<epam>

Test Planning and Its Tasks and Goals

Software Testing Introduction

TRAINING

CENTER

<epam>

Read and remember!

Test planning – the activity of establishing or updating a test plan.

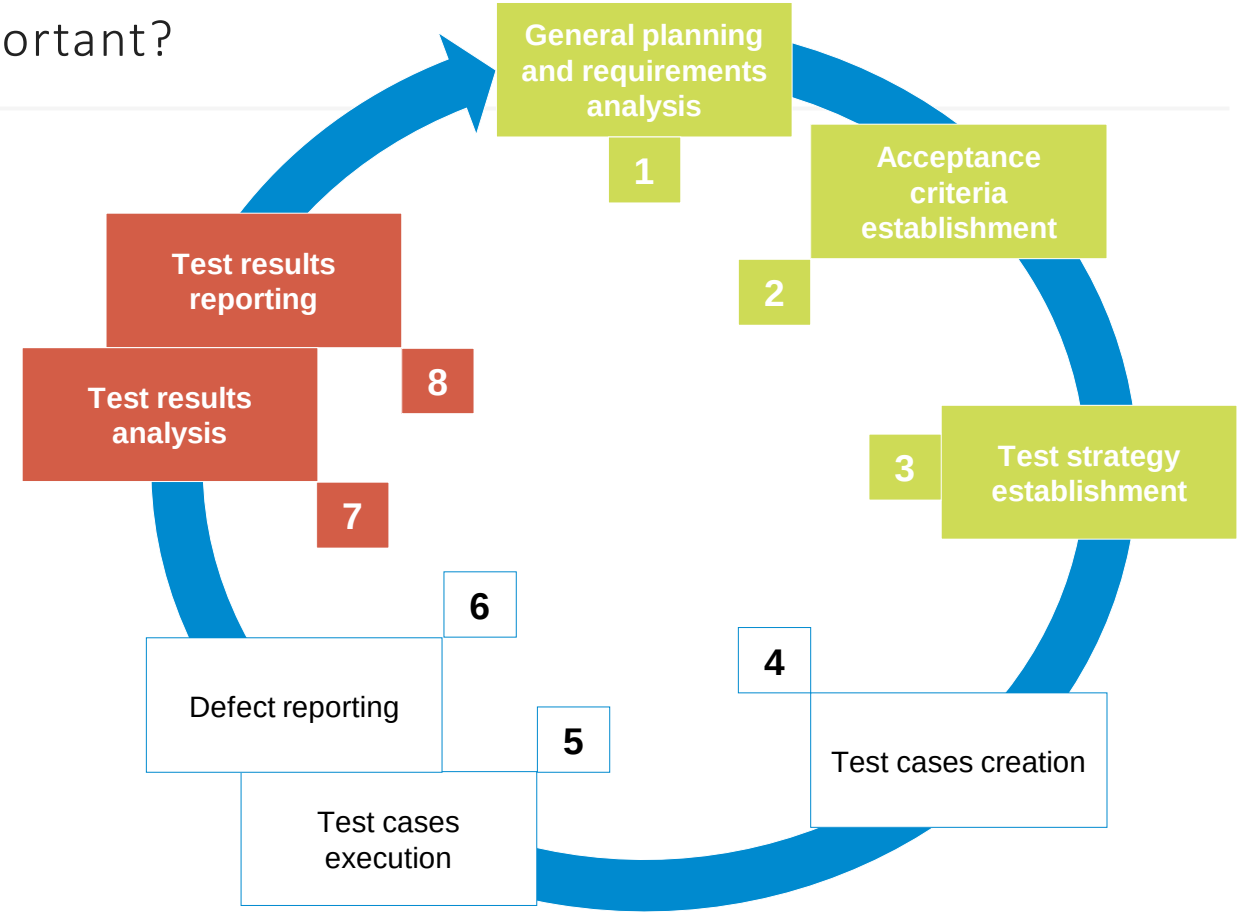
Test plan – documentation describing the test objectives to be achieved and the means and the schedule for achieving them, organized to coordinate testing activities.

Planning Tasks and Goals

Estimation of work scope and complexity	Designation of resources and ways to acquire them	Definition of schedule and milestones
Risk mitigation and countermeasures definition	Distribution of duties and responsibilities	Coordination of team-effort between teams/projects/groups

Why is Test Plan so important?

Test Plan is the basis for all further activities, it is a map to the defined destination, a starting point to the reporting and so on and so forth…



Planning Tasks and Goals (part 2)

Test Plan helps us to…

Get better results with less effort	Optimize resources usage, avoid waste	See the progress at any given moment
Understand what to do in any situation	Get better understanding between people	Cooperate better

<epam>

Test Planning and Its Tasks and Goals

Software Testing Introduction

TRAINING

CENTER

<epam>

<epam>

Test Plan Sections

Software Testing Introduction

TRAINING

CENTER

<epam>

Read and remember!

Test plan – documentation describing the test objectives to be achieved and the means and the schedule for achieving them, organized to coordinate testing activities.

General Test Plan Sections Overview

- Project scope and main goals
- Requirements to be tested
- Requirements NOT to be tested
- Test strategy and approach
- Criteria
- Resources
- Schedule
- Roles and responsibilities
- Risk evaluation
- Documentation
- Metrics
- …

Read and remember!

Project scope and main goals – a very brief description of the purpose of the application development.

This section is reminiscent of business requirements, but here the information is presented in an even more condensed form with an emphasis on the most important tasks.

E.g.:

Correct automated conversion of text documents in different source encodings to one destination encoding with performance significantly higher than human performance during the same actions.

Read and remember!

Requirements to be tested – the list of functional and / or non-functional requirements to be tested.

Not each and every requirement should be tested. Here we list those ones that should be covered with test-cases.

E.g.:

- *UR-1.*: smoke test.*
- *UR-2.*: smoke test, critical path test.*
- *UR-3.*: critical path test.*
- *BR-1.*: smoke test, critical path test.*
- *QA-2.*: smoke test, critical path test*

Read and remember!

Requirements NOT to be tested – the list of functional and / or non-functional requirements NOT to be tested.

As not each and every requirement should be tested, here we list those ones that should NOT be covered with test-cases. For each requirement we write the reason why shouldn't we test it.

E.g.:

- *SC-1: the application is a console one by design.*
- *SC-2, L-1: the application is developed with proper PHP version.*
- *QA-1.1: this performance characteristic is at the bottom border of typical operations performance for such applications.*

Read and remember!

Test strategy and approach – the description of the testing process in terms of the methods used, approaches, types of testing, technologies, tools, etc.

This description allows us to use the most effective and efficient way to achieve project goals in terms of quality.

E.g.:

Due to the team cross-functionality, a significant contribution to quality improvement can be expected from the code review combined with manual testing using the white box method. Unit-testing will not be applied due to extreme time limitations.

Read and remember!

Criteria – the list of miscellaneous testing criteria, such as acceptance criteria, entry criteria, suspension criteria, resumption criteria, exit criteria, etc.

Usually, each criterion has a reference to proper metric. Otherwise the decision on criterion fulfillment may be too subjective.

E.g.:

Testing resumption criteria: more than 50% of bugs found during the previous iteration are fixed (see "Ongoing defects fixed percentage" metric).

Read and remember!

Resources – the list of miscellaneous project resources, such as software, hardware, staff, time, finance, etc.

Usually, finance resources are too confidential to be included in test-plan, so only a reference to the budget is mentioned here.

E.g.:

- *Software: four virtual machines (two with Windows 10 Ent x64, two with Linux Ubuntu 18 LTS x64), two PHP Storm licenses (latest version available).*
- *Hardware: two standard workstations (8GB RAM, i7 3GHz).*

Read and remember!

Schedule – in fact, this is a kind of calendar with milestones and periods marked on it.

When the project is long enough (months and years long) there is no need to have a detailed schedule for the far future: the schedule may be adjusted and filled with details once we reach one milestone or another.

E.g.:

- *25.05 – requirements testing and finalizing.*
- *26.05 – test-cases and scripts for automated testing creation.*
- *27.05-28.05 – main testing stage.*
- *29.05 – testing finalization, reporting.*

Read and remember!

Roles and responsibilities – the list of all key project roles (related to the testing process) with their key areas of responsibility.

Though this list is rather unified for most projects, there may be some special cases, so this section usually is not omitted.

E.g.:

- *Senior developer: participation in requirements testing and code review.*
- *Tester: documentation creation, test-cases execution, participation in code-review.*

Read and remember!

Risk evaluation – the list of risks that are likely to arise during the project. For each risk there is a probability evaluation and some options for resolving the situation.

There are a lot of typical risks (actual for every project) so no need to list such risks here. This section is for risks specific (or even unique) to the project.

E.g.:

- *Personnel (low probability): if any team member is inaccessible, we can contact the representatives of the "Cataloger" project to get a temporary replacement (the commitment from the "Cataloger" PM John Smith was received).*

Read and remember!

Documentation – the list of test documentation with details on who should prepare it, when, how, etc.

This section is extremely useful when onboarding a new project member. As sometimes documentation list may consist of dozens of items, it's good to have them listed in one place.

E.g.:

- *Requirements. Responsible person – tester, deadline – 25.05.*
- *Test cases and defect reports. Responsible – tester, creation period – 26.05-28.05.*
- *Test result report. Responsible person – tester, deadline – 29.05.*

Read and remember!

Metrics – the list of numerical characteristics of quality indicators, methods for their evaluation, formulas, etc.

This section is actively referenced by “Criteria” section. Metrics have their dark side too, but in general it is better to have some objective way to tell the current project situation.

E.g.:

Test cases success percentage:

$T^{SP} = \frac{T^{Success}}{T^{Total}} \cdot 100\%$, *where*

T^{SP} – *percentage of successfully passed test cases,*
$T^{Success}$ – *quantity of successfully passed test cases,*
T^{Total} – *total quantity of executed test cases.*

<epam>

Test Plan Sections

Software Testing Introduction

TRAINING

CENTER

<epam>

«File Converter» Project

Test Plan

SAMPLE

Project Documentation

Background	Estimations, schedule, strategy, and metrics are needed to organize the testing process efficiently.
Purpose	To organize the testing process effective and efficient during the whole project period.
Scope	Testing process description, metrics, schedule, resources.
Audience	Management staff, QA team, project team.
File	02 03 - Test Plan Sample.docx

Contents

1. Project scope and main goals 3
2. Requirements to be tested 3
3. Requirements NOT to be tested 3
4. Test strategy and approach 3
   4.1. General approach 3
   4.2. Functional testing levels 3
5. Criteria 4
6. Resources 4
7. Schedule 4
8. Roles and responsibilities 4
9. Risk evaluation 4
10. Documentation 5
11. Metrics 5

1. Project scope and main goals

Correct automated conversion of text documents in different source encodings to one destination encoding with performance significantly higher than human performance during the same actions.

2. Requirements to be tested

See referenced sections in “File Converter Requirements.docx”:

- UR-1.*: smoke test.
- UR-2.*: smoke test, critical path test.
- UR-3.*: critical path test.
- BR-1.*: smoke test, critical path test.
- QA-2.*: smoke test, critical path test.
- L-4: smoke test.
- L-5: smoke test.
- DS-*: smoke test, critical path test.

3. Requirements NOT to be tested

See referenced sections in “File Converter Requirements.docx”:

- SC-1: the application is a console one by design.
- SC-2, L-1, L-2: the application is developed with proper PHP version.
- QA-1.1: this performance characteristic is at the bottom border of typical operations performance for such applications.
- L-3: no implementation required.
- L-6: no implementation required.

4. Test strategy and approach

4.1. General approach

The application is to be configured once by an experienced specialist and later used by end users, for whom only one operation is available – placing the file into the input directory. Therefore, issues of usability, security, etc. not explored during testing.

4.2. Functional testing levels

- Smoke test: automated with batch files under Windows and Linux.
- Critical path test: executed manually.
- Extended test: not executed as the probability of defects detection on this level is negligibly small.

Due to the team cross-functionality, a significant contribution to quality improvement can be expected from the code review combined with manual testing using the white box method. Unit-testing will not be applied due to extreme time limitations.

5. Criteria

- Acceptance criteria: 100% success of test cases on smoke test level and 90% success of test cases on critical path test level (see "Test cases success percentage" metric) if 100% of critical and major bugs are fixed (see "Overall defects fixed percentage" metric). Final requirements coverage by tests (see "Requirements coverage by tests" metric) should be at least 80%.
- Testing start criteria: new build.
- Testing pause criteria: critical path test must begin only after 100% success of test-cases on the smoke test (see "Test cases success percentage"); test process may be paused is with at least 25% test-cases executed there is at least 50% failure rate (see "Stop-factor" metric).
- Testing resumption criteria: more than 50% of bugs found during the previous iteration are fixed (see "Ongoing defects fixed percentage" metric).
- Testing finish criteria: more than 80% planned for the current iteration test cases are executed (see "Test-cases execution percentage").

6. Resources

- Software: four virtual machines (two with Windows 10 Ent x64, two with Linux Ubuntu 18 LTS x64), two PHP Storm licenses (latest version available).
- Hardware: two standard workstations (8GB RAM, i7 3GHz).
- Personnel:
    - One senior developer with testing experience (100% workload during all project time). Roles: team lead, senior developer.
    - One tester with PHP knowledge (100% workload during all project time). Role: tester.
- Time: one workweek (40 work hours).
- Finances: according to the approved budget.

7. Schedule

- 25.05 – requirements testing and finalizing.
- 26.05 – test-cases and scripts for automated testing creation.
- 27.05-28.05 – main testing stage (test-cases execution, defect reports creation).
- 29.05 – testing finalization, reporting.

8. Roles and responsibilities

- Senior developer: participation in requirements testing and code review.
- Tester: documentation creation, test-cases execution, participation in code-review.

9. Risk evaluation

- Personnel (low probability): if any team member is inaccessible, we can contact the representatives of the "Cataloger" project to get a temporary replacement (the commitment from the "Cataloger" PM John Smith was received).

- Time (high probability): the customer has indicated a deadline of 01.06, therefore time is a critical resource. It is recommended to do our best to complete the project by 28.05 so that one day (29.05) remains available for any unexpected issues.
- Other risks: no other specific risks have been identified.

10. Documentation

- Requirements. Responsible person – tester, deadline – 25.05.
- Test cases and defect reports. Responsible – tester, creation period – 26.05-28.05.
- Test result report. Responsible person – tester, deadline – 29.05.

11. Metrics

- Test cases success percentage:

  $T^{SP} = \frac{T^{Success}}{T^{Total}} \cdot 100\%$, where

  T^{SP} – percentage of successfully passed test cases,

  $T^{Success}$ – quantity of successfully passed test cases,

  T^{Total} – total quantity of executed test cases.

  Minimally acceptable borders:
  - Beginning project phase: 10%.
  - Main project phase: 40%.
  - Final project phase: 80%.

- Overall defects fixed percentage:

  $D_{Level}^{FTP} = \frac{D_{Level}^{Closed}}{D_{Level}^{Found}} \cdot 100\%$, where

  D_{Level}^{FTP} – overall defects fixation percentage by $Level$ during all project lifetime,

  D_{Level}^{Closed} – quantity of defects of $Level$ fixed during all project lifetime,

  D_{Level}^{Found} – quantity of defects of $Level$ found during all project lifetime.

  Minimally acceptable borders:

		Defect severity			
		Minor	Medium	Major	Critical
Project phase	Beginning	10%	40%	50%	80%
	Main	15%	50%	75%	90%
	Final	20%	60%	100%	100%

- Ongoing defects fixed percentage:

  $D_{Level}^{FCP} = \frac{D_{Level}^{Closed}}{D_{Level}^{Found}} \cdot 100\%$, where

  D_{Level}^{FCP} – defects fixation percentage by $Level$ (defects found in the previous build and fixed in the current build),

  D_{Level}^{Closed} – quantity of defects of $Level$ fixed in the current build,

  D_{Level}^{Found} – quantity of defects of $Level$ found in the previous build.

Minimally acceptable borders:

		Defect severity			
		Minor	Medium	Major	Critical
Project phase	Beginning	60%	60%	60%	60%
	Main	65%	70%	85%	90%
	Final	70%	80%	95%	100%

- Stop-factor:

$S = \begin{cases} Yes, T^E \geq 25\% \;\&\&\; T^{SP} < 50\% \\ No, T^E < 25\% \;||\; T^{SP} \geq 50\% \end{cases}$, where

S – decision to pause the testing process,
T^E – current T^E value,
T^{SP} – current T^{SP} value.

- Test-cases execution percentage:

$T^E = \frac{T^{Executed}}{T^{Planned}} \cdot 100\%$, where

T^E – test-cases execution percentage,

$T^{Executed}$ – quantity of executed test-cases,

$T^{Planned}$ – quantity of planned (to execution) test-cases.

Levels (borders):
  - Minimal: 80%.
  - Desired: 95%-100%.

- Requirements coverage by tests:

$R^C = \frac{R^{Covered}}{R^{Total}} \cdot 100\%$, where

R^C – requirements coverage by tests (percentage),

$R^{Covered}$ – quantity of requirements covered with test-cases,

R^{Total} – overall quantity of requirements.

Minimally acceptable borders:
  - Beginning project phase: 40%.
  - Main project phase: 60%.
  - Final project phase: 80% (90%+ recommended).