SE202
Introduction to Software Engineering

Lecture 7.2
Software testing techniques

Last class

- What is software testing? and why is it important?

  + Testing is the process of exercising a program with the specific intent of finding errors prior to delivery to the end user.

  + More than 50% percent of the development time is spent in testing.

    + Software is buggy (average 1-5 bugs/KLOC)

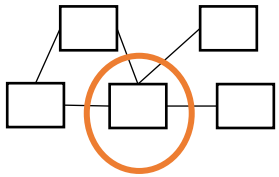
    + 100% correct mass-market software is impossible

  + For Verification & Validation purpose

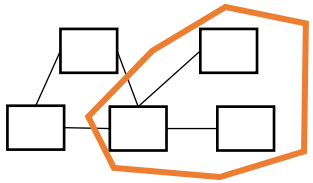
  + We must verify the software as much as possible.

Last class

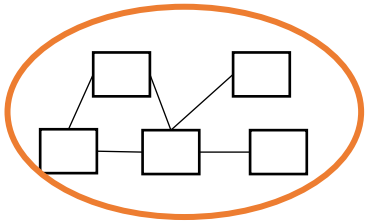
- Levels of testing



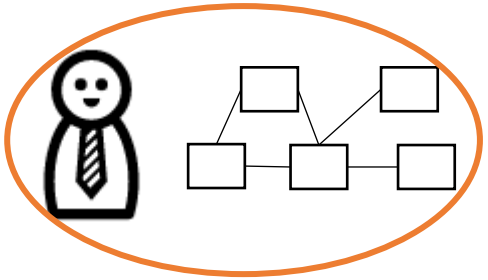
Unit testing



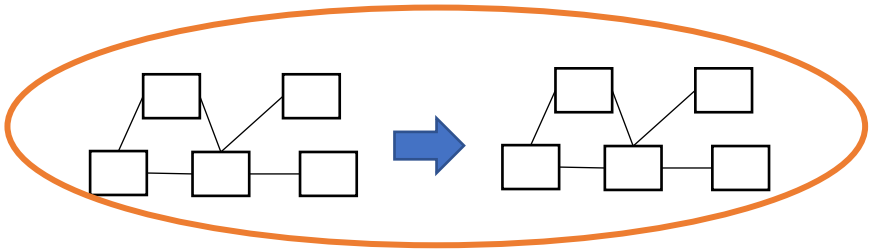
Integration testing



System testing



Acceptance testing

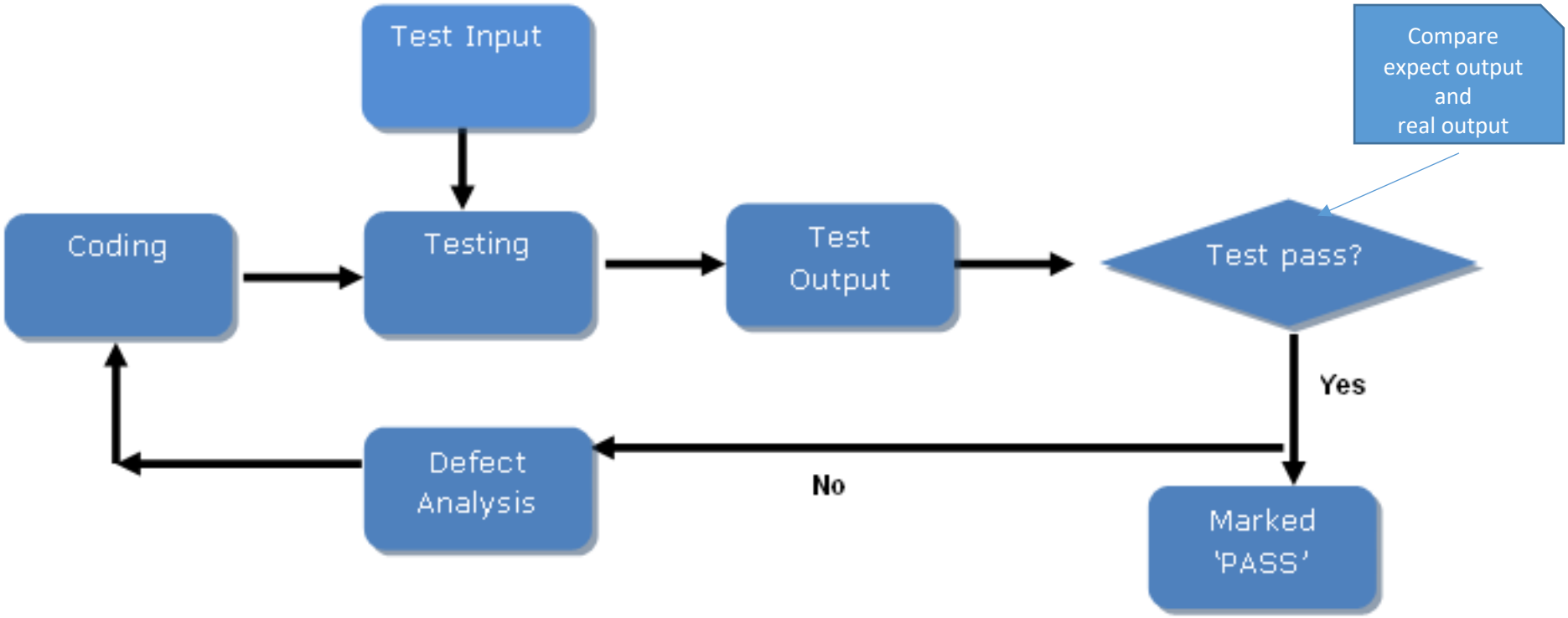


Regression testing

Today class

- Software testing techniques
  - Black-box testing

Testing process



Software testing techniques

Black-box testing

- Based on a description of the software (specification)
- Cover as much specified behavior as possible
- Cannot reveal errors due to implementation details

White-box testing

- Based on the code
- Cover as much coded behavior as possible
- Cannot reveal errors due to missing paths

Black-box testing

Advantages

- Focus on the domain
- No need for the code
  - Early test design
  - Tests are written before code
- Catches logic defects
- Applicable at all levels of testing

Black-box testing example

- *Specification:* inputs an integer and prints it.

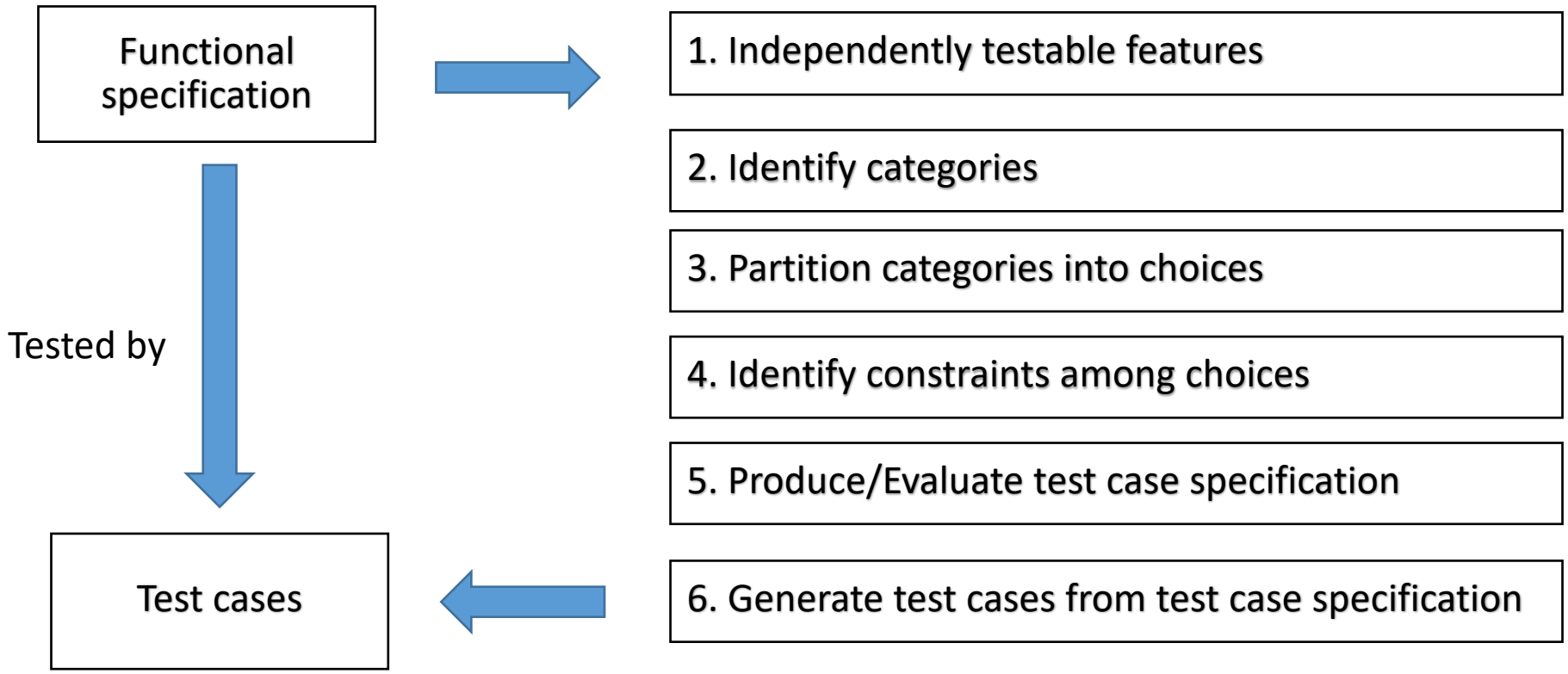
```
1.  void printNumBytes(param){
    .....

2.  }
```

What should be the expected outputs?

Input	Expected outcome	Actual outcome	Test result

```
1.  void printNumBytes(param){
2.      if (param < 1024) printf("%d",param);
3.      else printf("%dKB",param/124);
4.  }
```

A specific black-box testing approach (unit testing and integration testing)



1. Independently testable features

```
String split(String str, int size)
```

//pre-condition: the split method responses for separate the given string in two parts based on the given split size.
// post condition return the first part back to the caller.
//e.g. str = “abcd”; size=2; result=“ab”

Test all possible condition of input?

(abcd, 0)	result = “”
(abcd, 1)	result = “a”
(abcd, 2)	result = “ab”
(abcd, 3)	result = “abc”
(abcd, 4)	result = “abcd”
(abcd, 5)	result = ERROR

… many more!!

2. Identify categories

- Characteristics of each input element
- Example `String split(String str, int size)`

Input str
- *length*

- *content*

Input size
- *value*

3. Partition categories into choices

- Interesting cases (subdomains)
- Example `String split(String str, int size)`

Input str
- *length()*
  - 0
  - max_size str
  - ……..
- *content*
  - spaces
  - special characters
  - ……..

Input size
- *value*
  - > 0
  - = 0
  - < 0
  - max_int

4. Identify constraints among choices

- To eliminate meaningless combinations
- To reduce the number of test cases
- Three types: PROPERTY…….IF, ERROR, SINGLE
- Example

```
String split(String str, int size)
```

Input str
- length
  - 0 PROPERTY ZEROVALUE
- content
  - special characters IF !ZEROVALUE

Input size
- value
  - <= 0 ERROR
  - MAXINT SINGLE

5. Produce/Evaluate test case specification

- Can be automated
- Produce test specification
- Example

```
String split(String str, int size)
```

Test case specification #1

input str

length of str > 0 and <= max value of java String

content: special characters

input size

value: =>0 and <= length of str

5. Produce/Evaluate test case specification

- Can be automated
- Produce test specification
- Example

```
String split(String str, int size)
```

Test case no.	Test case specifiation
1	Test split method with input str with valid length (length of str > 0 and <= max value of java String) and special charaters content and input size with valid value (value: =>0 and <= length of str)
2	Test split method with input str with valid length (length of str > 0 and <= max value of java String) and special charaters content and input size with invalid value (value < 0)

6. Generate test cases from test case specification

- Simple instantiation of specifications
- Final result : set of concrete tests
- Example

split(string str, int size)

Test case no.	Test case specifiation	Input	Expected output
1	Test split method with input str with valid length (length of str > 0 and <= max value of java String) and special charaters content and input size with valid value (value: =>0 and <= length of str)	str = “ABCC!\n\tr” size = 10	“ABCC!\n \tr”
2	Test split method with input str with valid length (length of str > 0 and <= max value of java String) and special charaters content and input size with invalid value (value < 0)	str = “ABCC!\n\tr” size = -1	Error

Class exercise

Write two test cases (black box approach) for the following java method (from step 1-6)

char charAt(String str, int index)

// pre-condition given a string and location of the element of character

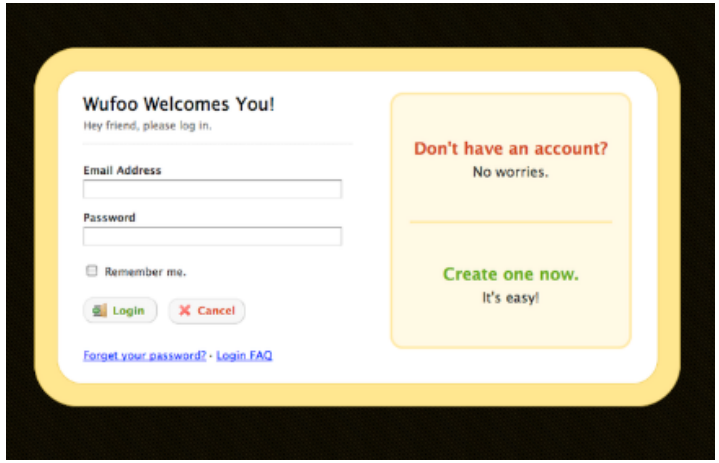
//post-condition : return the character regarding the specific location.

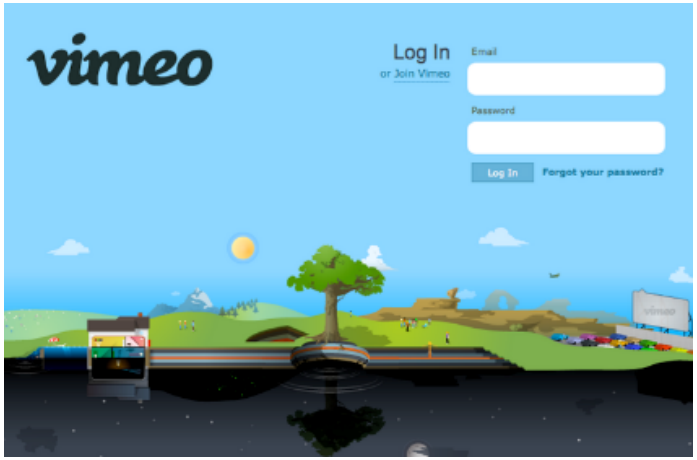
// e.g. str = “toey”, index = 1, result = ’o’

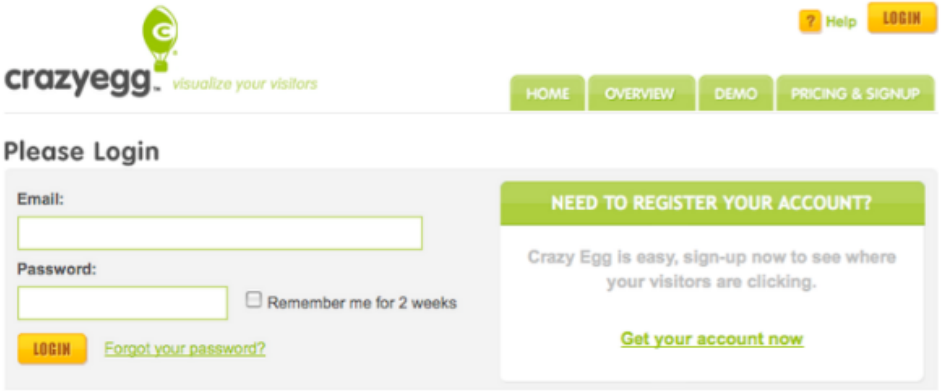
System test

- Type of testing to check the behavior of a complete and fully integrated software product based on the software requirements specification (SRS) document
- Use black box testing approach

1. Understand what requirement wants







2. Identify functional requirement

- **REQ01: User can login to the application**
  - **Input**: username, password
  - **Output**: user should be able to login and go to 'User Profile' page
  - **Computation**:
    1. When user access 'Home', website should display brand logo, input fields of username and password, login button, link to 'Forgot Password' page, link to 'Register' page
    2. When user input *correct username* and password, website should load 'User Profile' page
    3. When user input *incorrect username*, website remains at current page with message "Please enter correct username"
    4. When user input *incorrect password* for the first 3 times, website remains at current page with message "Please enter correct password. *Account will be locked after 3 times of failed login*"
  - **Data**: status of user login, counter of fail login
  - **Timing**: when user access 'Home' page of the application

3. Plan the test scenarios

Scenarios should be created to cover every possible case of the requirement especially to conditions of computation/input

- Case01: user can access 'Home' page with complete web elements
- Case02: user can use correct username and password to login
- Case03: When user use wrong upper/lower case of character to login, ….
- Case04: When user use wrong username to login, ….
- Case05: When user use right username but wrong password to login for the 1st time, ….
- Case06: When user use right username but wrong password to login for the 2nd time, ….
- Case07: When user use right username but wrong password to login for the 3rd time, ….
- Case08: When user use right username but wrong password to login for the 2 times but then use the right username and password to login the next time, ….
- Case09: [Continue from case08] user logout, user use right username but wrong password to login for the 2 times but then use the right username and password to login the next time, ….

4. Identify input and expected output to execute each scenario

- Case01: user can access ‘Home’ page with complete web elements
  - Input: ‘Home’ page url to web browser
  - Expected output: user can access web page with brand logo, input fields of username and password, login button, link to ‘Forgot Password’ page, link to ‘Register’ page
- Case02: user can use correct username and password to login
  - Input:
    - Username: mytestaccount
    - Password: 123456789
  - Expected output: user can access ‘User Profile’ page
- Case03: When user use wrong upper/lower case of character to login, ….
  - Input:
    - Username: Mytestaccount
    - Password: 123456789
  - Expected output: user cannot login and there’s a message above username box “Please enter correct username”

5. Execute the test and record result

- Case01: user can access ‘Home’ page with complete web elements
  - Input: ‘Home’ page url to web browser
  - Expected output: user can access web page with brand logo, input fields of username and password, login button, link to ‘Forgot Password’ page, link to ‘Register’ page
  - **Actual output:** web page contains every elements as expected
  - **Result: Pass**
- Case02: user can use correct username and password to login
  - Input:
    - Username: mytestaccount
    - Password: 123456789
  - Expected output: user can access ‘User Profile’ page
  - **Actual output: user can login**
  - **Result: Pass**
- Case03: When user use wrong upper/lower case of character to login, ….
  - Input:
    - Username: Mytestaccount
    - Password: 123456789
  - Expected output: user cannot login and there’s a message above username box “Please enter correct username”
  - **Actual output: user cannot login, input box was reset after click login button but there is no error message**
  - **Result: Fail**