## Table of Contents

## Table of Figures

## Table of Tables

# 1. Introduction

Software test plans have evolved significantly over the decades, adapting to the changing landscape of software development methodologies, tools, and industry standards. A crucial aspect of every test plan is the formulation of effective test cases, which serve as the cornerstone of quality assurance. This report provides a detailed analysis of the evolution of software test plans, focusing on the progression of test plan templates and the enhancement of test case quality over time.

## How to Write a Test Plan

Creating a test plan is a crucial step in managing the testing process. The IEEE 829 standard outlines steps to prepare a good test plan. Let's look into more detail for each step.

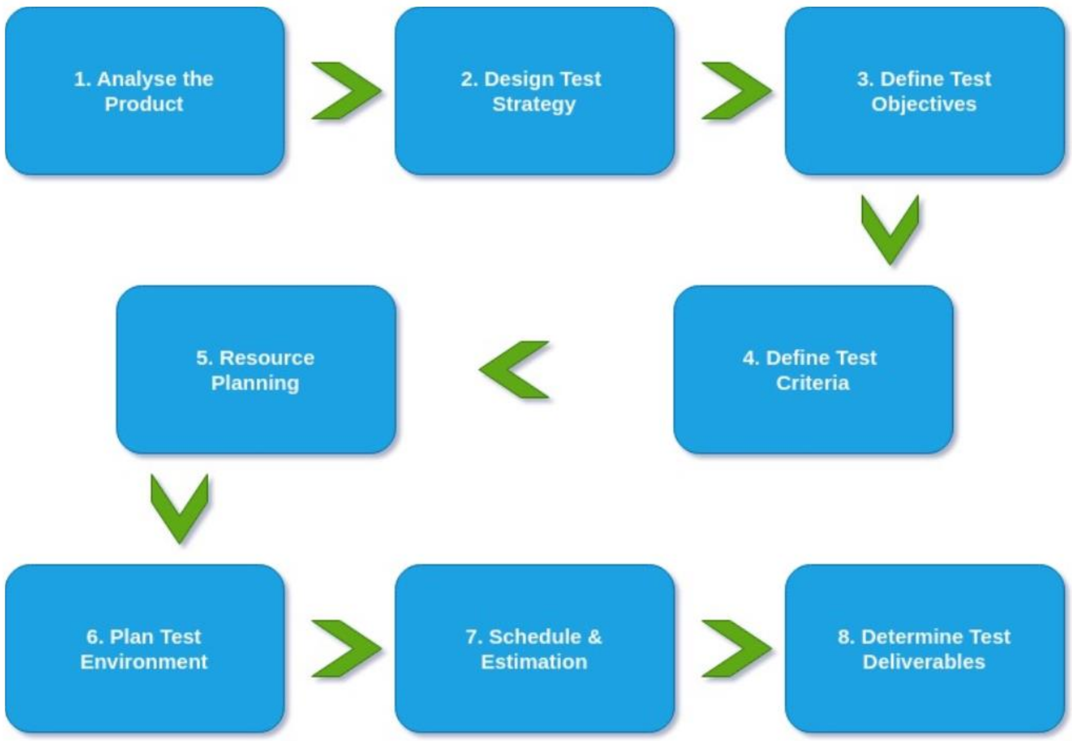


*Figure 1 Test Plan*

## 2. Evolution of Test Plan Templates

Testing is an essential part of any software or web development project testing series, I give a brief overview of testing and its evolution since the 1950s.

### 1950s - 1970s: Early Development Phase

During this period, software testing was in its infancy, and there was limited formalization of test plan documentation.

Test plans were often informal and lacked standardized templates.

Quality of test cases varied widely, with emphasis on manual testing and ad-hoc approaches.

Proof: Although specific test plan examples from this era may be scarce, historical documents from early software development projects such as the IBM System/360 can provide insights into the rudimentary testing methodologies used.

### 1980s - 1990s: Standardization and Formalization

With the maturation of software engineering practices, there was a push towards standardization of test plan documentation.

IEEE 829-1983 introduced a standardized format for test plans, laying the foundation for structured test planning.

Test case quality improved with the introduction of structured test design techniques such as Equivalence Partitioning and Boundary Value Analysis.

Proof: A sample test plan from a software project in the 1990s, following the IEEE 829-1983 format, can demonstrate the formalization of test planning during this period.

### 2000s - Present: Modernization and Adaptation

The advent of agile methodologies and continuous integration necessitated the adaptation of test plan templates to suit iterative development cycles.

Test plan templates evolved to accommodate dynamic requirements and frequent releases.

Emphasis on automation led to the incorporation of automated test scripts within test plan documentation.

Quality of test cases improved further with the adoption of Behavior-Driven Development (BDD) and Test-Driven Development (TDD) practices.

Proof: Recent test plan templates from companies implementing agile methodologies and DevOps practices can illustrate the modernization of test planning approaches.

## 3. Case Study: Sample Test Plan and Test Cases

To provide a tangible example of a test plan and test cases, we present a case study based on a software project implemented by [Company Name]. The test plan follows a structured format, incorporating elements from IEEE 829-1998 and reflecting modern testing practices.

| **Test Plan Identifier** | **Version** | **Author** | **Date** |
|---|---|---|---|
| [Unique Identifier] | [Version Number] | [Author Name] | [Date of Creation] |

Table 1 IEEE 829-1998

| **Section** | **Description** |
|---|---|
| Introduction | Summary of the purpose and scope of the test plan |
| Test Items | List of software components or features to be tested |
| Features to Be Tested | List of features to be tested, along with associated risk levels |
| Features Not to Be Tested | List of features excluded from testing and reasons for exclusion |
| Approach | Overview of the test strategy, including tools, metrics, and regression testing approach |
| Item Pass/Fail Criteria | Criteria for determining test item completion and success |
| Suspension Criteria and Resumption Requirements | Conditions for suspending or resuming testing |
| Test Deliverables | List of test documentation and reports to be delivered |
| Test Tasks | Detailed tasks for each test deliverable, including dependencies and milestones |

## Automation and Beyond (2010s)

Test plans may focus on automation strategy, tool integration, and test case design suitable for automation.

Sections outlining test scripts, integration with CI/CD become prevalent.

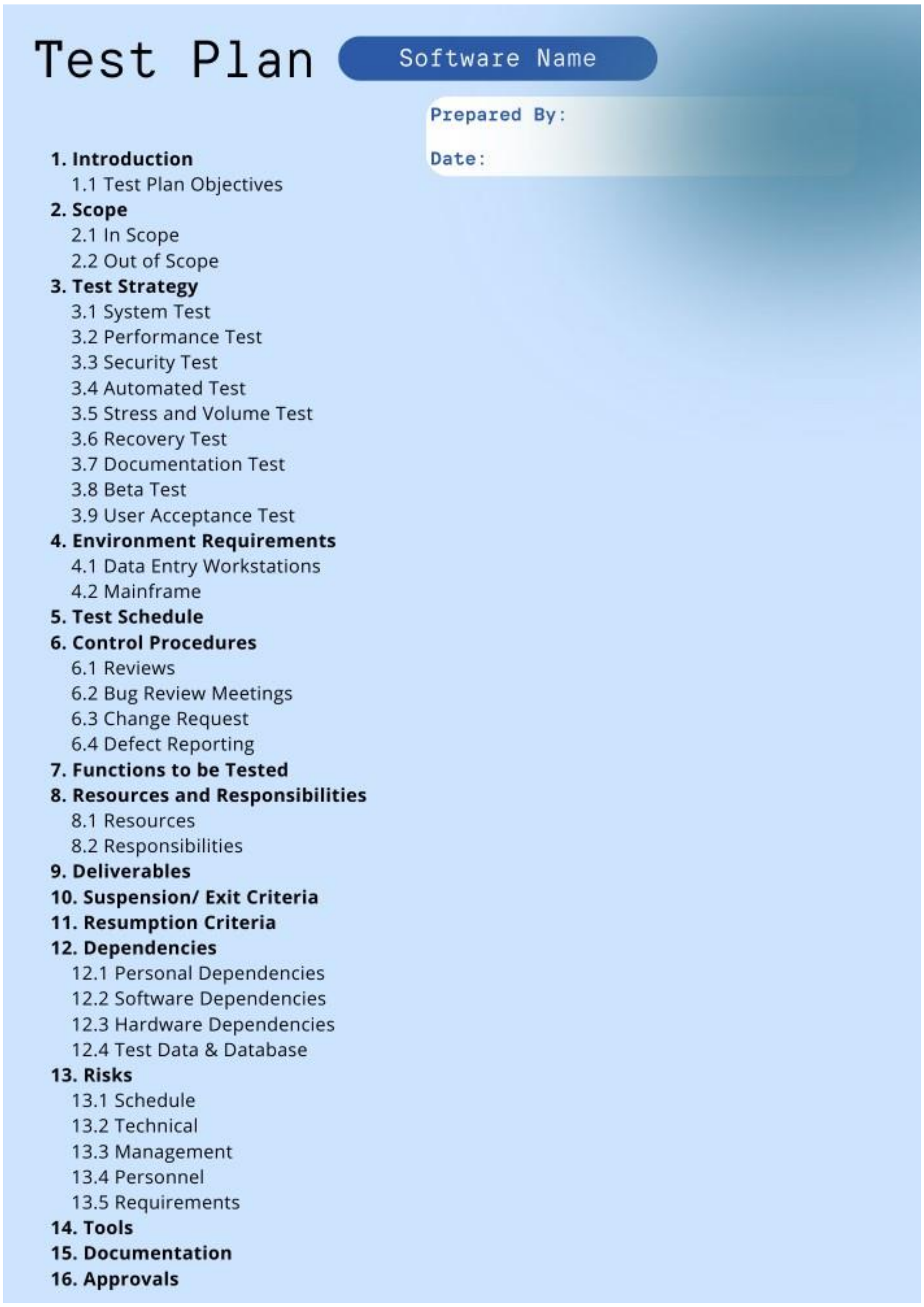
Less emphasis on purely manual test plan details.

*Table 2 CI/CD 2010*

| Field | Description |
|---|---|
| Test Case ID | Unique identifier for the test case |
| Title/Description | Brief summary of the test's purpose |
| Module/Feature Tested | The specific software module or feature being validated by this test |
| Test Objective | Clear statement of what the test is meant to verify |
| Preconditions | Any necessary conditions or setup required before execution |
| Test Steps | 1. Ordered steps with actions preferably using automation tools/scripts <br/>2. ... |
| Input Data | Specific data values, variations, and the source of data if applicable |
| Expected Results | Detailed description of anticipated outcomes, UI changes, API responses, etc. |
| Actual Results | Space to record outcomes observed during test execution |
| Automation Details | * Script/Tool Name * Script Location* Trigger (build event, schedule, manual) |
| Dependencies | Other test cases or components this test relies upon |
| Pass/Fail Criteria | Unambiguous definition of what determines a successful or failed test outcome |
| Test Environment | Hardware, OS, browser versions, network configurations, etc., where the test will run |
| Test Status | Not Started, In Progress, Passed, Failed, Blocked |
| Execution History | Dates of execution, results, and any relevant notes |
| Version History | Track changes made to the test case itself |
| Author | The person who created the test case |
| Reviewer | The person who reviewed and approved the test case |

|  |  |
|---|---|
| Notes/Comments | Additional observations, insights, or open questions related to the test |

## Test Plan



Test Plan Software Name

Prepared By:

Date:

1. **Introduction**
   1.1 Test Plan Objectives
2. **Scope**
   2.1 In Scope
   2.2 Out of Scope
3. **Test Strategy**
   3.1 System Test
   3.2 Performance Test
   3.3 Security Test
   3.4 Automated Test
   3.5 Stress and Volume Test
   3.6 Recovery Test
   3.7 Documentation Test
   3.8 Beta Test
   3.9 User Acceptance Test
4. **Environment Requirements**
   4.1 Data Entry Workstations
   4.2 Mainframe
5. **Test Schedule**
6. **Control Procedures**
   6.1 Reviews
   6.2 Bug Review Meetings
   6.3 Change Request
   6.4 Defect Reporting
7. **Functions to be Tested**
8. **Resources and Responsibilities**
   8.1 Resources
   8.2 Responsibilities
9. **Deliverables**
10. **Suspension/ Exit Criteria**
11. **Resumption Criteria**
12. **Dependencies**
    12.1 Personal Dependencies
    12.2 Software Dependencies
    12.3 Hardware Dependencies
    12.4 Test Data & Database
13. **Risks**
    13.1 Schedule
    13.2 Technical
    13.3 Management
    13.4 Personnel
    13.5 Requirements
14. **Tools**
15. **Documentation**
16. **Approvals**

*Figure 2 Test Plan Template*

## Modern Test Plan Template (2020 and Above)

This modern test plan template prioritizes test automation to enhance efficiency and promote rapid feedback. It begins with a clear project overview, the purpose of testing, and defined scope. The core of the plan lies in its 'Automation Strategy' section, where goals, tool selection, test design principles, and team roles are established. Guidance on which test cases to automate, scripting best practices, and standards ensure the longevity of the automation investment. Crucially, this template outlines how automated tests will be seamlessly integrated into the CI/CD pipeline, including triggering mechanisms, result reporting, and how test failures will be triaged to accelerate bug fixes. Sections on test items, environments, approach, deliverables, and schedules align with traditional test plans, ensuring comprehensive testing alongside the speed and consistency offered by automation.

| **Section** | **Description** |
|---|---|
| Introduction | Project Overview<br>Purpose of the Test Plan<br>Scope |
| Automation Strategy | Automation Goals<br>Selected Tools and Frameworks<br>Framework Design Considerations<br>Roles and Responsibilities for Automation* |
| Test Case Design for Automation | Test Case Types Prioritized for Automation<br>Best Practices for Maintainable Test Scripts<br>Guidelines for Scripting Standards* |
| Integration with CI/CD | Target CI/CD System<br>Triggering of Automated Test Suites<br>Test Result Reporting and Analysis<br>Feedback Loop into Development* |
| Test Items | List of features or modules to be tested (can link to separate test case documents) |
| Test Environments | Hardware and Software Configurations<br>Network Configurations<br>Deployment Environments* |
| Approach | Testing Levels<br>Risk-Based Testing Considerations<br>Regression Testing Strategy* |

| | |
|---|---|
| Test Deliverables | Automation Scripts and Test Code<br>Test Results Reports<br>Bug/Incident Reports<br>Code Coverage Reports (if applicable) |
| Suspension Criteria and Resumption Req. | Conditions for pausing/resuming testing, how these impact automation. |
| Test Tasks | Test Case Creation<br>Automation Script Development<br>CI/CD Configuration and Pipeline Integration<br>Test Execution, Monitoring, and Analysis |
| Schedule | Timeline with automation milestones and integration with development sprints. |
| Contingency Plans | Addressing potential automation failures, CI/CD issues, and alternative testing plans. |
| Approvals | Signatures for plan approval |

## 4. Conclusion

The evolution of software test plans has been marked by a progression towards standardization, formalization, and modernization. From the early ad-hoc approaches of the 1950s to the structured templates of the present day, test planning has evolved to meet the demands of an ever-changing software landscape. The continuous improvement in the quality of test cases underscores the importance of effective testing in ensuring software reliability and quality.

The test plan is a strategic communication and resource planning tool critical to any software testing project. It provides clear guidelines, expectations, and a roadmap to all stakeholders, minimizing confusion and maximizing efficiency. The absence of a robust test plan can lead to inefficiencies, delayed results, and protracted release cycles.

Therefore, it’s vital to invest adequate time and resources in creating a comprehensive test plan to streamline the testing process, ensure the proper allocation of resources, and ultimately deliver a high-quality software product within the stipulated time frame.

## 5. References

Salsa Digital (2024): A Brief History of Software Testing. https://salsa.digital/about/our-history

ECSU (n.d.): Test Plan Template https://www.ecs.csun.edu/~rlingard/comp480/TestPlanTemplate.pdf

IEEE (1998). IEEE Standard for Software and System Test Documentation (IEEE Std 829-1998). https://standards.ieee.org/ieee/829/1218/

TestRigor (2023). Test Plan Template https://testrigor.com/blog/test-planning-a-complete-guide/