Process Monitoring Test Plan

[**Introduction**](#_pcoegr8pkovp) **1**

[**Test Items**](#_5mieiqkjlbl) **2**

[**Features to be Tested**](#_g7gv648nmi79) **2**

[**Features not to be tested**](#_jpdrtlp6t12k) **2**

[**Approach**](#_x6chtrz8ozzb) **2**

[Test levels](#_4ikmmh9n6f54) 2

[Unit testing](#_5ob9uyi9n5x4) 2

[Integration testing](#_xlcvt5yixfbz) 2

[System testing](#_jso8e4qvylhy) 3

[Acceptance testing](#_7ni89xlea9ii) 3

[Test strategy](#_sheoh0789off) 3

[**Item pass/fail criteria**](#_nlvduz2687ym) **3**

[**Suspension criteria and resumption requirements**](#_o22na0v8afkn) **3**

[**Test deliverables**](#_4du7szqbosig) **4**

[**Responsibilities**](#_1ul3b3wjzzgq) **4**

[**Environmental needs**](#_1navng3qkvze) **4**

[**Staffing and training needs**](#_1yra9cze4vnp) **5**

[**Schedule**](#_8jwckcqmeb8t) **5**

[**Risks and contingencies**](#_72yx9d8ggfc1) **5**

[Product risks](#_2ww438lb4sre) 5

[Project risks](#_kvsizr546b15) 5

[**Approvals**](#_651kzskz9in9) **6**

# Introduction

The process monitoring application receives a process name, a duration monitoring period in seconds and optionally a polling time in seconds.

It is written with Python and it can be imported in other modules, used in the console or with its graphical user interface.

# Test Items

* metrics.py
* process\_monitor.py
* process\_monitor\_gui.py

Use <release> tag from GIT for all test items. Repository is at: <https://github.com/nicolae-chedea/process-monitor>

# Features to be Tested

All features will be tested:

* ability to collect metrics about a process by name
* interact with the application from the console
* interact with the application from the graphical user interface
* generate a csv file with the generated process metrics

# Features not to be tested

Not applicable

# Approach

## Test levels

### Unit testing

Unit testing will be performed by the development team for each Python class.

Unittest module from Python will be used for automated unit testing.

Expected entry criteria before starting unit testing are code to be completed, reviewed and pushed on the release branch.

Statement coverage will be measured during this phase.

Testing will finish once all tests are executed and pass rate is 100%.

### Integration testing

Integration testing will also be performed by the development team for each Python class, with focus on interfaces and data flow between the classes.

Unittest module from Python can also be used for automated integration testing.

Expected entry criteria are unit testing to be completed.

All interfaces are to be exercised.

Integration testing will finish when all tests are executed and pass rate is 100%.

### System testing

System testing will use black box testing techniques and will be done by the testing team.

Testing will begin with exploratory testing and continue with requirement based testing.

Requirement based tests will also be automated using Python and RobotFramework.

Testing will only start after integration testing is finished and requirement analysis is finished and open points are clarified.

Metrics collected during this testing phase are requirement coverage and automated tests from the total number of tests.

System testing will finish when requirement coverage is at least 90%, test pass rate is at least 90% and all known issues are considered as low priority by the product team.

### Acceptance testing

Will be done by the client at its site, will focus on testing real life, end to end scenarios.

## Test strategy

All tests will begin with the positive scenarios and continue with negative scenarios.

Tests will be based on existing documentation, specific for each phase.

All tests will be documented, including exploratory tests and all test results will be captured, analysed and stored.

Where possible, specific test design techniques will be used to reduce the number of tests and test runtime while keeping the same coverage as before, like: boundary value analysis, equivalence partitioning, decision tables, combinatorial techniques etc.

Static testing will also be used especially at unit testing level with the goal of measuring cyclomatic complexity of the code and identifying unreachable code or just identifying coding guidelines issues.

Compatibility testing is also important in this case, assessed by testing the application on different operating systems.

# Item pass/fail criteria

Item pass/fail criteria is defined at each test level. As most of the tests will be automated, reviewing the tests is important to make sure the evaluation criteria is the right one, based on design and specification documents.

All the automated tests should be independent so that a failing test doesn’t cause any subsequent failures in other tests.

# Suspension criteria and resumption requirements

Testing may be suspended in the following cases:

* client asks this - resume from where the testing was suspended
* other priorities appear and all the stakeholders approve this - resume from the testing was suspended
* there are major changes in the functionality - restart from the beginning
* unforeseen natural disasters - resume from where the testing was suspended
* unplanned changes in the team - resume from where the testing was suspended, with additional time planned for training new team members
* during each sprint if scope changes, adjust the test planning with the scrum master and the test manager

# Test deliverables

Test deliverables will be available for each test phase, in general they will include:

* test case - either in the test case management tool or in the form of a test script
* test data - where applicable
* test reports
* issues - collected in the bug tracking tool

# Responsibilities

Each test engineer will have multiple tasks:

* take part in requirement analysis and raise any potential issues from a testing perspective
* taking part in design specification reviews
* taking part in unit and integration testing reviews and act as a consultant for these steps
* create test cases
* automate test cases and create test scripts
* create a CI/CD pipeline to run automatically the test scripts
* report bugs
* adhere to the sprint planning and report on time any possible delay

The test manager additionally will also maintain the test plan, collect metrics and in general facilitate anything related to the development process.

The scrum master will help the testing team with planning and staying on track.

# Environmental needs

Besides the local machines that will be used by the testing team, a cloud setup is required for CI/CD. The minimum requirements are 3 machines - one for each supported operating system - Windows10, Ubuntu and MacOs.

# Staffing and training needs

For the test engineer in the team the following requirements are necessary:

* testing fundamentals knowledge, ISTQB Foundation certification preferred
* Python programming knowledge
* operating knowledge on Windows, Ubuntu and MacOS

Where gaps exist the test manager should schedule trainings - internal if possible, external if not.

# Schedule

Schedule will be tracked in another document and updated in each sprint by the test manager and the scrum master.

# Risks and contingencies

## Product risks

* Changing customer requirements

Likelihood: High

Impact: Medium, can affect project scope.

Mitigation: discuss any change to see how necessary it really is and adjust sprint scope based on the necessities.

## Project risks

* budget constraints

Likelihood: Low

Impact: Low, can affect project scope.

Mitigation: stick to original planning, identify from the beginning areas where savings can be done.

* staffing issues

Likelihood: Medium

Impact: Medium, affects team structure

Mitigation: document everything done on the project, have trainings available for new team members so ramp up takes less and keep the team members interested by having appropriate tasks

* environment issues

Likelihood: Medium

Impact: Low, impacts execution

Mitigation: have back-ups where possible, pick suppliers with a high service level agreement.

# Approvals

This test plan is created by the test manager, reviewed by senior test engineers and other engineering managers