Web Applications Testing
Ario Nejad, Christopher Choi
What is a Web Application?

Though the boundaries of what constitute a web application are vague, it is commonly perceived to be an application that utilizes a web browser as its medium.

A web application differentiates itself from a web site by serving a purpose beyond simply displaying information to a user.
Why do Web Applications Matter?

Web applications, increasingly, serve as the interface between a user and online transactions.

- Growth of cloud based technologies.
- The advent of the SaaS model and prominent use of thin clients.
- Coming full circle back to mainframe time-sharing systems?
Web Applications VS Traditional Software

- Number of users (pressure on web servers)
- Heterogeneous and autonomous client environment (compatibility)
- Performance (focus on information searching and indexing)
- Rate of content change (dynamic)
Rapid Application Development (RAD)

- RAD is a software development methodology that uses minimal planning in favor of rapid prototyping.
- The lack of extensive pre-planning generally allows software to be written much faster, and makes it easier to change requirements.
- Applying RAD often results in the lack of necessary documents come testing.
RAD Model
Testing Process for Traditional Software

1. Determine the testing objects and targets.
2. Generate the testing input.
3. Produce the expected output.
4. Execute the test cases and validate.
5. Amend the software and do regression testing.
The sequence and process for web application testing is generally the same as the traditional ones, only different in the realization techniques and methods.
Modeling the Testing Process for Web Applications

- Determine testing requirements based on application under testing (target, object, and method)
- Select test cases (input information, expected output, and test steps)
- Execute test cases under client side equipment (executor, monitor)
- Results analysis and comparison work
Testing Framework for Web Applications
A web application consists of multiple pages. Each page contains numerous page elements such as hyperlinks, displaying elements, grids, scripts, components, and forms. We need a web application model to determine the structure and relationship of the testing objects.
Web Application Diagram
Based on the detailed testing requirements, select relevant testing objects and choose the applicable testing methods.

- Functionality testing
- Performance testing
- Safety testing
- Usability testing
- Compatibility testing
- White box, black box, or gray box testing
Generating and Selecting Test Cases

- First the condition section (listing conditions and combinations of conditions).
  - Input variables to the page, Input Actions, and State before test execution.
- Second the action section (listing responses to be produced when corresponding combinations of conditions are true).
  - Expected Results, Expected Output Actions, and Expected State after test.
Executing Testing

- Testing should be executed on the client-side in order to model real-world usage.
- The strictly defined sequence of actions as defined in the testing requirements phase allows for automation.
- Improving testing intelligence is necessary in order to effectively deal with errors, due to the multi-layered nature of a web application.
Analyzing and Measuring Test Results

The results returned from the testing process are compared to a set of expected output values in order to determine correctness. However, due to the dynamic nature of web pages and the variety of environments the pages are displayed in, the individual elements are compared 1 to 1 in order to obtain an accurate assessment.
Modeling a Web Application to Meet Testing Requirements

The process of modeling a web application involves the use of objects, relationships, and states. This often results in a jumble of contents and structures. By using object, interactive relation, and architecture models; we can expect reasonable structures, full-aspect contents, and clear relationships. The models also correspond with unit, integration, and system testing, respectively.
Web Application Testing Model
Web Application Testing Model Cont.

- Object model - involved in functional testing.
- Interactive relation model - involved in testing of dynamic properties and used to describe transfer of object states.
- Architecture model - involved in performance testing.
Due to the use of the RAD cycle, web applications often lack the full set of design and analysis documents to generate a full suite of tests.

Testing requirements are instead gathered by:
- Directly analyzing the HTML page
- Denote the elements, components, and relationships from the page code
- Represent the objects and relationships graphically
Strategies for Generating and Selecting Test Cases

1. Combining the web application model with the testing requirements gives us testing paths. Testing paths allow us to determine the sequence in which the test is to be performed.

2. Collecting information about the objects allows us to determine input.

3. Expected output is determined during the analysis of the web application and testing requirements. Dynamic pages are tested in advance several times in order to generate a baseline.

4. Test cases are reduced using combinatorial testing methods.
Combinatorial testing is a test method that tests all possible discrete combinations of the input parameters using carefully chosen test vectors.
Intelligent Automated Testing

The testing process follows a rigid guideline making it so that manual testing is only useful when exception handling. Automated testing can be done by utilizing capture-replay tools which allow the testing sequence to be scripted. Machine learning allows for the automated testing agent to better handle dynamically generated pages.
Testing Execution Workflow

1. The testing monitor agent chooses corresponding test scenes from the test-side agent's repository depending on the testing target.
2. The monitor agent determines the testing steps and executing behavior, combined with the input parameters provided by the test case.
3. The execution agent visits the web pages under testing by implementing the testing sequence.
4. The execution agent outputs the test results to the monitor agent.
Testing Execution Workflow Diagram
Analyzing and Measuring Test Results

Due to the difficulty of comparing returned vs expected results page elements are removed and an XML format is applied to compare structure and data. In this way data can be directly compared instead of focusing on visual elements.

The quality of the web application can then be determined by quantitatively analyzing the test results to determine errors and faults alongside other metrics.
Questions?