



Testing Object-Oriented Software cont ...

Prof. Durga Prasad Mohapatra
Professor
Dept. Of CSE, NIT Rourkela



UML-Based Software Testing

- UML is a design and modeling language for object-oriented software and is widely used as a *de facto* standard that provides techniques to model the software from different perspectives.
- It supports facilities both for abstract high level modeling and for more detailed low-level modeling.

UML-Based Software Testing cont...

- It consists of a variety of graphical diagram types that can be extended for specific application areas.
- It is associated with a formal textual language, *OC*L (Object Constraint Language).
- UML also provides support for model-based testing.

UML diagrams in software testing

- The following UML diagrams are helpful in the testing activities mentioned below:
 - **Use-case diagrams:** testing of system-level functional requirements, acceptance testing
 - **Class diagrams:** class (module / unit) testing, integration testing
 - **Sequence diagrams, collaboration diagrams:** integration testing, testing of control and interaction between objects, testing of communication protocols between (distributed) objects

UML diagrams in software testing cont...

- **Activity diagrams:** testing of work flow and synchronization within the system, white-box testing of control flow
- **State diagrams (state charts):** state-based testing
- **Package diagrams, component diagrams:** integration testing
- **Deployment diagrams:** system testing

System Testing based on Use-Cases

- **Use-case** In general, a use-case is a sequence of interactions by which the user accomplishes a task in a dialogue with the system.
- When we use use-case in testing, we use the following notations:
 - use-case = one particular way to use the system
 - use-case (in testing) = user requirement
 - set of all use-cases = complete functionality of the system
 - set of all use-cases (in testing) = interface between the users (*actors*) and the system

System Testing based on Use-Cases cont...

- **Scenario** It is an instance of a use-case, expressing a specific task, by a specific actor, at a specific time, and using specific data.

System Testing based on Use-Cases cont...

- ***Use-case model*** This is a set of use-case diagrams, each associated with a textual description of the user's task.
- For testing, the use-case model must be extended with:
 - the domain of each variable participating in the use-cases,
 - the input/output relationships among the variables,
 - the relative frequency of the use-cases,
 - the sequential (partial) order among the use-cases.



System Testing based on Use-Cases cont...

- Use case-based testing is a technique for generating test cases and recommending configurations for system-level testing.
- The testers build a test model based on the standard UML notions of use-cases, actors, and the relationships between these elements.
- The use-cases are enhanced with additional information, including inputs from actors, outputs to the actors, and how the use-case affects the state of the system.



System Testing based on Use-Cases cont...

- To perform this testing, the tester needs to identify four things:
 - the use-cases of interest,
 - the actors involved in using the system,
 - the input, output, and system effects for the use-cases,
 - the flows of interest between the use-cases.



System Testing based on Use-Cases cont...

- A use-case is a semantically meaningful function that provides some value from the user's point of view.
- For example, saving a file in a word processing system would be represented by the Save File use-case.

System Testing based on Use-Cases cont...

- Each use-case can have input parameters associated with it, and for each parameter, a set of logical partitions of the values that parameter can take may be identified.
- Finally, the use-cases can be connected using flows that describe a sequence of usecases that are performed to accomplish some goal.



Case study: Parking management system

It deals with the management of parking in an office. For parkin a car, both the Driver and Building Manager have to log into the system. When a car enters in the parking, then the space availability sensor checks for car availability. If the space is available, then only the car can be entered. The building manager also generates report for the total cars in parking.



Actors

1. Car driver
2. Building parking manager
3. Building display system
4. Space availability sensor



Use-cases

1. Login
2. Display space availability in parking
3. Car arrival in building
4. Car leaving the building
5. Generate reports
6. Maintain car details


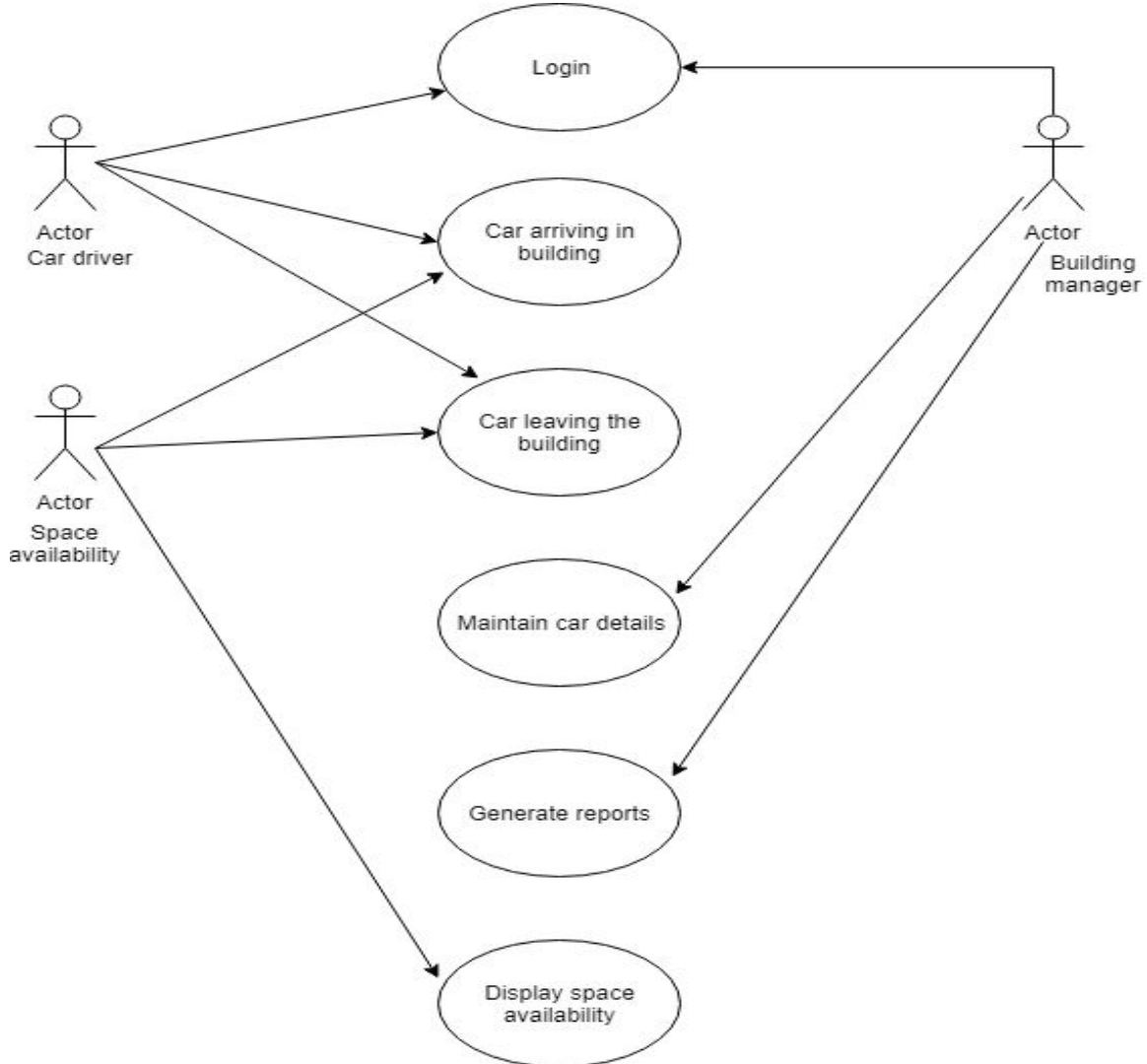
- 
- Every actor can interact with one or more use-cases for the specified purpose.
 - In this case study, we may identify the following actors corresponding to the use-cases identified above.

Table I: Actors for the use-cases

Sl.No.	Use-case	Actor
1	Login	Car driver
2	Display space availability in parking	Building display system
3	Car arrival in building	Car driver
4	Car leaving the building	Car driver
5	Generate reports	Building parking manager
6	Maintain car details	Building parking manager

Use-case diagram for parking management system





Generation of test cases from use-cases

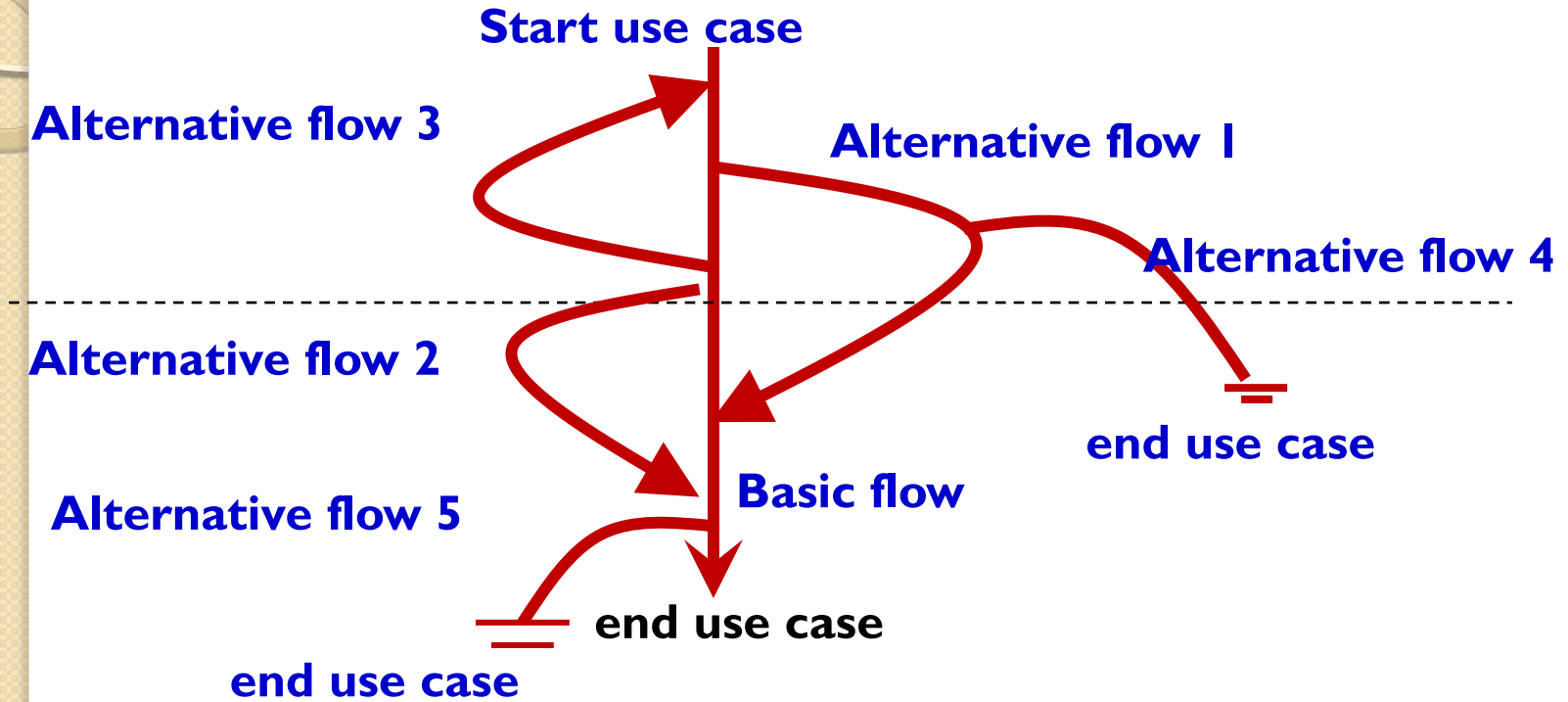
There is a systematic approach for the generation of test cases. The approach includes the following steps:

- Generation of scenario diagrams
- Creation of use-case scenario matrix
- Identification of variables in use cases
- Design of test case matrix
- Assigning actual values to variables



Generation of scenario diagrams

- In scenario diagrams the basic flow is one scenario or a complete path through a use-case.
- Various alternative paths show another scenario.
- Alternative paths are generated by different combinations of alternative flows.



Creation of use-case scenario matrix

A use-case scenario matrix gives all possible scenarios of the use-case scenario diagram.

Table 2: Use-case scenario matrix

Scenario 1	Log in successful	Basic flow	
Scenario 2	Login alternative flow: invalid user type	Basic flow	Alternative flow 1
Scenario 3	Login alternative flow: Invalid user name & password	Basic flow	Alternative flow 2
Scenario 4	Space not available	Basic flow	Alternative flow 3
Scenario 5	System down	Basic flow	Alternative flow 4
Scenario 6	User exists	Basic flow	Alternative flow 5

Identification of variables in use-case

The following use-case variables have been identified for use case Log in.

- User type
- User login id
- User password

Design of test case matrix

Test Case Id	Scenario Name	User Type	Login ID	Password	Expected Output	Remarks
TC1	Login	Valid input	Valid input	Valid input	User allowed to login	
TC2	Login alternate flow	Invalid input	Valid input	Valid input	Invalid user type	
TC3		Valid input	No input	No input	Please enter the login id and password	
TC4		Valid input	Valid input		Please enter the password	
TC5		Valid input	No input	Valid input	Please enter the login id	
TC6		No input	Valid input	Valid input	Please enter the user type	
TC7		Valid input	Invalid input	Valid input	Invalid login id	Login id does not exist in database
TC8		Valid input	Invalid input	Valid input	Login id is not in specific format	
TC9		Valid input	Valid input	Invalid input	Invalid password	Password does not exist in database
TC10		Valid input	Valid input	Invalid input	Password is not in specific format	
TC11		Valid input	Invalid input	Invalid input	Invalid login id and password	Login id and Password does not exist in database
TC12	Check availability of space	N/A	N/A	N/A	Parking allowed	
TC13		N/A	N/A	N/A	Parking full	
TC14	Exit	N/A	N/A	N/A		

Assigning actual values to variables

Test Case Id	Scenario Name	User Type	Login ID	Password	Expected Output	Remarks
TC1	Login	Car driver	Driver@ 1	abcd	User allowed to login	
TC2	Login alternate flow	Building manager	Driver@ 1	abcd	Invalid user type	
TC3		Car driver			Please enter the login id and password	
TC4		Car driver	Driver@ 1		Please enter the password	
TC5		Car driver		abcd	Please enter the login id	
TC6		No input	Driver@ 1	abcd	Please enter the user type	
TC7		Car driver	Driver2	abcd	Invalid login id	Login id does not exist in database
TC8		Car driver	Driver1	abcd	Login id is not in specific format	
TC9		Car driver	Driver@ 1	xyzs	Invalid password	Password does not exist in database
TC10		Car driver	Driver@ 1	XYZS	Password is not in specific format	
TC11		Car driver	driver	abcdef	Invalid login id and password	
TC12	Check availability of space	N/A	N/A	N/A	Parking allowed	
TC13		N/A	N/A	A	Parking full	
TC14	Exit	N/A	N/A	N/A	User logout from the system	



Thank you