INFO6255 - Software Quality Control and Management

Medi Servattalab: [M.Servattalab@northeastern.edu](mailto:M.Servattalab@northeastern.edu)

Work at Fresenius Medical Care, Lexington MA for 15 years => VP of Production Support Services

A screenshot of a cell phone

Description automatically generated

# Course Assessment

• Class Participation (10%) –10% makes a difference!!!

• Tests/Quizzes (30 %) – Short quizzes almost every week.

• Projects/Assignments (25%) – UFT/Selenium/Agile assignments for each group.

• Final Exam – Project (35%) – Final project. Your team will do the following:

• Test Condition Matrix/Test Scenarios/Test Cases/Traceability Matrix

• Presentation

• Test Plan

# Moore’s Law

The simplified version of this law states that processor speeds, or overall processing power for computers will double every two years.

**What does all of this really mean?**

1. The Internet is getting bigger and more complicated every day!!
2. Software Applications will become more complicated!!
3. The testing will get more complicated!
4. There will be greater need for Software Developers and Software testers!!!

# Quality Control Management (QCM)

• QCM is the act of overseeing all activities and tasks needed to maintain a desired level of excellence. This process includes implementing a strategy for:

1. Quality Planning
2. Quality Assurance
3. Quality Improvement
4. Quality Control

# Quality Management System’s Basics

• QMS’ Basic Rules:

A screenshot of a cell phone

Description automatically generated1. Document your QMS by stating the processes.

2. Follow the documented processes.

3. Prove that you are following the processes.

• By the form of Audits.

4. Continually review the processes and improve them.

ISO9001 is an example of a Quality Management System

# What is a Quality System?

A quality system is a specific implementation of quality philosophies/concepts, standards, methodologies and tools, for the purpose of achieving quality-related goals

Quality System = Quality Assurance + Quality Control

• **Quality Assurance** aims to prevent defects with a focus on the process used to make the product. It is a proactive quality process.

• **Quality Control** is a set of activities for ensuring quality in the final products.

• Monitor operations to ensure that they meet production standards.

• It is a subset of QA.

• Quality Control is the process that allows firms to ensure the conformity of their products or services. It is used to make sure it meets the: Correct Specifications & Quality Benchmarks

• QC Tools used: Checklists, Fishbone diagrams, Control Charts, Histograms, Andmore...

# IT Compliance

• **IT Compliance** is taking appropriate control of and protecting the information. It includes:

• How the information is **obtained and stored**

• How it is **secured**

• How it is distributed **internally and externally**

• How the **data is protected**

**• A few standards:**

• The Sarbanes -Oxley Act (SOX) of 2002 is a sweeping statute to regulate financial

transparency and reporting.

• HIPAA, or Health Insurance Portability and Accountability Act’s Title II section articulates policies and guidelines for regulating information, especially Protected Health Information (PHI) by insurers, medical providers, and employers who provide health care insurance.

• Basel III applies to the banking industry and helps determine the amount of capital they need to reserve in order to recover in the case of a loss.

• The Payment Card Industry Data Security Standard of 2001 (PCI DSS) is an industry deployed recommendation instituted by MasterCard, Visa, and other credit card companies to provide identity protections for members and service providers.

# What is the focus of this course?

• Software Quality Assurance (SQA) or Software Testing

• Where SQA fits within the SDLC

• Learning the QA Methodologies

• Learning about the QA Phases

• Learning two of the mostly used QA Tools: (UFT and Selenium)

• Learn more about the QA Jobs

# Reviewing the Course Topics

**What is Software testing?**

Software testing is a process, to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not. To identify the defects to ensure that the product is defect free in order to produce a quality product. [But no software is defect free!]

**What does “Analytical Skills” really mean?**

‘Analytical Skill’ is defined as the ability to visualize, articulate, conceptualize or solve both complex and uncomplicated problems by making decisions that are sensible given the available information.

• In Writing: to write test plans and test cases, defects and reports.

• Verbal: to explain the test cases, test reasoning and the issues/problems.

**SDLC – QA Methodologies**

1. Waterfall

• Business Requirement vs. Functional Requirements?

• Test Plan/Test Strategy

• Test Condition Matrix

• Traceability Matrix

• Equivalence Class Partitioning

• Test Coverage

1. Agile:

• Epics, Sprints, Scrums, User Stories, etc.

• Automation Testing

• Performance/Load Testing

• When do you know you are done with testing?

1. Waterfall vs. Agile

• Business Requirements vs Functional Requirements

• How to link the Requirements and the Test Cases and why?

• How to create a traceability matrix to ensure accurate coverage

• How to write test cases

• How to conduct testing in Agile

• What to do as part of the Sprints

• More about Scrums, Sprints, QA role?

# Equivalence Class Partitioning

• It greatly **reduces the number of test cases without compromising the test coverage or quality of testing.**

• It helps in reducing the overall test execution time due to reduced set of test data.

• It is highly used in cases where exhaustive testing is not possible but at the same time test good coverage needs to be maintained.

**• Disadvantages of equivalence class partitioning**

1. The identification of equivalence classes relies **heavily on the expertise** of the tester.
2. Having **incorrectly identified equivalence classes** leads to higher risk of defect leakage and less test coverage.
3. The equivalence classes need to be **partitioned just to the right amount** and groups as having too large partitions leads to risk of missing defects.
4. Whereas, partitioning into more groups of smaller sizes leads to redundant tests.

# Testing Phases & Types

A screenshot of a cell phone

Description automatically generated

**Happy Path Testing:** very simple flow testing (1 path).

**Smoke Testing:** A high level validation of the functionality.

**Equivalence Partitioning:** A technique to determine the minimum set of test cases that must be executed.

**RAD Model:** Rapid Application Development an iterative process.

**Test Stub:** A “dummy” routine to enable a function to be tested.

**Test Harness:** a pool of test stubs and test drivers put together.

**Test Summary Report:** A document showing the test results in a compacted format.

# QA Terminology

• **Test Dashboard/Test Report**

A matrix showing the test execution summary with pass/failure of the tests.

• **SIT**

Systems Integration Testing done before the actual testing begins.

• **Unit Testing**

Testing done by the development to ensure that their code functions as designed.

• **Integration Testing**

Testing multiple components interfacing with each other.

• **Regression Testing**

Repeating the running of the older test cases.

• **Manual Testing**

Running the test cases by humans.

• **Automated Testing**

Running the test cases using automated tools or scripting.

• **Performance Testing**

Running test to ensure that the product can function under a specified load.

• **Load and Stress Testing**

Same as above. Stress testing goes above the Performance testing to find the level of failure.

• **Test Evidence/Test Results**

Showing proof of testing by taking screen shots of the actual results.

• **Defects**

Reporting the failure of the test cases to the development.

• **Defect Severity & Priority**

The place in the queue for fixing and deploying into QA/Production.

• **Defect Process**

Agreed upon process on how to handle the defects Reporting, prioritization and fixing.

• **Builds**

Versioned code ready to be deployed into an environment.

• **Builds management**

Includes tools to deliver the code to QA in a consistent and controlled manner, including versioning.

• **Source Control**

Using various tools to keep track of what goes into a build so a well measured and a solid build is delivered to QA.

• **Dev/QA/Prod Environments**

QA environment is where the testing can be performed. It needs to be controlled from the code perspective and all builds into it are installed via Build Management software.

• **Test Data**

Ensuring the test data exists in the QA database for testing.

• **Change Request**

Change tickets that introduce changes to the frozen functionality.

• **QA Entrance/Exit Criteria**

Conditions that allows the start and finish the testing cycle.

• **QA Testing tools**

Running the testing using an automation tool as a regression test to speed up the process.

• **QA Signoff**

QA end of testing.

• **Sprint**

a small bundle of wish list items and the decisions on how to develop them.

• **Scrum**

is the agile software development method.

• **POC**

Proof of Concept. A prototype.

• **Test Log**

Keeping a log of the test execution during testing.

• **User Story**

A user/customer function that results into a system function (equivalent to a Functional Requirement).

• **Walkthrough**

A review session to walk through a test plan, test cases or performing a Code Walkthrough to go through the code.

# QA Methodologies

• Top 3

1. Waterfall: The Waterfall model is widely considered the oldest of the structured SDLC methodologies. It’s also a very straightforward approach: finish one phase, then move on to the next. No going back. Each stage relies on information from the previous stage and has its own project plan.
2. Agile: The Agile model has been around for about a decade. But lately, it has become a major driving force behind software development in many organizations.
3. DevOps: The DevOps methodology is the newcomer to the SDLC scene. It emerged from two trends:
4. The application of Agile
5. Lean practices to operations work and the general shift in business toward seeing the value of collaboration between development and operations staff at all stages of the SDLC process.

• Other

1. **Lean:** The Lean model for software development is inspired by **lean manufacturing** practices and principles. The seven Lean principles (in this order) are:
2. Eliminate waste
3. Amplify learning
4. Decide as late possible
5. Deliver as fast as possible
6. Empower the team
7. Build integrity in
8. See the whole
9. **Iterative:** The Iterative model is repetition incarnate. Instead of starting with fully known requirements, project teams implement a set of software requirements, then test, evaluate and pinpoint further requirements. A **new version of the software is produced with each phase**, or iteration. Rinse and repeat until the complete system is ready.
10. **Spiral:** One of the most flexible SDLC methodologies, the Spiral model takes a cue from the Iterative model and its repetition; the project passes through four phases (**planning, risk analysis, engineering and evaluation**) over and over in a “spiral” until completed, allowing for multiple rounds of refinement.

# What is a Use Case?

1. Use Case is another way of defining functional requirements.
2. It introduces an **Actor** who will perform various functions in the application.
3. It shows the **interaction of the Actor** with the system or an application.
4. The advantage is that it allows for better **understanding** of the process flow when an Actor is involved.
5. There could be **multiple Actors** describing the Use Cases from different user perspectives.
6. **Use Cases are not User Stories.** User Stories are short descriptions of what the customers do within the application.

# Web Based Testing

**• Web Based Application Testing**

Typical topology: Client -> Web Server -> App Server -> DB

**• Thin-Client vs. Thick-Client testing**

1. Thin-Client: most processing is happening on the Server side
2. Thick-Client: Most processing is happening on the Client side (Java Applets and ActiveX controls on the Client side)

**• Advantages and disadvantages**

1. Thick Client: Performance testing focus
2. Thin Client: Less of a performance issues risks

**• Testing Considerations:**

1. Browser Testing: Test as many as browsers as specified in the support document.
2. Functionality Testing: The functions are working?
3. Integration Testing: Are all interfaces working?
4. Usability and User Interface testing
5. Security Testing: Role based testing
6. Load and performance testing: For ensuring performance meets the SLAs
7. Printing
8. Caching
9. File Upload/Download

# Verification vs. Validation

• **Verification** comes early on in the project to verify the

1. Functional Requirements
2. Code Inspections & Walkthroughs
3. Application Design

• **Validation** comes later in the project during the actual testing. It includes all phases of the testing:

1. Functional Testing
2. Integrations Testing
3. End to End or System Testing
4. UAT

# Testing Related Tools

1. Test Management Tools: QC, ALM, Jira, TestRail
2. Test Automation Tools: UFT, Selenium
3. Load Testing Tools: Performance Center, LoadRunner, Jmeter, Webload
4. Cross Browser Testing Tools: BrowserShots, Saucelabs, Selenium Grid
5. Security Testing Tools: Wfuzz, Grabber, Zed Attack Proxy, SQLMap
6. Cyber Security Testing Tools: Metasploit, Nmap, Wireshark

# Cloud Computing

**• What is the Cloud?**

the practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server or a personal computer”

**• The Cloud Types**

1. Private Cloud – Operated for the organization only.
2. Community Cloud – Shared by specific community
3. Public Cloud – Available to the public
4. Hybrid Cloud – Combining two or more clouds

**• Cloud Testing Types [Testing in the Cloud]**

1. Performance Testing
2. Security Testing
3. Availability Testing
4. External API testing

**A screenshot of a cell phone

Description automatically generated**