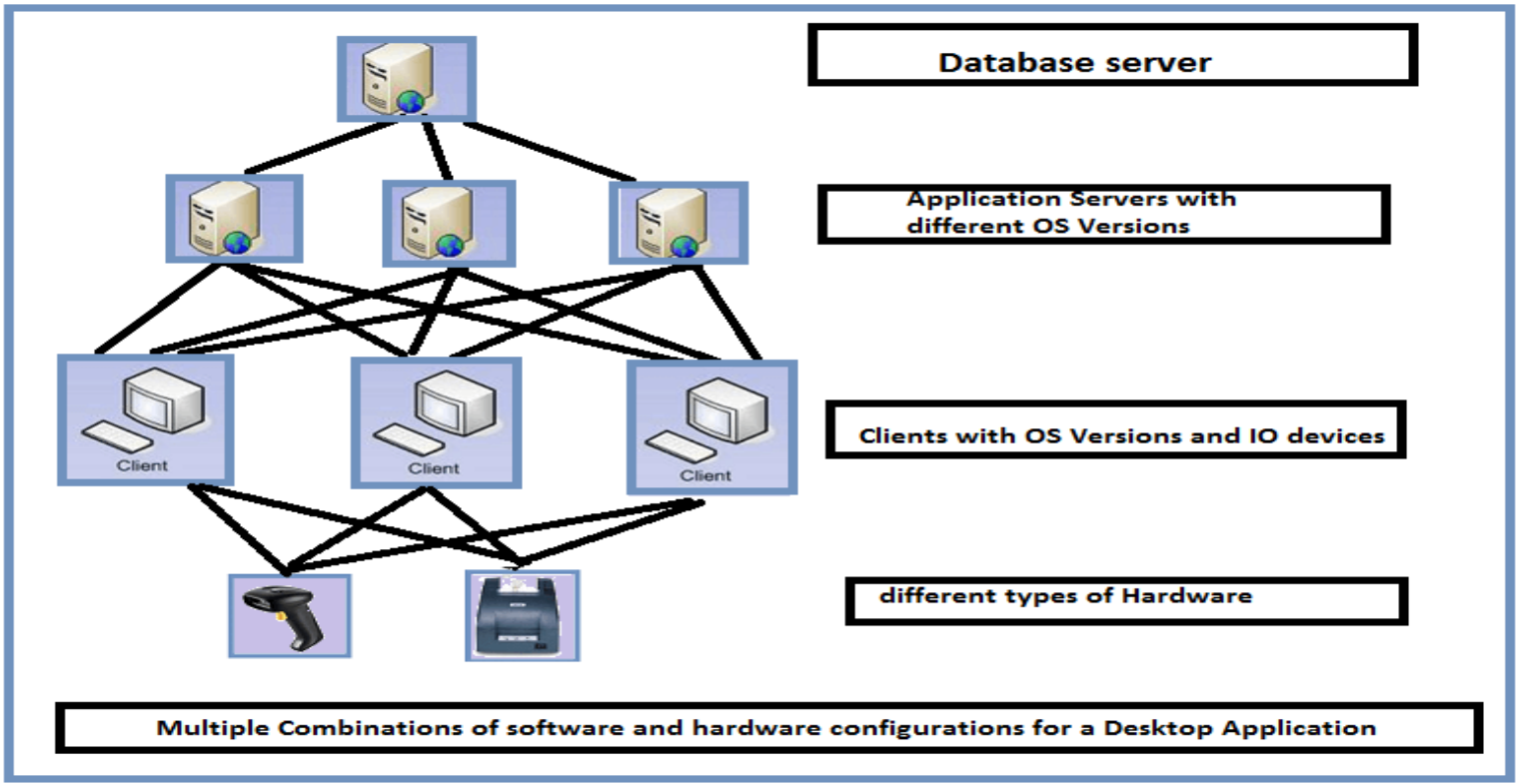
Chapter-IV
Applying Your Testing Skills

- *Configuration testing :* Configuration testing is the process of testing the system with each one of the supported software and hardware configurations.

- Executing Tests with Various Configurations:
- Operating System Configuration - Win XP, Win 7 32 bit/64 bit, Win 8 32 bit/64 bit
- Database Configuration - Oracle, DB2, MySql, MSSQL Server, Sybase
- Browser Configuration - IE 8, IE 9, FF 16.0, Chrome

- **Configuration Testing** is a software testing technique in which the software application is tested with multiple combinations of software and hardware in order to evaluate the functional requirements and find out optimal configurations under which the software application works without any defects or flaws.



- Objectives of Configuration Testing

1. Validating the application to determine if it fulfils the configurability requirements
2. Manually causing failures which help in identifying the defects that are not efficiently found during testing
3. Determine an optimal configuration of the application under test.
4. Analysing the system performance by adding or modifying the hardware resources.
5. Analysing system Efficiency based on the prioritization.
6. Verification of the system in a geographically distributed Environment to verify how effectively the system performs.

- Approaching the Task: general process that you should use when planning your configuration testing.

1. Decide the Types of Hardware You'll Need
2. Decide What Hardware Brands, Models, and Device Drivers Are Available.
3. Decide Which Hardware Features, Modes, and Options Are Possible.
4. Pare Down the Identified Hardware Configurations to a Manageable Set.
5. Identify Your Software's Unique Features That Work with the Hardware Configurations.
6. Design the Test Cases to Run on Each Configuration.
7. Execute the Tests on Each Configuration.
8. Rerun the Tests Until the Results Satisfy Your Team.
9. Obtaining the Hardware
10. Identifying Hardware Standards
11. Configuration Testing Other Hardware

- Configuration Testing Types:

1. Software Configuration Testing
2. Hardware Configuration Testing

- Compatibility Testing: Checking the functionality of an application on different software, hardware platforms, network, and browsers is known as compatibility testing.

- Types of Compatibility testing

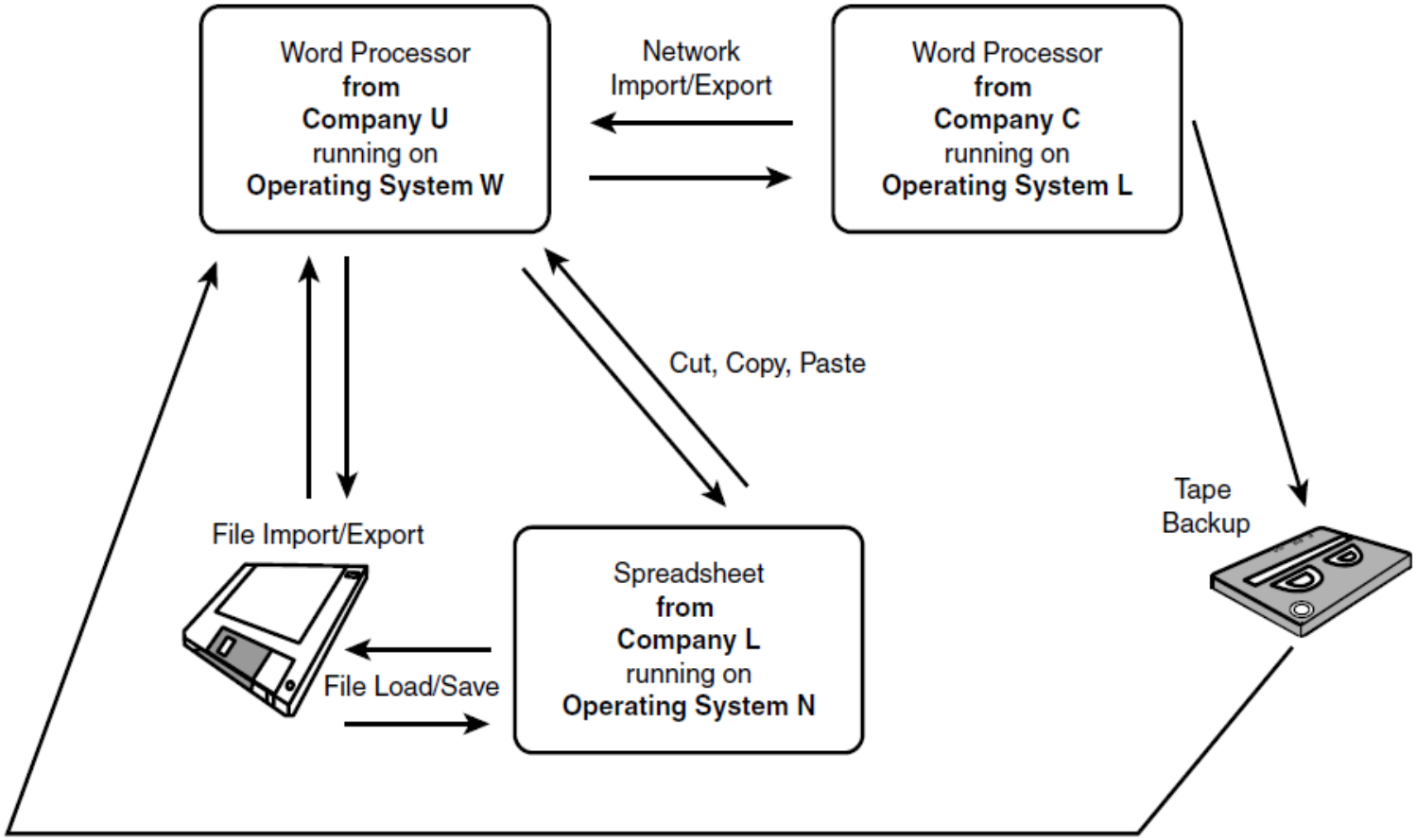
1. Software :Browser/Platoform-os/Versions
2. Hardware
3. Network
4. Mobile

- *Software compatibility testing* means checking that your software interacts with and shares information correctly with other software.

Examples of compatible software are

1. Cutting text from a Web page and pasting it into a document opened in your word processor.

2. Saving accounting data from one spreadsheet program and then loading it into a completely different spreadsheet program


Word Processor
from
Company U
running on
Operating System W
Network
Import/Export
Word Processor
from
Company C
running on
Operating System L
Cut, Copy, Paste
Tape
Backup
File Import/Export
Spreadsheet
from
Company L
running on
Operating System N
File Load/Save

- **Backward and Forward Compatibility:**
- If something is backward compatible, it will work with previous versions of the software.

- If something is forward compatible, it will work with future versions of the software.
- a .txt or text file. With Notepad 98

1. Windows 98 is backward compatible all the way back to MS-DOS 1.0.
2. It’s also forward compatible to Windows 2000

- Data Sharing Compatibility: A well-written Program that supports and adheres to published standards and allows users to easily transfer data to and from other software is a great compatible product.

1. *File save* and *file load* are the data-sharing methods.
2. *File export* and *file import*
3. *Cut*, *copy*, and *paste* are the most familiar methods for sharing data among programs without transferring the data to a disk.
4. *DDE(*Dynamic Data Exchange*) & OLE(*Object Linking and Embedding*)*

- Clipboard is that with DDE and OLE, data can flow from one application to the other in real time. Cutting and copying is a manual operation. With DDE and OLE, the transfer can happen automatically.

- User Interface Testing: **GUI Testing** is a software testing type that checks the Graphical User Interface of the Software. The purpose of Graphical User Interface (GUI) Testing is to ensure the functionalities of software application work as per specifications by checking screens and controls like menus, buttons, icons, etc.
- *Usability* is how appropriate, functional, and effective that interaction is.
- The means that you use to interact with a software program is called its *user interface*, or UI.
- Command Line Interface is where you type text and computer responds to that command.
- GUI stands for Graphical User Interface

- Traits common to a good UI

1. Follows standards and guidelines.
2. Flexible
3. Correct
4. Intuitive
5. Comfortable
6. Useful
7. Consistent

1. Follows Standards or Guidelines: These standards and guidelines were developed by experts in software usability.

- The single most important user interface trait is that your software follows existing standards and guidelines.

- If your software strictly follows the rules, most of the other traits of a good UI will happen automatically.

- Intuitive:

1. Is the user interface clean, unobtrusive, not busy?
2. Is the UI organized and laid out well?
3. Is there excessive functionality?
4. If all else fails, does the help system really help you?

- **Consistent:** Consistency within your software and with other software is a key attribute.

1. Users develop habits and expect that if they do something a certain way in one program, another will do the same operation the same way.

- **Shortcut keys and menu selections.**
- **Terminology and naming**
- **Audience**
- **Placement and keyboard equivalents for buttons**

- Flexible: Users like choices—not too many, but enough to allow them to select what they want to do and how they want to do it.
- Of course, with flexibility comes complexity.

1. **State jumping:** Flexible software provides more options and more ways to accomplish the same task.
2. **State termination and skipping:** software can skip numerous prompts or windows and go directly to where they want to go.
3. **Data input and output.** Users want different ways to enter their data and see their results.

- Comfortable: Software should be comfortable to use.

1. **Appropriateness.** Software should look and feel proper for what it's doing and who it's for.

1. **Error handling.** A program should warn users before a critical operation and allow users to restore data lost because of a mistake.

1. **Performance.** Being fast isn't always a good thing.

- **Correct:** When you're testing for correctness, you're testing whether the UI does what it's supposed to do.

1. **Marketing differences.** Are there extra or missing functions, or functions that perform operations different from what the marketing material says?
2. **Language and spelling.** Programmers know how to spell only computer language keywords and often create some very interesting user messages.
3. **Bad media.** Media is any supporting icons, images, sounds, or videos that go with your software's UI.
4. **WYSIWYG (what you see is what you get).** Make sure that whatever the UI tells you that you have is really what you do have.

- Useful: The final trait of a good user interface is whether it's useful.

1. The features you see actually contribute to the software's value .
2. Do they help users do what the software is intended to do?

- **Testing for the Disabled: Accessibility Testing:** A serious topic that falls under the area of usability testing is that of *accessibility testing*, or testing for the disabled.

1. Visual impairments
2. Hearing impairments
3. Motion impairments
4. Cognitive and language

Checklist

- Check all the GUI elements for size, position, width, length, and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.
- Check you can execute the intended functionality of the application using the GUI
- Check Error Messages are displayed correctly
- Check for Clear demarcation of different sections on screen
- Check Font used in an application is readable
- Check the alignment of the text is proper
- Check the Color of the font and warning messages is aesthetically pleasing
- Check that the images have good clarity
- Check that the images are properly aligned
- Check the positioning of GUI elements for different screen resolution.

Examples

- Testing the colors of the fonts.
- Testing the colors of the error messages, warning messages.
- Testing whether the image has good clarity or not.
- Testing the alignment of the images.
- Testing of the spelling.
- The user must not get frustrated while using the system interface.
- Testing whether the interface is attractive or not.
- Testing of the scrollbars according to the size of the page if any.
- Testing of the disabled fields if any.
- Testing of the size of the images.
- Testing of the headings whether it is properly aligned or not.
- Testing of the color of the hyperlink.

- Website Testing: Web testing is a software testing practice to test websites or web applications for potential bugs. It's a complete testing of web-based applications before making live.
- A web-based system needs to be checked completely from end-to-end before it goes live for end users.
- Web Testing checks for

1. Functionality
2. usability
3. Security
4. compatibility
5. Interface
6. performance of the web application or website.

- Web page features. A partial list of them includes

1. Text of different sizes, fonts, and colors
2. Graphics and photos
3. Hyperlinked text and graphics
4. Varying advertisements
5. Drop-down selection boxes
6. Fields in which the users can enter data

- Black-Box Testing:

1. **Text: Font/Colour /Alignment/Corner/Alt Text/visibility**
2. **Links:** Outgoing links/Internal links/Anchor Links/MailTo Links/*orphan pages*
3. **Forms :**Forms are the text boxes, list boxes, and other fields for entering or selecting information on a Web page.
4. **Cookies**
5. **Test HTML and CSS**
6. **Graphics**
7. **Objects and Other Simple Miscellaneous Functionality**

- Gray-Box Testing: Gray Box Testing is a software testing method, which is a combination of both White Box Testing and Black Box Testing method.

- Gray box testing is a software testing technique to test a software product or application with partial knowledge of internal structure of the application. The purpose of grey box testing is to search and identify the defects due to improper code structure or improper use of applications.

- White-Box Testing:

1. **Dynamic Content.** Dynamic content is graphics and text that changes based on certain conditions—for example, the time of day, the user's preferences, or specific user actions.
2. **Database-Driven Web Pages.** Many e-commerce Web pages that show catalogs or inventories are database driven.
3. **Programmatically Created Web Pages.**
4. **Server Performance and Loading.**
5. **Security**

- **Configuration and Compatibility Testing:** Web pages are perfect examples of where you can apply this type of testing.

1. **Hardware Platform**
2. **Browser Software and Version.**
3. **Browser Plug-Ins.**
4. **Browser Options**
5. **Video Resolution and Color Depth**
6. **Text Size**
7. **Modem Speeds**

- **Usability Testing:** The following list is adapted from his *Top Ten Mistakes in Web Design*:

1. **Gratuitous Use of Bleeding-Edge Technology.**
2. **Scrolling Text, Marquees, and Constantly Running Animations**
3. **Long Scrolling Pages.**
4. **Non-Standard Link Colors**
5. **Outdated Information.**
6. **Overly Long Download Times**
7. **Lack of Navigation Support**
8. **Orphan Pages.**
9. **Complex Web Site Addresses (URLs).**
10. **Using Frames**

- Interface Testing: Three areas to be tested here are - Application, Web and Database Server.

- Database Testing:Database is one critical component of your web application and stress must be laid to test it thoroughly.

- **Performance testing -** Performed to verify the server response time and throughput under various load conditions.
- **Load testing -** It is the simplest form of testing conducted to understand the behaviour of the system under a specific load.
- **Stress testing -** It is performed to find the upper limit capacity of the system and also to determine how the system performs if the current load goes well above the expected maximum.
- **Soak testing -** Soak Testing also known as endurance testing, is performed to determine the system parameters under continuous expected load. The main aim is to discover the system's performance under sustained use.
- **Spike testing -** Spike testing is performed by increasing the number of users suddenly by a very large amount and measuring the performance of the system. The main aim is to determine whether the system will be able to sustain the work load.

- **Security testing -** Performed to verify if the application is secured on web as data theft and unauthorized access are more common issues and below are some of the techniques to verify the security level of the system.
- Injection
- Broken Authentication and Session Management
- Cross-Site Scripting (XSS)
- Insecure Direct Object References
- Security Misconfiguration
- Sensitive Data Exposure
- Missing Function Level Access Control
- Cross-Site Request Forgery (CSRF)
- Using Components with Known Vulnerabilities
- Unvalidated Redirects and Forwards