Study Objectives

• Software Development contd.
• Quality Assurance
• Testing
Evolution of Programming Language (PL)

- PL evolution continues
- Today’s Internet Programming use dynamically typed languages for speed of getting the programs written
- Static typed languages are more structured, less error prone, but require compilation hence slow
- New languages are more flexible, e.g. C# 4.0 added dynamically typed feature, Dart (JavaScript replacement by Google, Swift (Objective C replacement by Apple)

The continuum of development languages.

Adapted from “Gradual Evolution, Dynamically Typed Languages Adopt Features of Static Typing to Cope up with growth,” by Neil Savage, Communication of the ACM, Oct 2014
Testing

• What’s true for Testing?

A. Testing ensures that code meets the Requirements expectation.
B. Testing ensures that the incremental functionality is provided by the software (Progression).
C. Testing ensures that the existing functionality continues to work (regression).
D. Testing ensures that the system meets performance expectation.
E. Testing ensures that when different components of the system are put together, the system as a whole continues to function.
F. Testing ensures when a software is released to production, it behaves the same way it had behaved in the lower environments.
G. Testing ensures that Project deliverables are on time.
H. Testing ensures that the budget planned for the project is met.
Types of Testing at Different Phases of SDLC (Week 4)

V-Model is a variant of water-fall model

Adapted from “CISA Review Manual 2014” ISACA, Exhibit 3.8”
Software Testing Strategy

Constitutes

– Goal of the testing
– types of testing to be conducted (for example, progression, regression, performance, etc.)
– Roles and responsibilities of the test teams and teams supporting testing
– Entrance and Exit criteria (for example, no Sev 1 or 2 open defects)
– Tools used to track Test Plan, Execution, and Defects (for example, Quality Center)
– Communication
– Artifacts (Test results, testing metrics etc.)
Elements of Testing

1. Test Plan
   • Functional Description of the Test Cases
   • Traceability helps keep track of the requirements are being tested
   • Typically one to many relationship between Requirements and Test Cases
   • “Bottom Up” (more common on large projects) or “Top Down” approach

2. Test Environment and Data Preparation
   • Ensure the test data is planned available to meet testing needs
   • Data often need to be stubbed for privacy reasons
   • Test environment is isolated from production

3. Test Execution
   • Running Test Cases and Reporting the status and test defects

4. Defects Management
   • Tracking of the defects
   • Fix of the defects
<table>
<thead>
<tr>
<th>No.</th>
<th>Requirement</th>
<th>Type</th>
<th>Priority</th>
<th>Must Have?</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Provide ability to collect name and address from the end users</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>Provide ability to enter person’s first name and last name in separate input fields</td>
<td>Functional</td>
<td>High</td>
<td>Must-have</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>Provide ability to enter address, which would have Street number, Street name, city, and zip code in separate input fields</td>
<td>Functional</td>
<td>High</td>
<td>Must-have</td>
<td></td>
</tr>
<tr>
<td>1.2.1</td>
<td>An address must have street name and city</td>
<td>Functional</td>
<td>Medium</td>
<td>Must-have</td>
<td></td>
</tr>
<tr>
<td>1.2.2</td>
<td>Zip code must be 5 character numbers</td>
<td>Functional</td>
<td>Low</td>
<td>Nice-to-have</td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>If user clicks on “submit” button without entering name and address fields, prompt user with “you must enter your name and address to continue”</td>
<td>Functional</td>
<td>Medium</td>
<td>Must-have</td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>If a user is a returning user, pre-populate the existing user name and address</td>
<td>Functional</td>
<td>Medium</td>
<td>Nice-to-have</td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td>Provide ability to store the name and address fields as part of the customer records</td>
<td>Functional</td>
<td>High</td>
<td>Must-have</td>
<td></td>
</tr>
<tr>
<td>1.6</td>
<td>Provide ability to submit the name and address page in less than 1 seconds</td>
<td>Performance</td>
<td>Medium</td>
<td>Nice-to-have</td>
<td></td>
</tr>
<tr>
<td>1.7</td>
<td>Provide ability to report on for manual re-entry by Support team, if the customer submits the name and address field and data is not successfully written in customer records</td>
<td>Operational</td>
<td>High</td>
<td>Must-have</td>
<td>Address part of reporting</td>
</tr>
<tr>
<td>1.8</td>
<td>The systems should be able to handle 2 million concurrent transaction with no degradation in performance</td>
<td>Performance</td>
<td>Medium</td>
<td>Must-have</td>
<td>At least 1 Million</td>
</tr>
</tbody>
</table>
### Develop Test Plan for The Web Ordering Example

<table>
<thead>
<tr>
<th>TC</th>
<th>Traceability</th>
<th>TC Name</th>
<th>TC Description</th>
<th>Step Name</th>
<th>Step Description</th>
<th>Expected Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.1 Func_First_Last_Name</td>
<td>User can enter first and last name successfully in the Personal Information screen</td>
<td>Step 1</td>
<td>Precondition - User must have clicked on &quot;order&quot; button on the home page and user is not a pre-registered user</td>
<td>Personal Information screen pops up</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Step 2</td>
<td>Check &quot;editability&quot; and initial data in first and last names fields</td>
<td>Both First and Last Name fields are editable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Step 3</td>
<td>Enter first name - &quot;John&quot; Enter last name - &quot;Doe&quot;</td>
<td>First name displays &quot;John&quot; Last name displays &quot;Doe&quot;</td>
</tr>
<tr>
<td>2</td>
<td>Func_First_Last_Name_No_Solchar_or_numbers</td>
<td>User should not be able to enter special characters and numerals first and last name successfully in the Personal Information screen (negative test)</td>
<td>Step 1</td>
<td>Precondition - User must have clicked on &quot;order&quot; button on the home page and user is not a pre-registered user</td>
<td>Personal Information screen pops up</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Step 2</td>
<td>Check &quot;editability&quot; and initial data in first and last names fields</td>
<td>Both First and Last Name fields are editable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Step 3</td>
<td>Enter first name - &quot;John123&quot; Enter last name - &quot;$5@Doe&amp;&quot;</td>
<td>First name displays &quot;John&quot; Last name displays &quot;Doe&quot;</td>
</tr>
<tr>
<td>3</td>
<td>1.2 Func_custaddress_after_name</td>
<td>User should be able to enter address, which would have Street number, Street name, city, and zip code in separate input fields. This is after entering the name.</td>
<td>Step 1</td>
<td>Precondition - User must have clicked on &quot;order&quot; button on the home page and user is not a pre-registered user</td>
<td>Personal Information screen pops up</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Step 2</td>
<td>Check &quot;editability&quot; and initial data in Street number, Street name, city, and zip code in separate input fields</td>
<td>Street number, Street name, city, and zip code in separate input fields are editable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Step 3</td>
<td>Enter first name - &quot;John&quot; Enter last name - &quot;Doe&quot;</td>
<td>First name displays &quot;John&quot; Last name displays &quot;Doe&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Step 4</td>
<td>Enter Street Number - 100 Enter Street Name - &quot;Main St&quot; Enter City - &quot;Anytown&quot; Enter Zip Code - 12345</td>
<td>Street Number displays - &quot;100&quot; Street Name displays - &quot;Main St&quot; City displays - &quot;Anytown&quot; Zip Code displays - 12345</td>
</tr>
</tbody>
</table>

Testing Classification

1. Unit Testing – Testing of Individual module or unit

2. Functional Testing – Testing functionality
   • Progression Testing: Test incremental functionality
   • Regression Testing: Test Existing functionality to ensure they continue to work, and a new problem has not been introduced
   • Sometimes can be part of Integration, Unit, or Systems Testing

3. Integration Testing – Interface between the systems is tested
Testing Classification

4. Systems Testing – Complete System is tested. It also includes:
   • Recovery Testing: system’s ability to recover from a software or hardware failure
   • Security Testing: Access control, security risks within the program
   • Performance Testing:
     ▪ Load Testing: Test the system at peak production simulated load
     ▪ Volume Testing: Test systems performance at incremental volume
     ▪ Stress Testing: Testing “breaking point”

5. Acceptance Testing
   • UAT (User Acceptance Testing): User Acceptance
   • QAT (Quality Assurance Testing): Final Regression
Other Testing Types

White Box and Black Box Testing

- **White Box Testing**
  - Testing program logic
  - Typically by developers during unit-testing phase
- **Black Box Testing**
  - Functional testing without going at the low level (program level)
  - Part of progression or regression testing
  - Can be during Integration or Systems Test phase

Alpha and Beta Testing

- **Alpha Testing**
  - Alpha version is early version of the application tested by “internal” users
- **Beta Testing**
  - Beta version of the application used by selected users
- Used to get users feedback before releasing the application widely
Other Testing Types

Pilot Testing
• Testing certain features of the application
• Proof of concept

Parallel Testing
• Feeding test data to two systems, typically one existing and other new
• Validates the data input and output through two systems

Sociability Testing
• Ensure the new or modified system can operate in the target environment without affecting other systems in the ecosystem
Example of Testing Tools

• Quality Center (HP) – For Test Case Management, Execution, Defects Management

• ServiceTest, QuickTestPro (HP) – For Service and UI Testing Automation

• LoadRunner (HP) – For Performance Testing
Question

Which of the following applies to regression testing?

A. Its purpose is to test internal program logic
B. It’s a means to test white-box functions
C. It ensures that the change did not introduce a new problem
D. It’s a means to test black-box functions
Upcoming Assignments/Tests

1. Individual Case Study -2 (Requirements and Use Case): Mon 10/20 before the class


3. Group Case Study -2 (Requirements): Mon 11/3 before the class

Questions?
Summary of Today’s Class

• QA
• Testing
• Focus of the Next Class and Reading
• Questions