System Testing & Evaluation

PRESENTER: P. ROXANE ARMFIELD MSN RN

CLINICAL ANALYST

NORTHEAST GEORGIA HEALTH SYSTEMS

Learning Objectives – Testing/Evaluation

Define the purpose of information systems testing

Identify five key components of a testing methodology

Understand the major levels of testing and their intended use

List three testing controls used to maintain the integrity of a testing process

Identify three specific areas that are addressed in test reports

Understand the importance of post-implementation evaluations

Purpose of Testing

The purpose of testing is to formally challenge the functioning of a program, application or system - under controlled conditions - specifically to detect errors or unexpected system responses in order to manage risks of developing, producing, operating and sustaining systems.

5 Key Components of Testing Methodologies

Planning – the strategy

Development – the test plan

Execution – running the plan

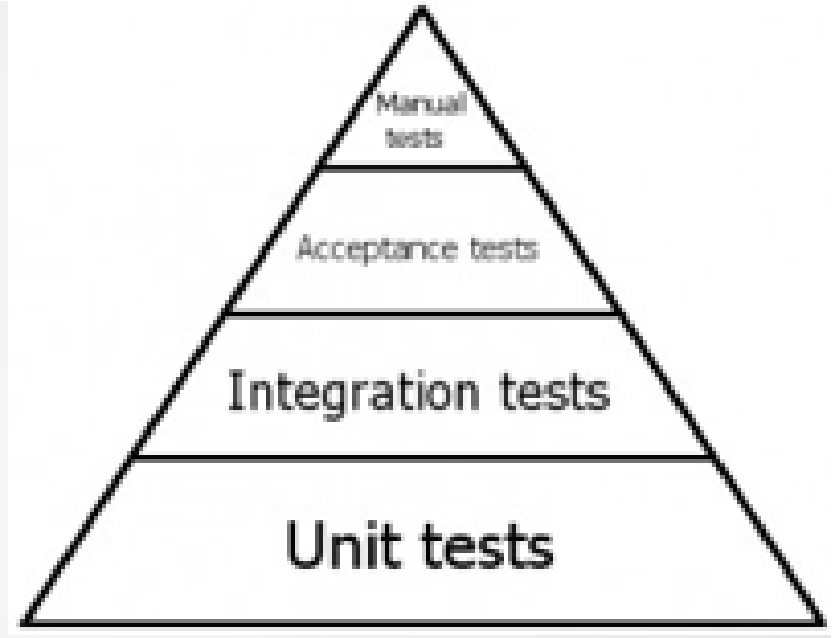
Reporting – includes effectiveness, status and defect reporting and determining if the system or SW is ready for the next level of testing or release

Evaluation – always follows implementation of a new release.

Test Strategy

Common test strategy components include:

- Testing scope and objectives
- Current business issues
- Roles and responsibilities
- Status reporting
- Test methods
- Industry standards
- Test tools
- Measurements and metrics
- Risks and mitigations
- Defect reporting and tracking
- Change management



Planning and Development

Testing Strategy – in Project Management terms – the “project charter” for testing

- Provides a formal description of how the organization plans to approach testing in terms of:
  - Resources
  - Infrastructure
  - Functional relationships
  - Practice standards
  - Defines importance of systems testing within the overall IT strategy
  - Informs key stakeholders of the effort, time and resources required
  - Achieves commitment and approval of the approach
- Software release levels
  - Defines testing events based on SW release levels: major, minor, emergency.
- Levels of testing
  - Unit – one component or module of a system
  - Functional – validates functionality against design specifications
  - Integration and Interoperability – identifies defects in the interfaces and integration points between other applications and systems
  - System – verifies functional integrity of a completely integrated system
  - Performance – measures response time at normal and stress conditions
  - Regression – have new changes undone functionality in previous releases
  - Acceptance – final stage and incorporates end-users acceptance of the system for production use

Planning and Development - 2

Control requirements describe the functional interactions between the (ITIL) configuration, change and release management processes.

- Contains the rules for retesting

Limitations and assumptions

- Part of strategy determining testing priorities and resources

Test Plan – tactical in nature

- How will the testing be done
- What will be tested
- When can it begin
- Who will do it
- How long will it take

Functional test types

- Normal case – expected valid inputs (generally not sufficient)
- Output forcing – all system inputs be selected to force all outputs
- Robustness – tests ability to handle invalid or unexpected inputs
- Combination of inputs – tests multiple functions fully executing business rules

Testing Methodologies: Waterfall

Waterfall Model

The waterfall model is one of the earliest structured models for software development. It consists of the following sequential phases through which the development life cycle progresses:

System feasibility. In this phase, you consider the various aspects of the targeted business process, find out which aspects are worth incorporating into a system, and evaluate various approaches to building the required software.

Requirement analysis. In this phase, you capture software requirements in such a way that they can be translated into actual use cases for the system. The requirements can derive from use cases, performance goals, target deployment, and so on.

System design. In this phase, you identify the interacting components that make up the system. You define the exposed interfaces, the communication between the interfaces, key algorithms used, and the sequence of interaction. An architecture and design review is conducted at the end of this phase to ensure that the design conforms to the previously defined requirements.

Coding and unit testing. In this phase, you write code for the modules that make up the system. You also review the code and individually test the functionality of each module.

Integration and system testing. In this phase, you integrate all of the modules in the system and test them as a single system for all of the use cases, making sure that the modules meet the requirements.

Deployment and maintenance. In this phase, you deploy the software system in the production environment. You then correct any errors that are identified in this phase, and add or modify functionality based on the updated requirements.

Testing Methodologies: Iterative

Incremental or Iterative Development

The incremental, or *iterative*, development model breaks the project into small parts. Each part is subjected to multiple iterations of the waterfall model. At the end of each iteration, a new module is completed or an existing one is improved on, the module is integrated into the structure, and the structure is then tested as a whole.

For example, using the iterative development model, a project can be divided into 12 one- to four-week iterations. The system is tested at the end of each iteration, and the test feedback is immediately incorporated at the end of each test cycle. The time required for successive iterations can be reduced based on the experience gained from past iterations. The system grows by adding new functions during the development portion of each iteration. Each cycle tackles a relatively small set of requirements; therefore, testing evolves as the system evolves. In contrast, in a classic waterfall life cycle, each phase (requirement analysis, system design, and so on) occurs once in the development cycle for the entire set of system requirements.

Testing Methodologies: Agile

Agile Methodology

Most software development life cycle methodologies are either iterative or follow a sequential model (as the waterfall model does). As software development becomes more complex, these models cannot efficiently adapt to the continuous and numerous changes that occur.

Agile methodology was developed to respond to changes quickly and smoothly. Although the iterative methodologies tend to remove the disadvantage of sequential models, they still are based on the traditional waterfall approach. Agile methodology is a collection of values, principles, and practices that incorporates iterative development, test, and feedback into a new style of development.

Test Tools

Manual testing involves direct human interaction with the system under test; manual test tools include:

- Written test plan
- Test script/scenarios to follow
- Method(s) of recording results

Automated testing is the use of testing software to control the text execution and reporting

- Primary automated test tool is the testing software
- Many test automation programs are available in the commercial market
- Most of these test automation programs can be specifically tailored to the organization executing the test

Test Execution

Methods include:

- Unit testing: A development procedure where programmers create tests as they develop software
- Integration testing: Testing where hardware and/or software components are combined and tested to confirm that they work together
- Acceptance testing: A test conducted to determine if the requirements of a specification or contract are met
- Regression testing: Regression testing is any type of software testing that seeks to uncover software errors by partially retesting a modified program

Test Controls

Test controls are implemented to protect a system and its data

Common test controls include

- Version control (Revision control)
  - Tracks and provides control over changes to source code
- Security audits
  - Manual or systematic measurable technical assessments of a system or application
- Change control
  - Formal process used to ensure that changes to a system are introduced in a controlled and coordinated manner

Test Reporting and Final Evaluation

Test Reporting

- Occurs throughout the testing process
- Should be planned out early; can be challenging!
- Reports should include test mission, system(s) under test, organizational risk, test techniques/methods, test environment, status, and any obstacles

Final Evaluation

- Usually the most important deliverable in a test project
- Final evaluation corroborates to the stakeholders that the system has met expected results
- Should address ROI for testing, quality expectations, system readiness, effects of concurrent use, and current risks
- Excellent resource for test team; lessons learned for future improvement

Execution and the Integrity of Testing

Execution of a test plan can be very resource intensive and time consuming. Test cases should mimic high-use and high-risk functional activities

Records of all discrepancies or unexpected results should be maintained and updated after investigation and resolution

Change management processes are especially critical during the testing process

Design changes based on testing should be auditable to system specifications and requirements, and should be reviewed and approved by stakeholders before implementation.

Controls:

- Versioning
- Change
- Quality
- Pre- and post- testing audits
- System access and user security profile integrity and alignment

Post Implementation Review Elements

- How is the system performing?
- How is it being used?
- What opportunities exist for improvement?
- Key outcomes include:
  - Did the release meet its objectives?
  - Did it deliver planned benefits and are the stakeholders satisfied?
  - Did it address contractual and design specifications?
  - Can we improve the implementation process (including testing)?
  - What are the lessons learned?
- Should result in specific action items – communicated to stakeholders – to address areas for further improvement.
- Update testing plans and scripts for use with the next release.

Thank you. Good Luck on your Test!

Questions?

Roxane.Armfield@nghs.com