# SOFTWARE ENGINEERING II

## LECTURE: 5

## TESTING WEB APPLICATIONS

# TESTING QUALITY DIMENSIONS

***Content* is evaluated at**

- **syntactic level:** spelling, punctuation and grammar are assessed for text-based documents.
- **semantic level:** correctness (of information presented), consistency (across the entire content object and related objects) and lack of ambiguity are all assessed.

***Function* is tested for**

- correctness, instability, and general conformance to appropriate implementation standards.

***Structure* is assessed to ensure that it**

- properly delivers WebApp content and function
- is extensible
- can be supported as new content or functionality is added.

# TESTING QUALITY DIMENSIONS

***Usability* is tested to ensure that each category of user**

- is supported by the interface
- can learn and apply all required navigation syntax and semantics

***Navigability* is tested**

- all navigation syntax and semantics are exercised to uncover any navigation errors (e.g., dead links, improper links, erroneous links).

***Performance* is tested under a variety of operating conditions, configurations, and loading to ensure that**

- the system is responsive to user interaction
- the system handles extreme loading without unacceptable operational degradation

## TESTING QUALITY DIMENSIONS

***Compatibility* is tested by executing the WebApp in a variety of different host configurations on both the client and server sides.**

- The intent is to find errors that are specific to a unique host configuration.

***Interoperability* is tested to ensure that**

- the WebApp properly interfaces with other applications and/or databases.

***Security* is tested**

- *Assess potential vulnerabilities*
- *Any successful penetration attempt is deemed a security failure*

# CONTENT TESTING

A student googled for "JAVA Basics" and found many search results

She clicked on the first link

There's a topic titled as Java Basics on the website. But when she started to read she found it was not related to JAVA.

Will she return to the website now?

## CONTENT TESTING

Content testing has three important objectives:

- to uncover syntactic errors (e.g., typos, grammar mistakes) in text-based documents, graphical representations, and other media
- to uncover semantic errors (i.e., errors in the accuracy or completeness of information) in any content object presented as navigation occurs, and
- to find errors in the organization or structure of content that is presented to the end-user.

## ASSESSING CONTENT SEMANTICS

Is the information factually accurate?

Does the content infringe on existing copyrights or trademarks?

Have proper references been provided for all information derived from other sources?

Is the content offensive, misleading, or does it open the door to litigation?

# DATABASE TESTING

**Functional Database Testing** is a type of database testing that is used to validate the functional requirements of a database from the end-user's perspective.

The main goal of functional database testing is to test whether the transactions and operations performed by the end-users which are related to the database works as expected or not.

**Whether the field is mandatory while allowing NULL values on that field?**

**Whether the length of each field is of sufficient size?**

## USER INTERFACE TESTING

- Interface features are tested to ensure that design rules, aesthetics, and related visual content is available for the user without error.
- Individual interface mechanisms are tested in a manner that is analogous to unit testing.
- Each interface mechanism is tested within the context of a use-case for a specific user category and the complete interface is tested against selected use-cases.
- The interface is tested within a variety of environments (e.g., browsers) to ensure that it will be compatible.

## TESTING INTERFACE MECHANISMS

**Links—**navigation mechanisms that link the user to some other content object or function.

**Forms—**a structured document containing blank fields that are filled in by the user. The data contained in the fields are used as input to one or more WebApp functions.

- form fields have proper width and data types
- form establishes appropriate safeguards that preclude the user from entering text strings longer than some predefined maximum
- all appropriate options for pull-down menus are specified and ordered in a way that is meaningful to the end user
- browser “auto-fill” features do not lead to data input errors
- tab key (or some other key) initiates proper movement between form fields.

## TESTING INTERFACE MECHANISMS

**Dynamic HTML**—leads to content objects that are manipulated on the client side using scripting or cascading style sheets (CSS).

- Change of color of a text heading when a user passes a mouse over it
- Allowing a user to "drag and drop" an image to another place on a Web page

**Client-side pop-up windows**—small windows that pop-up without user interaction. These windows can be content-oriented and may require some forms of user interaction.

- the pop-up is properly sized and positioned
- the pop-up does not cover the original WebApp window

# USABILITY TESTS

Rather than focusing intently on the semantics of some interactive objective, usability reviews and tests are designed to determine the degree to which the WebApp interface makes the user's life easy

Conducted by End Users

Develop a mechanism for different levels of abstraction:

- the usability of a specific interface mechanism
- the usability of a complete Web page
- the usability of the complete WebApp

## USABILITY TESTS

- ***Interactivity***—Are interaction mechanisms (e.g., pull-down menus, buttons, pointers) easy to understand and use?
- ***Layout***—Are navigation mechanisms, content, and functions placed in a manner that allows the user to find them quickly?
- ***Readability***—Is text well written and understandable? Are graphic representations easy to understand?
- ***Aesthetics***—Do layout, color, typeface, and related characteristics lead to ease of use?
- ***Personalization***—Does the WebApp tailor itself to the specific needs of different user categories or individual users?
- ***Accessibility***—Is the WebApp accessible to people who have disabilities?

## CONFIGURATION TESTING

**Server-side**

- Is the WebApp fully compatible with the server OS?
- Are system files, directories, and related system data created correctly when the WebApp is operational?
- Do system security measures (e.g., firewalls or encryption) allow the WebApp to execute and service users without interference or performance degradation?
- Is the WebApp properly integrated with database software? Is the WebApp sensitive to different versions of software?
- Do server-side WebApp scripts execute properly?

## CONFIGURATION TESTING

***Client-side***

- Hardware: CPU, memory, storage and printing devices
- Operating systems: Linux, Macintosh OS, Microsoft Windows, a mobile-based OS
- Browser: Internet Explorer, Mozilla/Netscape, Opera, Safari
- Plug-ins: QuickTime, RealPlayer, and many others
- Connectivity: cable, DSL, regular modem

# SECURITY TESTING

**Vulnerabilities**

- client-side environment
- server-side environment

**On the client-side**

- can often be traced to pre-existing bugs in browsers, email programs, or communication software

**On the server-side**

- denial-of-service attacks
- malicious scripts
- can be passed along to the client-side or used to disable server operations

## PERFORMANCE TESTING

- Does the server response time degrade to a point where it is noticeable and unacceptable?
- At what point (in terms of users, transactions or data loading) does performance become unacceptable?
- What system components are responsible for performance degradation?
- What is the average response time for users under a variety of loading conditions?
- Does performance degradation have an impact on system security?
- Is WebApp reliability or accuracy affected as the load on the system grows?
- What happens when loads that are greater than maximum server capacity are applied?

# LOAD TESTING

The intent is to determine how the WebApp, and its server-side environment will respond to various loading conditions

N, the number of concurrent users

T, the number of on-line transactions per unit of time

D, the data load processed by the server per transaction

Overall throughput, P, is computed in the following manner:

$$P = N \times T \times D$$

## LOAD TESTING EXAMPLE

Consider a popular sports news site. At a given moment, 20,000 concurrent users submit a request (a transaction, T) once every 2 minutes on average.

Each transaction requires the WebApp to download a new article that averages 3K bytes in length.

Therefore, throughput can be calculated as:

P = [20,000 × 0.5 × 3Kb]/60 = 500 Kbytes/sec

The network connection for the server would therefore have to support this data rate and should be tested to ensure that it does

# STRESS TESTING

**Stress testing is a continuation of load testing, but in this instance the variables, N, T, and D are forced to meet and then exceed operational limits**

Does the system degrade ‘gently’ or does the server shut down as capacity is exceeded?

Does server software show “server not available” message?

Are transactions lost as capacity is exceeded?

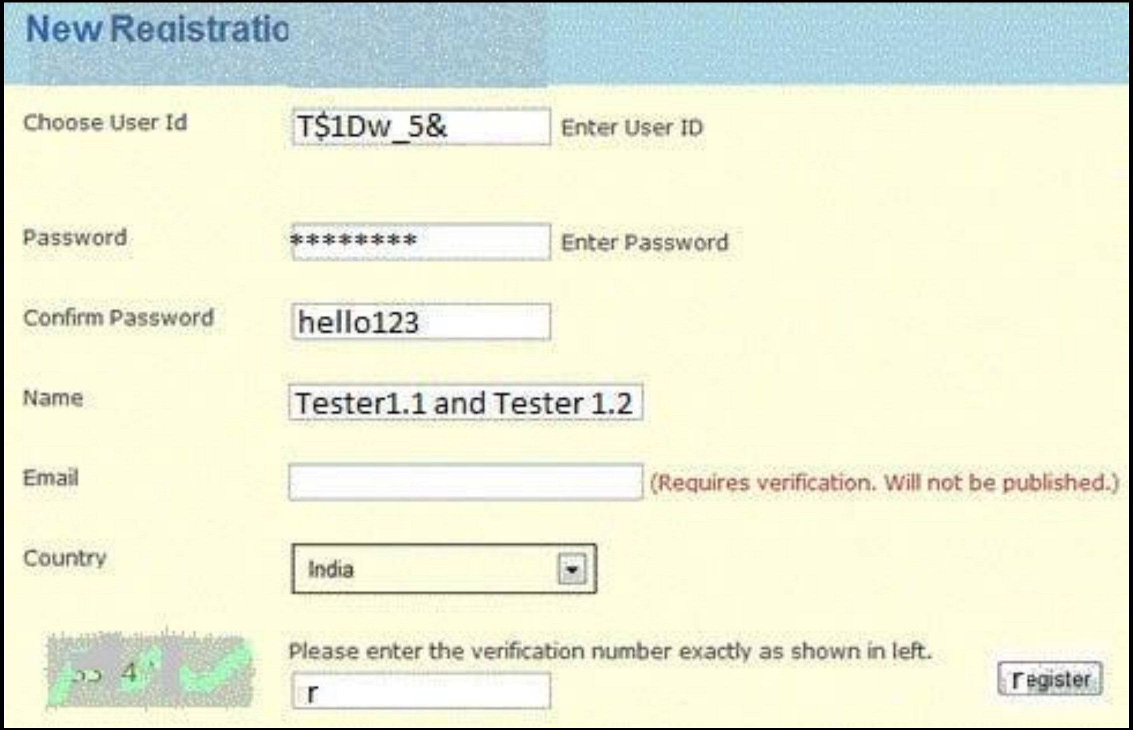
What values of N, T, and D force the server environment to fail? How does failure manifest itself? Are automated notifications sent to technical support staff at the server site?

If the system does fail, how long will it take to come back on-line?

Are certain WebApp functions (data streaming capabilities) discontinued as capacity reaches the 80 or 90 percent level?

# EXAMPLE 1

What bugs you find in this form? List all of them.



New Reqistratio

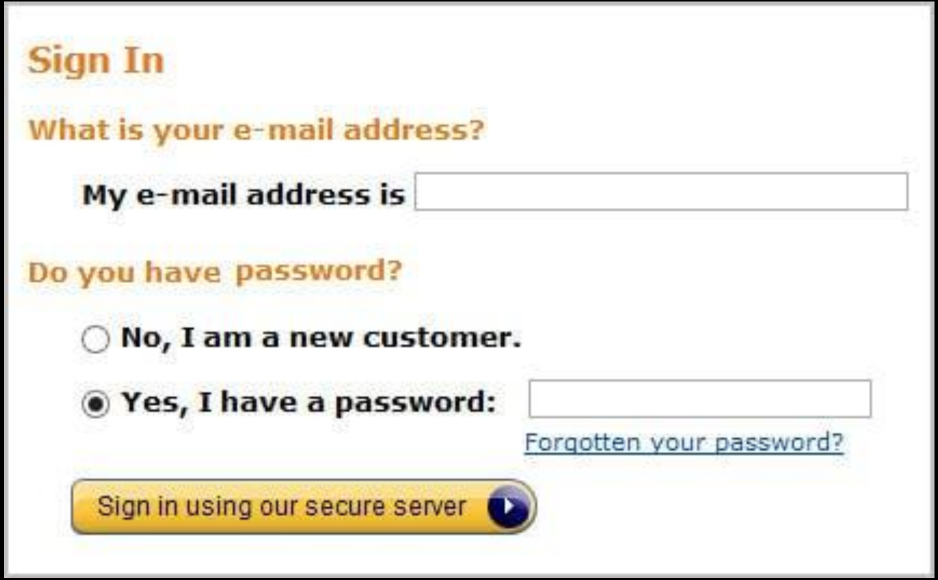
| | | |
|---|---|---|
| Choose User Id | T$1Dw_5& | Enter User ID |
| Password | ********* | Enter Password |
| Confirm Password | hello123 | |
| Name | Tester1.1 and Tester 1.2 | |
| Email | | (Requires verification. Will not be published.) |
| Country | India | |

Please enter the verification number exactly as shown in left.

r

# EXAMPLE 2

How would you improve the login page?



Sign In

What is your e-mail address?

My e-mail address is

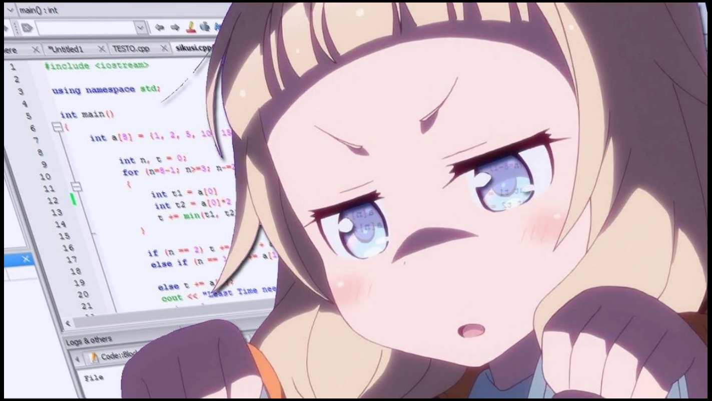
Do you have password?

No, I am a new customer.

[✓ Yes, I have a password:]

Forgotten your password?

# SOFTWARE RISKS



## Uncertainty

- The risk may or may not happen; that is, there are no 100 percent probable risks.

## Loss

- If the risk becomes a reality, unwanted consequences or losses will occur.

## CATEGORY OF RISKS

**Project risks t**hreaten the project plan. That is, if project risks become real, it is likely that the project schedule will slip and that costs will increase. Project risks identify potential budgetary, schedule, personnel (staffing and organization), resource, stakeholder, and requirements problems and their impact on a software project.

**Technical risks** threaten the quality and timeliness of the software to be produced. If a technical risk becomes a reality, implementation may become difficult or impossible. Technical risks identify potential design, implementation, interface, verification, and maintenance problems.

## CATEGORY OF RISKS

**Business risks are**

- Building an excellent product or system that no one really wants (market risk)
- Building a product that no longer fits into the overall business strategy for the company (strategic risk)
- Building a product that the sales force doesn’t understand how to sell (sales risk)
- Losing the support of senior management due to a change in focus or a change in people (management risk)
- Losing budgetary or personnel commitment (budget risks).

# CATEGORY OF RISKS

**Known risks** are those that can be uncovered after careful evaluation of the project plan, the business and technical environment in which the project is being developed, and other reliable information sources (e.g., unrealistic delivery date, lack of documented requirements or software scope, poor development environment).

**Predictable risks** are extrapolated from past project experience (e.g., staff turnover, poor communication with the customer, dilution of staff effort as ongoing maintenance requests are serviced).

**Unpredictable risks** can and do occur, but they are extremely difficult to identify in advance.

## RISK EXPOSURE (IMPACT)

The overall risk exposure, RE, is determined using the following relationship:

$$RE = P \times C$$

where

P is the probability of occurrence for a risk, and

C is the cost to the project should the risk occur.

# RISK EXPOSURE EXAMPLE

Risk identification. Only 70 percent of the software components scheduled for reuse will, in fact, be integrated into the application. The remaining functionality will have to be custom developed.

- Risk probability. 80% (likely).
- Risk impact. 60 reusable software components were planned. If only 70 percent can be used, 18 components would have to be developed from scratch (in addition to other custom software that has been scheduled for development). Since the average component is 100 LOC and local data indicate that the software engineering cost for each LOC is $14.00, the overall cost (impact) to develop the components would be 18 × 100 × 14 = $25,200.
- Risk exposure. RE = 0.80 × 25,200 ~ $20,200.

# END OF LECTURE 5

LECTURE: 5

TESTING WEB APPLICATIONS