**Week 5 Discussion Forum 2 – Project Test Plan**

Vicki Kelm

The University of Arizona Global Campus

CST499: Capstone for Computer Software Technology

Dr. Amjad Alkilani

January 24, 2024

**Week 5 Discussion Forum 2 – Project Test Plan**

The purpose of this paper is to develop a project test plan using the template of the student’s choice. The document will guide the activities during the most important phase of the software development life cycle related to testing. This paper will describe the testing teams, illustrate the exit criteria, explain the test estimated effort, explain the incident reporting strategy, identify how defects are classified, propose management strategies for configuration, and will identify the different tools that will be used throughout the lifecycle of the testing.

1. **Test Management Strategy**
   1. **Test Teams**

“An independent testing team is beneficial for test quality and comprehensiveness” (Spillner, Linz, & Schaefer, p. 32). A tester is able to view the test object without bias. They will need to possess necessary knowledge regarding the test object to create test cases, but will not have any assumptions or misunderstandings that the developer may develop. A developer should not be one to test their own product due to being blind to their own code. “Because there is tendency to be blind to our own errors, it is much more efficient to let different people perform testing and development and to organize testing independently as possible from development” (Spillner, Linz, & Schaefer, p. 169). For this reason, designated testers on the development team should perform component testing. Additionally, “test personnel will consist of three to five IT and test specialists responsible for integration activities, non-functional tests, test automation, and support of test tools (“technical test”)” Spillner, Linz & Schaefer, p. 171). The ultimate leader of the test team will be the test manager who provides the test planning and test control.

* 1. **Test Roles**

The following test roles should be assigned to individuals when conducting testing of the Student Course Enrollment System:

* **Test Manager**: Leader, responsible for test planning and test control experts, possessing knowledge and experience in the fields of software testing, quality management, project management, and personnel management
* **Test Designer**: Test analyst, expert in test methods and test specification, has knowledge and experience in the fields of software testing, software engineering, and formal specification methods.
* **Test Automator**: Test automation expert(s) with knowledge of testing basics, programming experience, and deep knowledge of the testing tools and script languages.
* **Test Administrator**: Expert(s) for installing and operating the test environment (system administrator knowledge).
* **Tester**: Expert(s) for executing tests and reporting failures, IT basics, basic knowledge of testing, using test tools, understanding test object.
  1. **Exit Criteria**

Exit criteria along with test entry “is an important part of test planning” (Spillner, Linz, & Schaefer, p. 179). Both assist with defining the point at which testing will start and stop. The importance of exit criteria is to prevent the test work is not ended prematurely or by chance. Each test case should contain specific exit criteria to guide the testers for determining when a test is complete. Some examples of test criteria and indicators are listed below:

* **Achieved Test Coverage**: Tests run, covered requirements, code coverage
* **Product Quality**: Defect density, defect severity, failure rate, and reliability of the test object
* **Residual Risk**: Tests not executed, defects not repaired, incomplete coverage of requirements or code
* **Economic Restraints**: Allowed cost, project risks, release deadlines, market chances

Any project-specific test exit criteria will be included in the test plan as defined by the Test Manager. The exit criteria will be measured regularly and evaluated during test execution to serve as the decision basis by test and project management.

* 1. **Test Estimated Effort**

The Test Manager will provide in the test plan any details pertaining to estimating test effort and test costs to include; re-estimating, and re-planning the testing tasks during later testing work. It is the Test Manager’s responsibility to initiate test effort estimation during the planning phase to ensure that resources are assigned and distributed properly. “Task-driven test effort estimation tends to underestimate the testing effort. Estimating based on the data of experience of similar projects or typical values usually leads to better results” (Spillner, Linz, & Schaefer, p. 184).

* 1. **Test and Risk**

Test and risk are combined as Risk-Based Testing which uses information on project and product risks and directs testing to areas with high risk. “Risk-based prioritization of the tests ensures that risky product parts are tested more intensively and earlier than parts with lower risk” (Spillner, Linz, & Schaefer, p. 189). This will aid in finding severe problems early on reducing delays or the need for corrective rework. The demand for systemic risk management is included in the standards for quality assurance and test plans that include the following:

* Regularly identifying what can go wrong (risks)
* Prioritizing identified risks
* Implementing actions to mitigate or fight those risks

Testing will decrease any uncertainty about risks, will help to estimate risks, and will ultimately identify new risks.

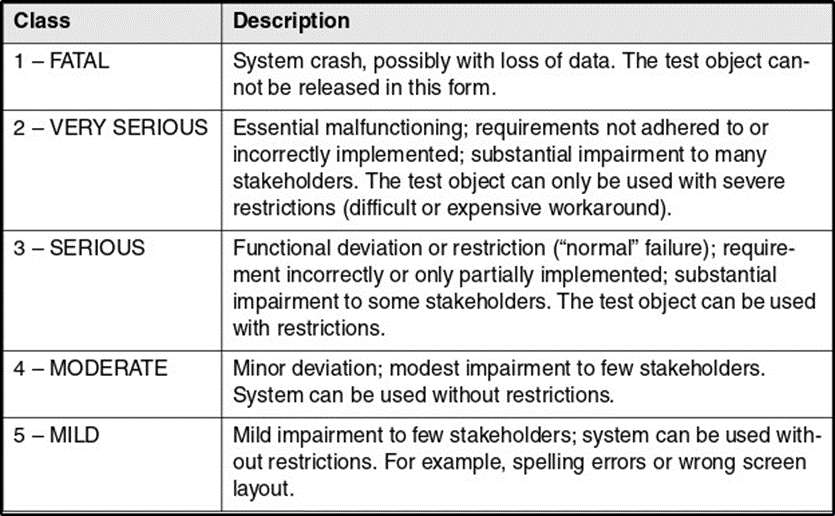
* 1. **Incident Reporting**

A test incident report is a document/report generated after the culmination of the software testing process. The various incidents and defects are reported and logged by the team members to maintain transparency among the team members and to take essential steps to resolve these issues. An incident report is created for the test object that caused the problem and is done for every unexpected behavior or observed deviation from the expected results found in the test log (Spillner, Linz, & Schaefer, p. 193). The incident report is a formal document that describes any unexpected event, issue, or problem that is encountered during the testing process. A central database is created for each project in order to manage each incident that is discovered. Incidents may be reported by testers, Quality Assurance Engineers, or other stakeholders of the project. The report will include information regarding the tested software, test environment, tester’s name, defect classification, defect prioritization, and any relevant information to reproduce or locate the defect.

* 1. **Defect Classification**

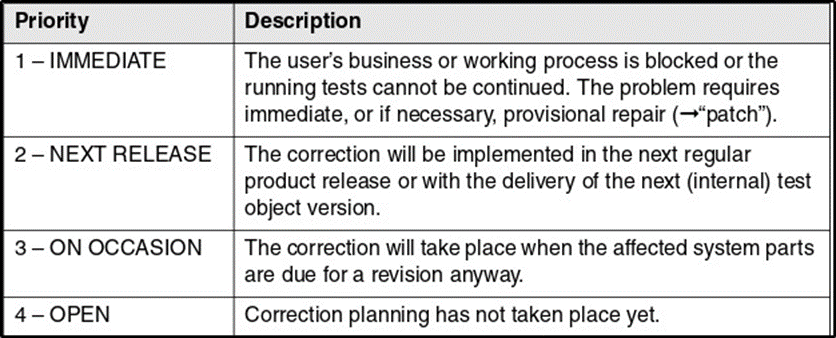
Defect classification, also known as defect severity, “is the classification of a defect based on its level of destructive impact on the requirements specification of the software” (Defect Severity, 2020). The defects, also known as bugs or faults, are anything that causes threading to the software's value, quality, and aim. The tester can identify it through the variance of expected results and actual results of execution of test cases. The classification includes levels of severity class as well as fault priority. There are five class levels of severity: 1 – FATAL, 2 – VERY SERIOUS, 3 – SERIOUS, 4 – MODERATE, 5 – MILD. There are four levels of priority: 1 - IMMEDIATE, 2 – NEXT RELEASE, 3 – ON OCCASION, 4 – OPEN. The diagrams below provide the levels and details of each as provided in the course text:

**Figure 1**

Failure Severity Class Table

**Figure 2**

Fault Priority Table



* 1. **Configuration Management**

In order to have sufficient and successful configuration management, several requirements must be met. These requirements include: Version Management, Configuration Identification, Incident and Change Status Control, and Configuration Audits.

* **Version Management**: Provides cataloguing, filing, and retrieving the different versions of a configuration item, and includes securing comments on the reason for a particular change.
* **Configuration Identification**: The identification and management of all files (configuration objects) in the particular version, comprises a subsystem (configuration) and requires version management as a prerequisite.
* **Incident and Change Status Control**: Documenting of incident reports and change requests and possibilities to reconstruct their application on the configuration objects.
* **Configuration Audits**: Used to check the effectiveness of configuration management, useful to organize configuration audits. Audits offer the possibility to check whether the configuration management documented all software components, whether configurations can be correctly identified, etc.

Tech Target states, “Configuration management heavily depends on policy, process and automation, which must be integrated into the CM tool or platform”. It is important to revisit, audit and test the configuration management policy, and regularly review and update the process and automation elements to ensure that tool and its usage maintain consistency with IT and business goals.

1. **Test Tools**
   1. **Introduction**

There are many tools that provide critical support to the different phases of testing. “Test tools are normally used for these purposes: improving test efficiency, enabling tests, and improving test reliability” (Spillner, Linz, & Schaefer, p. 205). The general categories of tools commonly used in various testing phases are:

* **Test planning tools**: Help testers to define the objectives, scope, strategy, and schedule of the testing project.

Examples: Microsoft Project, Jira, Trello, and TestRail

* **Test design tools**: Help testers to create test cases, test scripts, test data, and test automation scripts based on the requirements and specifications of the software being tested.

Examples: TestComplete, Selenium, Appium, and Postman

* **Test execution tools**: Help testers to run the test cases, test scripts, and test automation scripts on the software under test and collect the test results.

Examples: Perfecto LambdaTest, Jenkins, and Bugzilla

* **Test reporting tools**: Help testers to analyze the test results, generate test reports, and communicate the test findings and recommendations to the stakeholders.

Examples: TestRail, Allure, Kibana, and Power BI

Some key characteristics for test tools that allow for objective evaluation include: cost, usability, functionality, and support. The tool should fit within the budget of the organization and provide a good return on investment. “The cost should include not only the licensing fee, but also the operational, maintenance, and support costs” (Bose, 2022). The tool should be easy to use and learn for the testers, and provide a user-friendly interface and documentation. “The tool should also be compatible with the existing tools and technologies used by the organization” (Software Testing Help, 2023). The tool should support the testing needs and objectives of the organization, and provide the required features and capabilities. “The tool should also be reliable, stable, and scalable, and support various types of testing, such as functional, performance, and security (Software Testing Help, 2023). The tool should have a good customer service and technical support, and provide regular updates and bug fixes.

* 1. **Test Management and Control Tools**

“Test management tools provide mechanisms for easy documentation, prioritization, listing, and maintenance of test cases” (Spillner, Linz, & Schaefer, p. 206). These tools provide the ability to document and evaluate details for if, when, and how often a test case has been executed. Advanced test management tools support requirements-based testing by capturing system requirements and link them to the tests, which tests the corresponding requirements. Some of the leading test management tools include: Tuskr, PractiTest, and Jira.

* Tuskr: Powerful. Aesthetically pleasing and intuitive cloud-based test management tool, allows for easy organization of test cases into projects, suites, and sections.
* PractiTest: End to end test management tool, enables full visibility into testing process, provides broader understanding of testing results, fully customizable and flexible.
* Jira: Performs automatic build, test, and release in a single place, supports many technologies and languages.
  1. **Test Specification Tools**

Tools for test specification allow for test cases to be reproducible, the pre- and postconditions as well as test input data and expected results need to be specified. Some tools used for test specification are: QACoverage, TestCaseLab, and Test Collab.

* QACoverage: Test design module to create manual tests cases, monitor completeness of requirements traceability, test case execution.
* TestCaseLab: Test cases, test plans, test runs, integrations, intuitive to use, create unlimited projects and users.
* Test Collab: Test cases categorized/managed in one place, test execution report, custom reports, integrated bug tracker, requirements linked with test cases.
  1. **Static Testing Tools**

Before there are executable programs, static analysis can be executed on source code or on specifications before there are executable programs. “Tools for static testing can therefore be helpful to find faults in early phases of the development cycle” (Spiller, Linz, & Schaefer, p. 210). Some tools for static testing include: Checkstyle, Soot, and SourceMeter.

* Checkstyle: Helps to write Java code, automates process of checking Java code, finds class/method design problems, ability to check code layout and formatting issues.
* Soot: Java optimization framework, provides four representations for analyzing and transforming Java bytecode.
* SourceMeter: Built for precise static source code analysis of C/C++, Java, C#, Python, and RPG projects, finds weak spots of a system under development from source code.
  1. **Dynamic Testing Tools**

Dynamic testing tools reduce the mechanical work involved in test execution. “These tools send input data to the test object, record its reaction, and document test execution” (Spillner, Linz, & Schaefer, p. 211). Some tools include: Selenium, JUnit, and Daikon.

* Selenium: An open-source test automation framework that supports multiple programming languages and browsers.
* JUnit: A unit testing framework for Java programming language that allows developers to write and run tests.
* Daikon: A dynamic testing/analysis tool that helps to identify and remediate bugs in Java, C, and C++ programs.
  1. **Non-Functional Testing Tools**

Tools for nonfunctional tests provide support for load and performance tests. Some tools for non-functional testing are listed below:

* **Load Testing Tools**: Tools such as Apache JMeter, Gatling, and LoadRunner help simulate high user loads and measure system performance under heavy traffic.
* **Security Testing Tools**: Security tools like OWASP ZAP, Burp Suite, and Nessus aid in identifying vulnerabilities and weaknesses in the system’s security.
* **Performance Monitoring Tools**: Performance monitoring tools like AppDynamics, New Relic, and Dynatrace help monitor the performance of applications and infrastructure.
* **Usability Testing Tools**: Usability testing tools like UserTesting, UserZoom, and Optimizely help test the usability of applications and websites.
* **Test Management Tools**: Test management tools like TestRail, Zephyr, and qTest help manage test cases, test plans, and test runs.
* **Compliance Testing Tools**: Compliance testing tools like Compliance Sheriff, Siteimprove, and AChecker help ensure that web content meets accessibility standards.

“Testing is a complex activity that involves many activities and thus must be planned” (Tsui, Karam, & Bernal, p. 212). This paper described the testing teams, illustrated the exit criteria, explained the test estimated effort, explained the incident reporting strategy, identified how defects are classified, proposed management strategies for configuration, and identified the different tools that will be used throughout the lifecycle of the testing.

**References**

Bose, S. (2022, June 2). *What to consider when evaluating a Test Automation Tool: Checklist*. <https://www.browserstack.com/guide/test-automation-tool-evaluation-checklist>

GeeksforGeeks. (2020, June 04). *Defect Severity*. <https://www.geeksforgeeks.org/defect-severity/>

GeeksforGeeks. (2023, July 19). *Software Engineering | Black box testing*. <https://www.geeksforgeeks.org/software-engineering-black-box-testing/>

Software Testing Help. (2023, June 22). *How To Choose The Best Automation Testing Tool (A Complete Guide)*. <https://www.softwaretestinghelp.com/automation-testing-tutorial-4/>

Spillner, A., Linz, T., & Schaefer, H. (2014). *Software Testing Foundations: A Study Guide for the Certified Tester Exam (4th ed.)*. <https://ashford.redshelf.com/app/ecom/shelf/course-section/6094369>

Tsui, F., Karam, O., & Bernal, B. (2018). *Essentials of Software Engineering* (4th ed.). Jones & Bartlett Learning. <https://platform.virdocs.com/read/2348054/11/#/4/2[ch06]/2/2,/3:0,/3:0>