System Development Methods

CT00046-3-2

System Implementation (Construction and Testing)

Learning Outcome

- At the end of the module, you should able to:
  - Explain the purpose, structure and scope of modern Information System Development Methodologies and select and justify appropriate methods of analysis, design and implementation for a particular component or application, be it traditional, multimedia or web based.

Key Terms you must be able to use

- If you have mastered this topic, you should be able to use the following terms correctly in your assignments and exams:
  - System Construction
  - Good Programming Practice
  - System Testing

System Construction

- The process of building the rest of the system through a programming language.

- Done when;
  - Design of the system is approved.
  - Design Specification is available.

- Deliverable
  - Fully working software / system

Coding Strategies

Things to Consider Before Coding

- Software Architecture to be used / Platform (s)
- Choice Programming Language(s)
- Programming / Coding Standards
- Types of testing to be carried out.
- User Involvement
- Code Repository
- Version Control
- Security and Copyright

Coding Strategies

Things to Consider Before Coding

- Software Architecture to be used / Platform (s)
  - Types of Platform to build / supported by your application.
- Choice Programming Language(s)
  - The most flexible/compatible language based on your requirements
- Programming / Coding Standards
  - Check whether your coding need to comply with certain standards, IEEE, Open Source, etc.
- Types of testing to be carried out.
  - Type of testing necessary for Pre and Post product deployment.

Coding Strategies

Things to Consider Before Coding

- User Involvement
  - Degree of user involvement, availability and user's expertise
- Code Repository
  - Where to store codes, secure and sharable to other developers.
- Version Control
  - Control of modifications made on codes,
  - backup and restore or workable codes)
- Security and Copyright
  - Level of security implemented for the system / application.
  - Artifacts that needed to be patented / copyright.

Good Programming Practice

- Efficiency
  - Keep it Short and Simple
- Portability
  - Use good variables and no hard codes
- Security
  - Source Codes, Encapsulation
- Readable codes
  - meaningful and informative
  - Names, Comments, Formats
- Refactoring
  - Further Reading;
  - *https://msdn.microsoft.com/en-us/library/aa260844(v=vs.60).aspx*

Software Testing

- A process of executing a program or application with the intent of finding the software bugs.

- Aka; A process of validating and verifying that a software program or application or product: Meets the business and technical requirements that guided it's design and development.

Software Testing Objectives

- Finding defects / bugs.
- Prevent defects (avoid expensive recovery)
- Improving level of quality (code and product)
- To make sure that the end result meets the business and user requirements (in specifications).

Testing Approach

- **Static Testing**
  - Test and find defects without executing code.
  - Done during verification process (verify requirements)
  - Includes techniques - reviewing of the documents, static analysis, reviewing, walkthrough, inspection, etc.
- **Dynamic Testing**
  - Software code is executed.
  - Done during validation process (satisfy customers)
  - Includes test types - Unit Testing, Integration Testing, System Testing, etc.
  - Recommended – automated testing tools.

Common Testing Terms

- White Box Testing
  - Testing where all INPUT, PROCESS (code functions) and OUTPUT is seen.
  - Usually test by programmer.
- Black Box testing
  - Testing where only INPUT and OUTPUT is seen, PROCESS (internal code and functions) are hidden.
  - Usually test by User / Customer.
- Stub Testing
  - Testing just one line of execution, normally to check presence of data

Testing Level

Unit Testing

Integration Testing

Component Interface Testing

System Testing.

Testing Level

- **Unit testing**
  - Aka; component testing
  - Tests that verify the functionality of a specific section of code, usually at the function level.
  - (In an object-oriented environment, this is usually at the class level)
  - One function might have multiple tests
  - Ensure that the building blocks of the software work independently from each other.
  - Variation of white-box testing

Testing Level

- **Integration Testing**
  - Any type of software testing that seeks to verify the interfaces between components against a software design.
  - Works to expose defects in the interfaces and interaction between integrated components (modules).
  - Larger software components test the architectural design

Testing Level

- **Component Interface Testing**
  - To check the handling of data passed between various units, or subsystem components
  - Help explain unexpected performance in the next unit.
  - Variation of black-box testing

- **System Testing**
  - Tests a completely integrated system to verify that it meets its requirements
  - Variation of black-box testing

Popular Testing Types

- **Installation Testing**
  - assures that the system is installed correctly
- **Compatibility Testing**
  - compatibility with other application software, operating systems
- **Smoke And Sanity Testing**
  - determines whether it is reasonable to proceed with further testing
- **Regression Testing**
  - focuses on finding defects after a major code change has occurred.
- **Acceptance Testing**
  - performed by the customer for approval and feedback

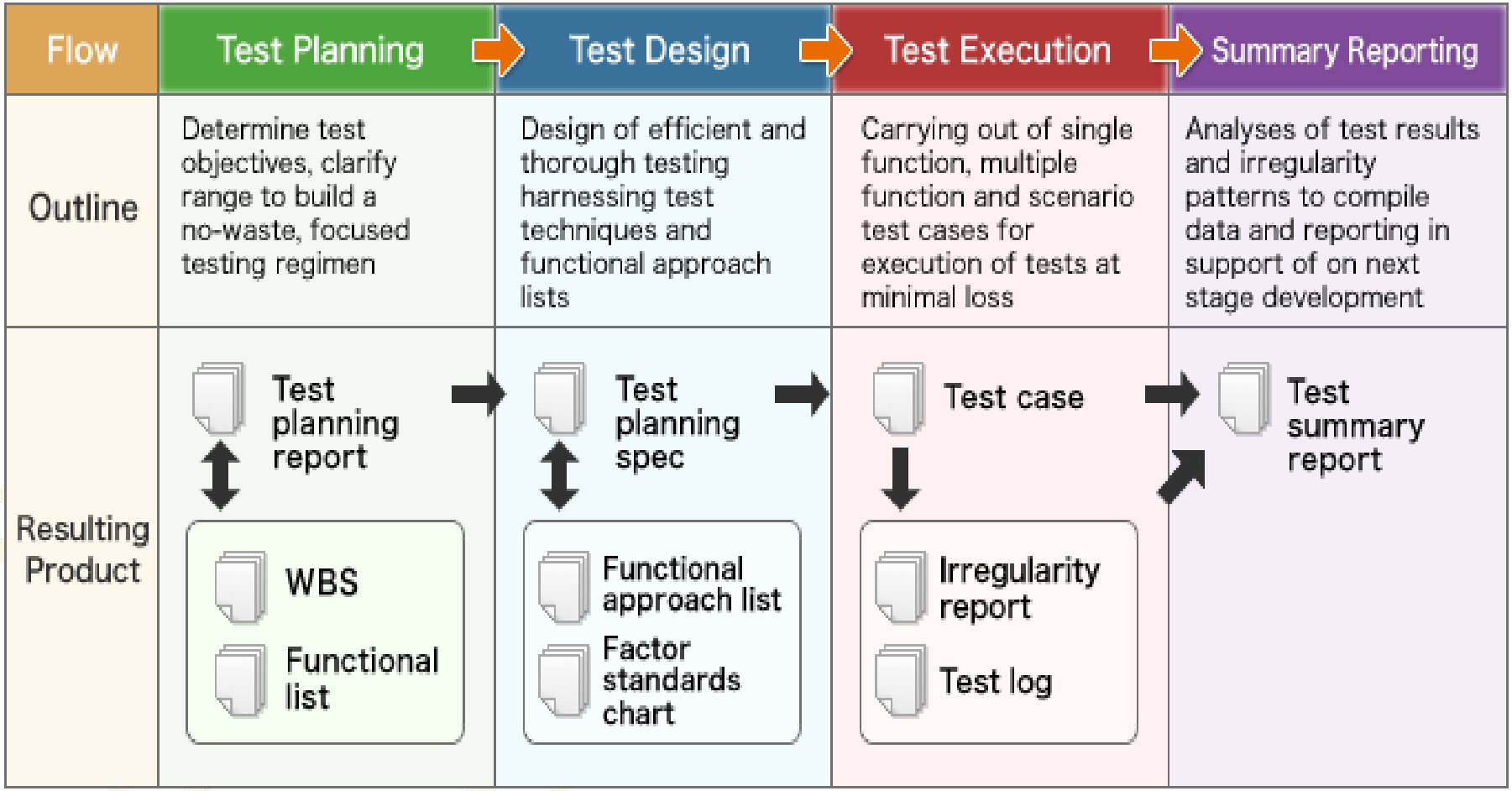
Popular Testing Types .. cont

- **Alpha Testing**
  - simulated or actual operational testing by potential selected / internal users
- **Beta Testing**
  - after alpha testing, Beta testing is a form of external user acceptance testing
- **Destructive / Robust Testing**
  - test software durability and tolerance.
- **Security Testing**
  - for software that processes confidential data to prevent system intrusion.

Types of Testing Tools

- Program monitors
  - Permitting full or partial monitoring of program code
  - Ex; Instruction set simulator, Hypervisor, Program animation, etc
- Formatted dump or symbolic debugging
  - Tools allowing inspection of program variables on error or at chosen points
- Automated functional GUI
  - Testing tools are used to repeat system-level tests through the GUI
- Benchmarks Tools
  - Allowing run-time performance comparisons to be made with rivals
- Performance analysis (Profiling tools)
  - Tools that can help to highlight hot spots and resource usage

Software Testing Process



Test Plan

- Used as a guide to test software.
- Uses ‘dummy data’ (sample data)
- Usually prepared before software is built
- Prepared by ‘software testers’

Test Plan - Example

TEST DATE / TIME:		**PROJECT ID:**	
TEST BY:		**SYSTEM TITLE:**	

TEST CASE	TEST TYPE	TEST STEPS	EXPECTED RESULT	ACTUAL RESULT	REMARKS
1. User Sign-in	Unit Test	1. Open application 2. Enter User Name as "X" and …….	User successfully signed-in into the system, welcome page displayed.	Error – does not show error when 'space' is used in ID	Correction made to code to check ID validity.
2. Update User Profile					

Question & Answer

Tutorial 1

- Explain 5 major **problems** that you would face if you **don't** adhere to Good Programming Practices.

- Discuss 5 good **programming practices** when building an application for mobile phone.

Tutorial 2

- Explain 5 advantages and disadvantages of Software Testing.
- Find 3 Testing Tools (software) which can be used to test other software.
- Crate a simple test plan to test 3 functions of Webspace-2

Next Session

- System Deployment