QUALITY ASSURANCE & TETST- STANDARDS & PRACTICES

Test Strategy(9AAD134969)

9AAD134969

Department	GF-IS ADM Applications Performance & Excellence (APE)
Approver	Giulio Bitella, Global Department Manager
Owner	Anna Pietras, Quality Assurance Lead (QA)

For the latest distributable version of this and other Quality Assurance& Test standards please visit this link to ABB Library.

WHAT IS THIS?

This document describes the test strategy for development and maintenance of software applications supported by ABB software development units or delivered by External Vendors. The test strategy lays down the overall approach to verification and validation to the extent it can be generalized for projects.
The testing process follows from the strategy approach: it defines phases of testing, relationships between them and their entry and exit criteria.
Intended readers of this document are members of any software development units mainly Project Managers, Release Managers, Application Mangers, Service Managers, Developers and Testers. There are two usage scenarios for this document: (1) as a statement of direction, to drive changes in the way testing is done and (2) to guide projects in planning of their testing activities.

An important part of the strategy is to describe how software product quality risks are mitigated, which test cases are performed in the different test phases, and the distribution of responsibility for the testing.

TABLE OF CONTENTS

Table of Contents

OVERVIEW

Assumptions

- For application delivered by Vendor – Vendor must appoint Quality Assurance Lead who will be speaking partner to ABB regarding overall quality of delivered software/application.
- For application delivered by Vendor - ABB reserves every right to control the quality of the Intellectual Property (IP) Vendor delivers as a result of agreement made, source code is the primary medium of software IP.
  Note : for application from the box and customize it should be defined in Test Plan.
- For all type of projects: delivered by Vendor or developed in ABB - ABB requires to create and fallow Test Plan.
- The Application Documentation like technical and functional (requirements) documentation, user stories will be completed prior to the ST, SIT ,UAT activities.
- Any changes to requirements after Application Documentation need to be reviewed and agreed to would be discussed as part of functional backlog prioritization process and/or the Change Control Process.
- Test Scripts will be created based on user stories or detailed requirements.
- Each test phase is successfully completed as per the exit criteria or agreed to by all relevant stakeholders before advancing to the next phase.
- Test environments do not experience any significant downtime.
- Code is delivered as per agreed upon time and schedule.
- All above assumptions should be documented in a test Plan that should be officially review during Gate 3 project review

Risks

- Unavailability of the environment, wrong configuration or too small number of available environments may cause downtime or delays in testing.
- Unavailability of the tools or their wrong configuration may cause downtime or delays in testing.
- Lack of requirements, their unclearness or lack of possibility to contact analyst may cause downtime or delays in testing.
- Lack or incompleteness of documentation or its inaccessibility may cause downtime or delays in testing.
- Lack of details required for test data preparation may cause downtime or delays in testing.

TEST STRATEGY

Below is a description of the individual test phases which can be implemented on the projects. The selection of these phases will depend on projects complexity , size and business priorities. It must be defined per each project in Test Plan (both for Waterfall and Agile projects – in Scrum in particular defined test phases should be part of Definition of Done).

Continuous Testing using CI

Continuous Integration (CI) is a software development practice that facilitates immediate feedback on the quality of code.

In the Testing Approach , Continuous Integration is meant foremost as a paradigm of applying a set of automated testing phases as frequently as possible during the coding stage, with the aim of

- detecting defects as early as possible
- minimizing the impact of a defect made by a developer on the rest of his team via immediate feedback on the quality of code and triggering corrective actions
- minimizing the cost of diagnosing and correcting a problem
- keeping the application under development stable

An exact sequence of builds, checks and deployments depends on several factors, including:

- inherent complexity of the application under test, e.g., dependency on other applications, test environment needs
- technology in which the application is being developed and the toolset used for development
- number tests cases and time needed for their execution

Test Automation

Automated tests are indispensable part of Continuous Testing approach. They represent a powerful regression test suite and are essential for early error detection. To maximize return on the investment in test automation, testing approach will subscribes to the Test Pyramid concept as an example of long term approach, **in which the unit tests form the vast majority of automated tests**. They exercise thoroughly the code of the application and can be maintained by developers on equal rights with the code that they test. The medium layer of the pyramid is made of tests that exercise the “services” - functionality of the whole application or its component that can be accessed via headless (non-UI) interface. The UI-based tests exercise end-to-end functionality of the application. They are at the top of the test pyramid, i.e., with the least focus on automation because they are less thorough, more expensive to maintain and slower than service tests.

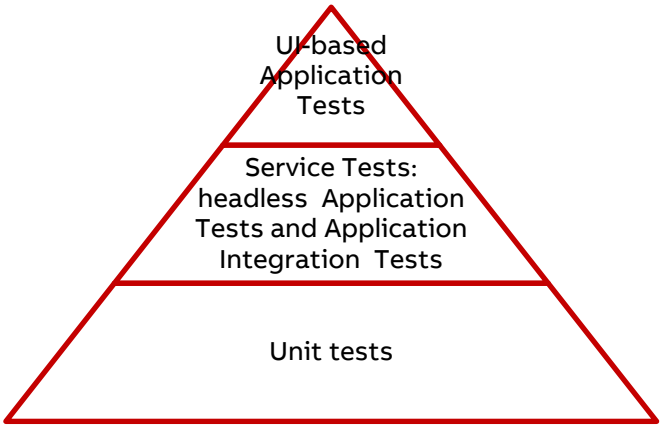

Figure 4. Long-term approach to test automation

The above test pyramid describes an ideal case; an application team may decide to put more focus UI test automation to compensate for a bad design or missing unit tests in legacy code. While writing and updating unit tests can be incorporated into a daily development routine of developers(be part of Definition of Done for Scrum projects), securing resources for test maintenance is prerequisite for starting automation of service and UI tests**. The details of Test automation should be agree and defined in Test Plan for particular project. All type of automated test must be written with codding standards. Tests automated scripts must be updated when functionality is changed.**

Risk-based approach to testing

Since there might be significant number of test cases and pressure of testing schedule a tester must choose a subset of possible tests to execute.
Risk-based strategy involves the identification of areas that can be impacted by introduced changes (impact analysis) and assessment of risk of failures.
Based on the analysis results IS QA Lead should decide on the extent and depth of testing as well as order of test execution.
Risk-based approach to testing mandates that:

- The list of quality risks should be identified as part of Test Plan and maintained throughout the project execution
- Tests should be selected and prioritized for execution to mitigate the risks. Highest quality risks should be addressed as early as possible in the project. The extent of the risk determines the thoroughness of the test.
- When necessary, new regression test cases should be developed
- The test coverage should be monitored throughout the project execution

Regression testing

Regression testing is performed when the previously tested software or its environment is changed. It is done to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as a result of the changes made. Verifying that the bugs are fixed and the newly added features have not created in problem in previous working version of software. For a software under tests, all existing test cases form a pool from which a subset of tests can be selected for regression testing. A regression test suite varies, depending on the outcome of impact analysis

for the change, effort needed to execute specific test cases and the acceptable risk. Test selection criteria should be define in Regression strategy in Test Plan.

a) Manual tests regression

Testers perform functional testing when new build is available for verification. The intend of this test is to verify the changes made in the existing functionality and newly added functionality. When this test is done tester should verify if the existing functionality is working as expected and new changes have not introduced any defect in functionality that was working before this change. Regression test should be the part of release cycle and must be considered in test estimation. Regression testing is usually performed after verification of changes or new functionality. But this is not the case always. For the release taking months to complete, regression tests must be incorporated in the daily test cycle. For weekly releases regression tests can be performed when functional testing is over for the changes.

b) Automated Regression tests

Automated Regression Testing is the testing area where we can automate most of the testing efforts. We run all the previously executed test cases on new build. This means we have test case set available and running these test cases manually is time consuming. We know the expected results so automating these test cases is time saving and efficient regression test method.
Choosing test cases for automation should be done carefully based on defined by project/test team criteria described in Test Plan (for example: time consuming factor , big number of dependencies and combination , stability)
ABB recommended tool for Automated Regression tests (functional) is Selenium Web Driver or HP UTF or Katalon

TEST PHASES

This section describes in more detail Test Phases that were outlined in the previous sections. Before reading these descriptions, one should note of the following explanations regarding specific points in the tabular descriptions below:

Accountable: The one role in a project ultimately answerable for the correct and thorough completion of a testing phase. The accountable signs off (approves) on testing that *Responsible* does and it is the main contact for the Release Manager to provide a plan and report on progress and results. It is assumed that Accountable has support of the Service Level Manager and Release Manager in her/his duties, e.g. time is allocated in the project plan to execute the test phase.

Responsible: The core roles to carry out testing activities in a particular testing stage. Other roles can be delegated to assist in the work required. The Accountable one can be also the Responsible one.

Requirement REVIEW

Test Phase Purpose	• Ensure the completeness and adequacy of the requirements • Ensure traceability to higher level requirements • Ensure testability • Identify and resolve issues • Prevent misunderstanding before the requirements are translated into the product
Test Basis	• List of functional and non-functional requirements and their traceability to higher level requirements • Existing system requirements • Standards, regulations, rules, plans, and procedures • Requirements quality evaluation checklist (optional)
Test Items	• SRS with attached functional and non-functional requirements • Initial Stakeholder Requests and Traceability Matrix
Scope	• All new and changed functional and non-functional requirements in the context of the existing system requirements and initial stakeholder requests
Entry Criteria	• System Requirements Specification is documented and ready for review • Traceability Matrix is available
Exit Criteria	• All problems found in review are adequately addressed • Approval for SRS is granted • Review outputs/summary with a list of approved requirements and review decision provided to all meeting participants and other interested parties.
Test Methods	• Technical review with development and test teams • Review meeting with stakeholders' involvement, possibly with checklist • Impact analysis • Simulations, prototypes, demonstrations or other method defined in the project Plans
Accountable	Project Manager/Service Manager
Responsible	• Requirements Providers/Stakeholder • Release Manager • Functional Analyst • IS QA Lead or dedicated Tester
Related documents	•

Design review

Test Phase Purpose	• Ensure that the design of a software product, its component subsystem or its software and hardware environment correctly implements the driving requirements. • Ensure that a change to the design will not have an adverse effect of fulfillment of other functional and non-functional requirements. • A change to the design includes also selection of a third party component or development framework. • A design review can cover multiple quality characteristics or be focused on one of them, e.g., performance, security, usability. • Design reviews facilitate early detection of design defects and provide an additional benefit of spreading knowledge about software product amongst new project members
Test Basis	• SRS (Software requirements specification) • , design documents, interface documents, design review checklist (optional), static analysis metrics (optional)
Test Items	• Design document and/or working application • Design Review can happen at various points in the project lifecycle and be concerned with new or implemented design
Scope	• For projects following the Gate Model, IS design artifacts mandatory for Gate 3, • Design changes selected by the Solution Architect for review to address a risk, based on the list of identified quality risks in Test Plan
Entry Criteria	• A design document /software ready for review • Static analysis metrics available, in case they are necessary to assess specific quality characteristics targeted by a review, e.g. maintainability
Exit Criteria	• All problems found in review are addressed or fixed, • Review record available, • Approval for Design document is granted (for Gate 3)
Test Methods	Technical review with optional usage of a checklist Reviews may include simulations, prototypes, demonstrations or other method defined in a project plans.
Accountable	Solution Architect
Responsible	Solution Architect
Comments	For application delivered by Vendor – Technical Lead from Vendor side must cooperate with Solution Architect on Design Review actions.

Static analysis

Test Phase Purpose	• Automatically detect code defects such as un-initialized variables, indexing beyond array bounds, memory leaks, and security vulnerabilities • Automatically detect coding guidelines violations • Automatically measure code complexity and duplication
Test Basis	• Defined application-specific set of checks: static analysis tool configuration settings – standard and custom rules • Coding guidelines
Test Items	Source code
Scope	Whole source code
Entry Criteria	• Static Analysis rules are approved by the Technical Lead • Compiled, locally built source code committed by a developer to an integration build. • Defined thresholds/quality goals
Exit Criteria	• For new and changed code, all new alerts shall be analyzed and resolved • Metrics/reports are available
Regression Test Approach	Static analysis is run systematically on the whole source code following an integration build of the application
Test Automation Approach	Static analysis tools selected by the application team. Static analysis is automatically triggered by integration builds, e.g., via CI Server and fails the build in case of errors
Test Methods	Subset of checks offered by the selected tool and additional custom rules, as defined in the Test Plan
Accountable	Solution Architect
Responsible	Developers (designated developer responsible for the tool setup and rules maintenance)
Comments	• **ABB recommended tool is SonarQube** • **ABB reserves every right to control the quality of the Intellectual Property (IP) Vendor delivers as a result of agreement made, source code is the primary medium of software IP.**

Code review

Test Phase Purpose	The purpose is to find bad programming practices, not reusing utility code, possible concurrency problems. Additional benefit of code reviews is not only to find defects but also to spread knowledge amongst new project members.
Test Basis	Code review checklist customized by an application team
Test Items	Source code
Scope	All new and changed code
Entry Criteria	• Code review checklist approved by the Solution Architect • Compiled, locally built source code committed by a developer to an integration build. Static Analysis and Unit Tests completed. • Coding standards
Exit Criteria	All issues resolved Review records available (for Formal Inspections)
Test Automation Approach	Optional use of a code review tool for asynchronous, geographically distributed peer code reviews. Support for a code-review step in Change Request workflow item tracking.
Test Methods	• Pair-programming • Walkthrough • Formal inspection
Accountable	Solution Architect
Responsible	Solution Architects ,Developers/Development Team
Comments	Code review should be part of development culture on daily basis . In agile project Code Review is included in Definition of Done. **For application delivered by Vendor- ABB reserves every right to control the quality of the Intellectual Property (IP) Vendor delivers as a result of agreement made, source code is the primary medium of software IP. Note : for application from the box and customize it should be defined in Test Plan**

Unit testing

Test Phase Purpose	The purpose is to detect implementation errors in units of code, e.g., a method or stored procedure
Test Basis	Formal (e.g. a design document) or informal specification of the code
Test Items	Source code

Scope	All new and changed code in case the application design permits. An existing module could be re-factored to facilitate unit tests. The Technical Lead decides the scope and coverage targets for unit tests in an application and documents it in Test Plan
Entry Criteria	Compiled, locally built source code committed by a developer to an integration build
Exit Criteria	Unit test coverage satisfies the target level. A tests run detects no errors
Regression Test Approach	The whole set of unit tests runs systematically on the whole source code following an integration build of the application.
Test Automation Approach	• Fully automated using a unit test frameworks and mocking libraries selected by the application team • Unit testing is automatically triggered by integration builds, e.g., via CI Server and fails the build in case of errors or low coverage
Test Methods	• Control flow and data coverage • Test Driven Development
Accountable	Solution Architect
Responsible	Developers/Testers
Related documents	
Comments	**ABB requires minimum 70% of code covered by unit tests.**

Component Integration testing

Test Phase Purpose	The purpose is to detect defects in the integration of components into the application: programming-related (incorrect interactions among components) and non-programming-related (due to incompatibility of components)
Test Basis	• Software design documents • Software architecture (inferred from the working code)
Test Items	Application or a subset of its components
Scope	New and changed component interface invocations per architecture partitioning defined by the Solution Architect
Entry Criteria	• Changes to component interactions identified • Formal integration built
Exit Criteria	Changes to interfaces and interface invocations covered by tests. A tests run detects no errors.
Regression Test Approach	The whole set of integration tests runs systematically on the whole source code following an integration build of the application.
Test Automation Approach	• Fully automated using a unit test frameworks selected by the application team

	• Integration testing is automatically triggered by integration builds, e.g., via CI Server and fails the build in case of errors
Test Methods	Grey-box methods: Communications Behavior Testing, Protocol Testing
Accountable	Solution Architect
Responsible	Developers
Related documents	
Comments	In the specified approach, Developer prepares simultaneously Unit and Integration Tests for the code he changes according to the rules set in the Test Plan

Functional and non- functional testing

Test Phase Purpose	The purpose is to assure that functional and non-functional requirements specified at the level of entire application are met
Test Basis	• System Requirements Specification and other application level specifications, e.g., regression test suites • Performance measurements baseline
Test Items	Whole application
Scope	All new, changed, and impacted areas of the application.
Entry Criteria	• Requirements to be verified are approved • An integration build of the application that implements new/changed requirements and/or bug fixes passes unit tests and static analysis and has code review issues resolved • Specification of the target deployment environment (hardware, software – OS, databases, browsers, cooperating systems) is approved • The environment that facilitates the tests is set up
Exit Criteria	• Tests that cover all new/changed requirements pass • Regression tests pass • All defects are addressed: re-tested and closed or postponed in a defect tracking system • Test reports are available and reviewed
Regression Test Approach	At the Design, Development and Continuous Integration stage, a set of regression tests is executed in a regular way following an integration build, e.g., every night, as defined in Test Plan At the Acceptance Testing stage: • A broader set of functional test cases that cover all possibly impacted areas is selected via analysis of new and changed requirements. This regression test set is executed at the Acceptance Testing stage

	• All tests that cover new and changed requirements are re-executed • Performance tests are executed
Test Automation Approach	• A subset of functional tests should be automated for regression testing run after an integration build • Additional test cases could be automated for data-driven testing • Test tools usage for performance tests • Test tools usage for testing web services
Test Methods	Black-box test methods are used, such as boundary values tests, positive/negative tests, state based testing, operational profiles, etc.
Accountable	Quality Assurance Lead
Responsible	Testers
Related documents	

Usability testing

Usability validation is a process of ensuring that basic usability checks have been verified while testing a feature or an application. Usability testing can be a separate phase of testing, however 10 usability heuristics designed by Jakob Nielsen (https://www.nngroup.com/articles/ten-usability-heuristics/)
are to ensure that basic usability validation is done.
The result of usability validation should be a list of recommendations to improve the usability but also information about what users liked or dislike while using application. While the focus should be on providing recommendations for the project team, defects in test management tool should be raised for those issues that are clearly not in line with acceptance criteria or UX requirements.

Each of the recommendation should receive a severity rating to prioritize the most serious problems. Severity should be decided based on three factors: frequency of the problem, easiness to overcome the problem, and persistence of the usability problem.
Based on the prioritization the most critical problems should be listed and placed in project backlog to be corrected.

Security Testing

Security Testing is type of non-functional testing (white box or black box) that is done to check whether the application or the product is secured or not.
Usually it is an authorized simulated attack on a system/server/application/device that looks for security weaknesses, potentially gaining access to the system's features and data.

In ABB we consider security testing as integral part of overall testing process for applications developed internally or bought from the third party vendors. All applications must be mandatory scanned by automatic scanners and penetration testing should be performed by designated teams from Information Security.
Development teams should perform manual security testing (e.g. using Burp Suite) and perform vulnerability scanning using Qualys tools. Only combination of manual testing and automatic security scans will reduce number of potential vulnerabilities.
Additionally manual penetration testing should be performed. This type of testing is done at the end of the development process. It should be done on final version of the code to be implemented into production environment.

Performance testing

For every project performance requirements must be determined and documented.
Following should be reflected:

- Response times (interactive work) for end users – defined by averages or percentiles
- Processing times (scheduled activities) – defined by averages or percentiles
- Throughput number of operations/transactions/transfer per time unit (typical hour, peak hour, off-hour)
- Concurrency number of users working with and idle while being logged in application/system at the same time
- Scalability required ability to be enlarged to accommodate growing amounts of work or data
- Availability prearranged level of operational performance will be met during a contractual measurement period – defined in percentage and allowed downtime schedule

Every performance requirement must be reviewed. Additionally it should be analyzed in different potential and planned contexts affecting performance, for example:

- Different volume of data processed
- Range of functionality implemented
- Location class
- Network connection (slow, fast, typical, VPN etc.)
- Operating system
- Kind of device (PC, mobile PC, handheld etc.)
- Workload profile
- Peak hour, typical hour, off hour
- Hardware platform (server and client)

Additional result of design review phase should be a list of risks associated with performance requirements in regard to used solutions and architecture. For example bottlenecks, single points of failure etc. These risks should be monitored in subsequent phases of the project.

During static analysis and code review an influence of individual system components on previously identified risks associated with performance requirements should be identified. Particular attention should be paid to extract the elements having biggest influence on final performance of the system or application.

As soon as the application or system prototypes are ready, measurements of values specified in the performance requirements should be made. At this stage, these measurements should be performed on each successive version of the prototype to monitor the risks of exceeding the performance requirements and avoid regression.

Stage and production testing phases of the project should be verification of the fulfillment of the previously accepted, verified and validated performance requirements.

System integration testing

Test Phase Purpose	The purpose is to exercise interactions between the application under test and other applications that are necessary to accomplish business processes defined in SRS
Test Basis	System Requirements Specification Enterprise architecture design Interface specification
Test Items	System comprised of the application under test and the cooperating application(s)
Scope	New and changed interface invocations between application under test and the cooperating applications.
Entry Criteria	• Requirements covering the new/changed interactions are approved • Integration builds of cooperating applications that implements new/changed interactions should pass unit tests, static analysis, code review and core functional regression tests • The environment that facilitates the tests is set up
Exit Criteria	• Tests that cover all new/changed interactions pass • Test reports available and reviewed

Regression Test Approach	• Whenever feasible, cross-application regression tests should be executed at the Design, Development and Continuous Integration stage • They must be included in the regression test suite at Acceptance Test stage
Test Automation Approach	Not applicable – manual testing mostly
Test Methods	Black Box methods (Requirements, Models, Interfaces, Data)
Accountable	Quality Assurance Lead
Responsible	Testers
Related documents	

User Acceptance Testing

Test Phase Purpose	To gain confidence that business requirements are met and the application is fit for deployment
Test Basis	Business requirements
Test Items	The whole Software Under Test
Scope	New/changed/unchanged requirements for the release
Entry Criteria	• SRS is fully implemented in the application • All new/changed requirements are re-tested in the Acceptance Test Stage by the product test team • Quality Assurance Lead and Solution Architect grants approval for UAT • Validation scenarios prepared by the application team (optional) • Stable environment (preferably Stage)
Exit Criteria	Formal approval is granted for deployment by the designated end user representatives specified in the Test Plan, e.g. Application Owners in the involved business unit
Regression Test Approach	End users should include testing of unchanged functionality in their exploratory testing
Test Automation Approach	Not applicable – manual testing
Test Methods	Exploratory testing, usage scenarios
Accountable	Requirements Provider/Business
Responsible	Designated end user representatives and IS QA Lead
Related documents	

Production deployment testing

Test Phase Purpose	To check that the application is operational in the Production Environment
Test Basis	Deployment Test Suite/Checklist
Test Items	Software Under Test deployed in the Production environment
Scope	Correctness of installation and configuration
Entry Criteria	• Stage Deployment Testing is completed • UAT Completed • Approval for deployment to Production is granted (Gate 5)
Exit Criteria	• Tests pass • Test reports are available and reviewed
Test Methods	• Checklist • Usage scenarios
Accountable	Application Manager* for Deployment correctness IS QA Lead for Usage Scenarios
Responsible	Developer Tester
Related documents	

* Please note that depending on Project Setup this role can be held by Solution Architect or Developer. This need to be agreed and described in Test Plan.

DEFECT MANAGEMENT

If a test is considered to have failed, a corresponding defect must be logged against it using defect tracking tool. This should be done by a person who executed this test or encountered the failure in any other way (e.g. using the application). Defect statistics can be used to make testing scope decisions.
Formal defect reporting shall be followed for all testing phases. Roles and responsibilities defined for the Defect Management process are described in section “Roles and Responsibilities”.
Responsibility chart including people assignment to roles is a part of Test Plan that must be created for each project.

Defect Severity

Critical Defect: The defect affects critical functionality or critical data. It does not have a workaround.
Example: Unsuccessful installation, complete failure of a feature.

High Defect: The defect affects major functionality or major data. It has a workaround but is not obvious and is difficult.

Example: A feature is not functional from one module but the task is doable if 10 complicated indirect steps are followed in another module(s).
Medium Defect: The defect affects minor functionality or non-critical data. It has an easy workaround. Example: A minor feature that is not functional in one module but the same task is easily doable from another module.
Low Defect: The defect does not affect functionality or data. It does not even need a workaround. It does not impact productivity or efficiency. It is merely an inconvenience. Example: Petty layout discrepancies, spelling/grammatical errors.

Defect Priority

Designation for defect priority depends on two factors : delivery model and tool used in project for defect management . It may be presented as a values from 1 (highest priority) to 4 or as level : Urgent, High, Medium or Low.
Details about defect priority must be described in Test Plan for particular projects (based on selected tool and delivery methodology).

Defect Statuses and Lifecycle

Process depends on few factors:

- selected delivery methodology
- tools selected to manage tasks and defects
- Application lifecycle – different approach for defects found during development phase and different for production issues.

Details must be described in Test Plan for particular projects.

ENTRANCE CRITERIA

Entrance criteria are the required conditions and standards for work product quality that must be present or met prior to the start of a test phase.

Waterfall approach

Test Planning:

- All required functional documentation is finished and available before test planning.

Test Execution:

- Prior test phase has been completed meeting its exit criteria.
- No open critical/major or average severity defects unless the issue is determined to be low impact and low risk defects remaining from the prior test phase.
- Development of all items to be tested is completed and deployed on test environment.
- Testing environment is configured and ready.
- All required information (test data requirements, test cases & scenarios,

testers are familiar with system being tested) is available before test execution.

Agile approach(Scrum)

Test Planning:

- User Story with clear and univocal Acceptance Criteria is available before test planning.
- Defined and shared Definition of Done.

Test Execution:

- No open critical/major defects impacting functionality (User Story) under tests.
- Development of all items to be tested is completed and deployed on test environment.
- Testing environment is configured and ready.

All required information (test data requirements, test cases & scenarios, testers are familiar with system being tested) is available before test execution.

EXIT CRITERIA

The set of generic and specific conditions, agreed upon with the stakeholders for permitting a process to be officially completed. The purpose of exit criteria is to prevent a task from being considered completed when there are still outstanding parts of the task which have not been finished. Exit criteria are used to report against and to plan when to stop testing.

More detailed criteria are identified by project manager after consultation with business team, analyst and test team on a specific project.

Waterfall approach

Exit criteria are the required conditions and standards for work product quality that block the promotion of incomplete or defective work products to the next test phase of the component. Exit criteria shall include the following:

- Successful execution of the test script(s) for the current test phase.
- No open critical, major, or average severity defects unless the issue is determined to be low impact and low risk.
- Component stability in the appropriate test environment.

Agile approach(Scrum)

Exit criteria define when the User Story can be considered as closed, and ready to be added to the current increment. They shall include:

- Successful execution of the Test Script (common approach is one Test Script per User Story).
- Meeting all Acceptance Criteria (agile approach allows changes in AC when agreed with Product Owner).
- No open critical, major or average severity defects (unless the issue is determined to be low impact and low risk).

According to Scrum, above exit criteria should be part of definition of done and should be confirmed by entire Team.

TEST ENVIRONMENTS

The Test Plan should enumerate environments that will be used to test an application.

A test environment description should specify properties that facilitate:

- systematic recovery of a clean environment
- deterministic test execution
- comparison of the test environment to the target Production environment

This should include:

- hardware for test execution, virtualization solutions, networking setup, e.g. isolation in a sub-network separated by a firewall,
- software environment into which the application will be deployed: operating systems, application server and database server versions, browser types and versions on client machines, and other,
- connections to other cooperating applications,
- options for configuration of the application installation,
- data sets and data feeds to be used in testing.

The exact test environment types and the number of instances depend on the application under test and the anatomy of its deployment pipeline. The common environments are listed below:

Build Verification: facilitates execution of automated tests following an integration build. This involves initialization of the environment with specific data sets for deterministic tests.

Development: used for frequent, e.g., daily deployment of builds that passed automated tests but have not been subject to manual regression tests yet. It is used by developers and testers for preliminary testing of Change Requests/Features and for development of tests.

Test: a family of environments used for Application and System Integration Testing to verify formally that a requirement has been implemented and to run regression tests. These environments should be as close to the Production environment as it is practical. At the same time, they should be isolated enough to be kept in a stable configuration. Separate environments might be needed for functional and for performance testing. Deployments to Test environments are less frequent than they are to

Development environment due to time it takes to execute a test cycle and the need to keep the environment stable throughout the cycle.

Stage: ideally, identical to the Production environment, used for validation of the application in its intended environment via Stage Deployment Testing and User Acceptance Testing. The degree of control of this environment by the test team might be limited. The Stage environment could be switched with the Production environment at the release time.

Production: used for deployment verification and integration with ABB external applications that could not be tested in Stage.

TEST TOOLS

Test Scenario Management and Defect Management for **new project** , ABB recommended to use Azure Dev Ops Services(VSTS).
For Test Automation ABB recommended to use Selenium WebDriver or HP UTF or Katalon.
For Performance and API testing - ABB recommended to use JMeter /Apica/Gatling.
For Code quality ABB recommend to use SonarQube and ReSharper.

METRICS

Software testing metrics are a way to measure and monitor test activities. If you measure the correct metrics in a right way and transparently, they will guide you to understand the team progress towards certain goals and show the team's successes and deficiencies.
The whole team approach also critical on the metrics that you will measure and report so it is very important to popper introduce metrics.

Test execution reporting should contain below information:

- Overall number of test cases (by status)
- Test execution trend
- Number of defects and defect severity distribution
- Defect resolution time
- Defect open vs defect closed (trend)

All list is available under this link

For more complex projects more sophisticated metrics for reporting might be created. (please note that those ale only example metrics, Test Plan for specific application should defined what kind of data should be gather and presented as a metrics).

- Percentage of escaped defects
- Percentage of rejected defects
- Percentage of duplicate defects
- Critical/high severity defects index

- Test Effectiveness
- Percentage of test automated (regression)
- Percentage of false negatives (automated tests)
- Test Case Related defects
- Percentage of Passed Tests
- Turnaround time of regression testing

ABB recommended to use metrics build in/or configurated with test management tool. Details about metrics scope and availability must be agree on Test Plan for particular project.

REPORTING

On daily basis test progress should be tracked by Quality Assurance lead. In Agile (Scrum) daily standups is place when status is provided.
Before each release -Test Team/ Development team should prepare summarize report with overall quality .It should contains test execution status (from each phase of testing) list of defects and their status and metrics gather during the development phase. Test Report must be send to IS QA Lead/Solution Architect/ Service Manager.
For User Acceptance Testing ABB recommends daily reports about progress and reported issues.

TEST PLAN AND STRATEGY IMPLEMENTATION GUIDELINES

The aim of this section is to specify how to use the Test Strategy document in a project. The Test Strategy does not constitute a detailed prescription of how testing should be done in a project for several reasons:

- Its scope is very broad, covering verification and validation activities across the whole software development lifecycle
- The Test Strategy will get more precise and include additional important information with time and feedback from application teams
- Test Strategy does not follow single software development methodology
- Testing processes need to be adjusted to the project release scope and schedule, quality risks, architecture and other project and application-specific factors
- The Test Strategy mandates that each project must have its Test Plan document that comprehensively prescribes the scope, approach, resources, and schedule of the testing activities in the project.

Test Strategy is a guideline for building a Test Plan. A Test Plan Template is a companion document that is associated with a version of Test Strategy. A software project team is also to follow the referenced processes and procedures that are available now, applying their tailoring guidelines to adjust to the project at hand. Additional process materials, trainings and the tools infrastructure to help carry out

these processes will be systematically developed following short-term organizational objectives in regards to ABB Test Strategy.

Test Plan should include

- mapping of the test phases specified in this document into the software development process followed by an application team
- specifics of test phase implementation
- release-specific information pertaining to testing, like the schedule of testing activities, staffing and application quality risks
- quality improvement goals and testing process improvements planned in the release – optional, in case these goals are not stated elsewhere
- other defined in template like defect management, tools

IS QA Lead is the owner of the Test Plan He/she is also responsible for maintenance of the document.

TESTING IN AN OUTSOURCED DEVELOPMENT SETUP

Some ABB software products are maintained by External Vendors (an outsourcing partner) that develops subsequent versions of the product. A degree of control over the testing process at the outsourcing partner may vary. Nevertheless, the handover package delivered by an external supplier must include, besides the product itself, an evidence that testing was done. With respect to testing, the outsourced development setup creates the following obligations for the outsourcing partner and the receiving party in ABB.

Requirements for a contracted outsourcing partner:

- Follow ABB Test Strategy
- Create and execute on its own (Supplier) Test Plan based on ABB Test Plan template
- Supplier should use Test Management Tool recommended by ABB
- ABB should have access to Test Management Tool in order to check test execution , traceability with requirements and list of defects
- With each supplied version, provide Release Notes that state whether the acceptance criteria were met and lists of known issues in the supplied deliverables – missing or unimplemented requirements, open defects
- Upon request, attach to Release Notes:
  - a test execution report
  - requirement-to-test traceability matrix
  - a list of defects closed in testing
- Hand over a test suite that was used in any Phase of Testing – including Test Automation solution and scripts
- Hand over of Test Cases

Requirements for the receiving party in ABB:

- Maintain Quality Assurance Lead role, a contact to the outsourcing partner on testing

- Agree with the contracted partner on the defect classification and establish a channel for communication of defects.
- Include quality expectations in the acceptance criteria for the deliverables.
- Review the Supplier Test Plan and request corrections if necessary.
- Review Release Notes and testing deliverables (test report, test suite, defect lists) and request corrections if necessary.
- Maintain an internal Test Plan (may contain references to the external Supplier Test Plan) that includes.
  - Functional Testing (might include re-execution of tests provided by the outsourcing partner as well as execution of test cases developed internally)
  - System Integration Testing
  - Stage Deployment Testing
  - User Acceptance Testing
  - Production Deployment Testing
- Maintain tests suites for internal testing; adapt test cases supplied by the out-sourced partner for internal testing.

ROLES AND RESPONSIBILITIES (INCLUDING DEFECT MANGANGEMENT)

Functional Analyst

Functional Analyst is responsible for gathering, creation, maintenance of the project requirements. In regards to testing, the Functional Analyst shall:

- validate description of detailed requirements
- aid Developers and Testers in understanding requirements
- participate in Requirements Reviews
- aid stakeholders in User Acceptance Testing
- support defect triage :

  Participate to agree on the priority and action for defects as well as for rejecting and deferring defects. Consults Testers and Development Teams on the expected solution behavior and provides clarifications. SME’s for business.

Stakeholder/Requirements Provider

The stakeholder may be an external party (business partner, end user) or internal (development team member) providing requirements. In regards to testing, the Stakeholder shall do:

- in Requirements Review, validate description of business requirements
- if needed, assist in a Design Review pertaining to requested requirements
- participate in User Acceptance Testing and approve implementation of a requested requirement

Release Manager (for some cases Project Manager or Application Manager)

Release Manager has the overall responsibility for the release of an application within the approved scope and budget according to the project plan. In regards to testing, the Release Manager shall:

- ensure that resources (staff, environments, tools) specified in the approved Test Plan are available to the project,
- include testing tasks specified in the Test Plan in the overall project plan,
- cooperate with Technical Lead and Quality Assurance Lead in execution of the Test Plan,
- cooperate with Functional Analyst and IS QA Lead execution of User Acceptance Tests and solicit the Stakeholders for their approval of implementation of their requirements and the overall readiness of application for the release.

Solution Architect

- Write up of the technical specifications based on the BBEM
- Create Solution Design with Architecture
- Initiate Architecture Review
- Approves Architecture
- Support the delivery team in their implementation
- Validate the solution from technical prospective before its submission to the functional validation
- Maintain the technical documentation updated
- Support the preparation of the End User training material
- Support the RUN organization to fix an incident/problem if it affects the Technical design of a solution.

Technical Lead

The Technical Lead role supervises the development team (Developers) and is responsible for delivery of technical solution to the requested requirements. In regards to testing, Technical Lead:

- defines approach and targets for , Unit Testing, Static Analysis and Code Review
- provides input on the above to the Test Plan
- initiates and supervises Design Reviews
- supervises the execution of the aforementioned test phases, reports to Release Manager on their progress towards acceptance criteria, provides relevant metrics
- makes sure Developers have necessary skills and resources to perform the above tests
- makes sure the integration builds provided to the test team fulfill the entry criteria for this phase
- notifies about the readiness of the project - Implementation Baseline for the Acceptance Testing,
- participates in the reviews of Quality Risks,
- participates in defect triage meetings,
- coordinates correction of defects found in testing,

- cooperates with Quality Assurance Lead on making the application testable and on test automation,
- provides environments for testing by the development and test teams

Developer/Development team (or Vendor)

Developer implements a technical solution for the assigned requirements and also corrects defects. In regards to testing, Developer shall:

- develop Unit and Integration Tests related to the code he delivers as specified in the Test Plan
- perform basic functional testing of the delivered requirement implementation or bug fix
- ensure that the code committed to a shared branch for an integration build does not break the build and passes Static Analysis, Unit and Integration testing
- initiate and participate in Code Reviews
- participate in Requirements and Design Reviews as requested by Technical Lead
- Analyze and fix defects considering all quality assurance best practices. Perform internal peer review and unit test the defect fix. Prepare deployment package that could be propagated by IO Team to upper environments (including environment for which defect was raised).

- if needed, assist testers in automation of functional tests (In Scrum tester is part of Development team)

IS Quality Assurance Lead

This role is accountable for test phases independent of development: Functional Testing, System Integration Testing, User Acceptance Testing and Stage Deployment Testing. Also, he/she is the owner of the Test Plan. IS QA Lead supervises the test team (Testers).
IS QA Lead shall:

- build the Test Plan for the aforementioned test phases, including:
    - product quality risks and their mitigation countermeasures,
    - test design methods and test automation,
    - estimates for test activities,
    - resource requirements (staff, environments, tools),
    - schedule (in cooperation with the Release Manager)
- solicit the Technical Lead for parts of the Test Plan related to Design Reviews, Unit Testing, Integration Testing, Static Analysis and Code Reviews
- supervise the execution of the Functional Testing, System Integration Testing and Stage Deployment Testing phases by Testers, report to the Release Manager the progress towards acceptance criteria, provide relevant metrics
- communicate defects found in these test phases, supervise defect management in the defect tracking system and organize defect triage CCB meetings

- make sure Testers have necessary skills and resources to perform the assigned tests
- inform the project steering committee on the progress of testing and quality of the application under test, provide relevant metrics
- notify interested parties about readiness for UAT
- notify interested parties about the readiness of the project - Product/Release baseline for deployment into the Production environment
- organize reviews of product quality risks
- monitor progress of testing phases towards their acceptance criteria
- cooperate with IS QA Leads of other projects on System Integration Testing
- cooperate with Technical Lead on making the application testable and on test automation
- contribute to the overall Application Operations test strategy and testing processes
- for IS QA Lead responsibilities related to Manage Services please refer to section “TESTING IN AN OUTSOURCED DEVELOPMENT SETUP”

Tester/Development Team

The Testers perform Functional Testing, System Integration Testing , Non-functional Testing and Stage Deployment Testing. Tester shall:

- apply black-box and gray-box methods to design test cases for functional and nonfunctional requirements,
- implement test cases, including test automation
- execute test cases and file defect reports
- prepare test data
- create test automation frameworks
- participate in Requirements and Design Reviews as requested by IS QA Lead
- cooperate with Developers on reproducing defects and their analysis
- provide estimates on the assigned testing tasks and report on their progress
- Identifies and logs defects, provides supporting information, and validates the fix in the product once it is available.

REFERENCES

1. https://www.istqb.org/
2. https://www.istqb.org/downloads/glossary.html
3. https://www.nngroup.com/articles/website-response-times/
4. https://www.nngroup.com/articles/ten-usability-heuristics/
5. https://abb.sharepoint.com/teams/AppTechmgmtteam/Shared%20Documents/Forms/AllItems.aspx?id=%2Fteams%2FAppTechmgmtteam%2FShared%20Documents%2FPublic%2FABB-Dev-Code%20Review%20Process%2Epdf&parent=%2Fteams%2FAppTechmgmtteam%2FShared%20Documents%2FPublic
6. https://share.library.abb.com/api/v4?cid=Root&q=9AAD135107
7. http://search.abb.com/library/Download.aspx?DocumentID=9AAD135096&LanguageCode=en&DocumentPartId=&Action=Launch

RECOMMENDED READING

1. https://leaksource.files.wordpress.com/2014/08/the-web-application-hackers-handbook.pdf

REVISION HISTORY

Rev.	Page	Change Description	Author(s)	Date
A	all	initial version	Anna Pietras	2018-10-06
B	all	update	Anna Pietras	2019-01-24

Information Systems

Applications Performance Excellence (APE)

Quality Assurance & Test- Standards & Practices