential branch nodes**
- **Flow among steps are adges**

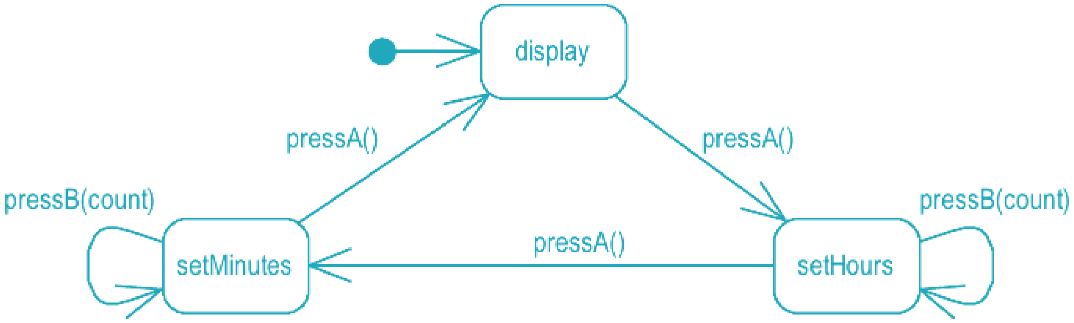


Simple Activity Graph



Activity diagrams usually have some helpful characteristics:

- **Few loops**
- **Simple predicates**
- **No abvious DU pairs**



Covering Graph

- Node Coverage
    - Inputs to the software are derived from labels on nodes and predicates
    - Used to from test case value
- Edge Coverage
- Data flow techniques do not apply
- Scenario Testing
    - Scenario: A complete path through a use case activity graph
    - Should make semantic sense to the users
    - Number of path often finite
    - If not, scenarios defined based on domain knowledge
    - Use "specified path coverage", where the set S of paths is the set of scenarios
    - Note that specified path coverage does not necessarily subsume edge coverage, but scenarios should be defined so that it does

Summary of Use Case Testing

- Use cases are defined at the requirements level
- Can be very high level
- UML activity diagrams encode use cases in graphs
    - Graphs usually have a fairly simple structure
- Requirements-based testing can use graph coverage
    - Straightforward to do by hand
    - Specified path coverage makes sense for these graphs