I S S A I

Framework to run automated tests specified in Kiwi Test Case Management System

Requirement Specification

Version 0.1

Table of contents

Aim of the project

Kiwi test case management system (TCMS) offers attributes to specify automated test plans and test cases, however, it does not provide the functionality to do so.

The project's goal is to implement a customizable framework to run test plans or test cases specified in Kiwi TCMS. The name issai was chosen, because it denotes a small sort of kiwi fruit.

The framework also implements export and import functionality, which is not contained in Kiwi either.

Exports are useful for backing up test specifications and running tests offline without connection to Kiwi TCMS.

Imports can be used to restore test specifications and allows to create test specifications in another tool or a document.

Requirements

General

RQ-1 Supported platforms
The framework shall support the operating systems Linux, BSD and Windows; on all architectures where a Python ecosystem is available.

RQ-2 Unversioned products
The framework shall support the labeling scheme, where a product is unversioned.
The framework shall rely on the fact, that the Kiwi TCMS product will not contain software versions or builds other than the defaults in that case.

RQ-3 Coarsely granular versioned products
The framework shall support the labeling scheme, where a product is developed using software versions, but not builds.
The framework shall rely on the fact, that the Kiwi TCMS product will not contain builds other than the defaults in that case.

RQ-4 Fine granular versioned products
The framework shall support the labeling scheme, where a product is developed both using software versions and builds.

RQ-5 Other product versioning schemes
The framework shall not be designed for other labeling schemes, these may or may not work.

RQ-6 Localization
The framework shall support localization of all messages issued by the framework itself.
The framework shall supply and use English locale as default.
The framework shall use the first supported locale found in the following order:
1. Locale referenced by environment variable `ISSAI_LANG`
2. Platform locale
3. English locale

Configuration

RQ-101 Environment variables
The framework shall recognize environment variable `ISSAI_CONFIG_ROOT` for the specification of a custom root directory for configuration files.
The framework shall recognize environment variable `ISSAI_LANG` for the specification of the language to use for messages issued by the framework, independent from the platform setting.

RQ-102 Configuration file format
The framework will accept configuration files in TOML format only.
The framework will allow references to already defined TOML keys to be used within string values, the key must be put between `"${"` and `"}"` and key parts separated by dots.

RQ-103 Configuration root directory location
The framework shall be configurable by files located underneath the configuration root directory.
The framework shall determine the location from environment variable `ISSAI_CONFIG_ROOT`, if present.
The framework shall assume directory `.config/issai` under the user's home directory, if environment variable `ISSAI_CONFIG_ROOT` is not defined.
The framework shall refuse operation, if the configuration root directory doesn't exist.

RQ-104 Configuration root directory structure
The framework shall recognize file `default.toml` in the configuration root directory as master configuration for product independent parameters.
The framework shall treat all subdirectories under the configuration root directory as Issai product names, the product names used in Kiwi TCMS may differ.
The framework shall require a product specific configuration file `product.toml` to be present in each subdirectory.
The framework shall ignore products without valid product specific configuration file.
The framework shall refuse operation, if no products are recognized.

RQ-105 Product related configuration parameters
The framework shall require product specific configuration parameters to be defined in file `product.toml` in the product's subdirectory under configuration root directory.
The framework shall require all product specific configuration parameters to be specified under top level key [`product`].
The framework shall recognize mandatory string parameter `name` for the case sensitive product name as defined in Kiwi TCMS.
The framework shall recognize mandatory string parameter `repository-path` for the root directory containing all product files.
The framework shall recognize optional string parameter `source-path` for the product source code (default `"${product.repository-path}/src"`).
The framework shall recognize optional string parameter `test-data-path` for the product test data files (default `"${product.repository-path}/testdata"`).
The framework shall recognize optional string parameter `nature` for the product nature (one of `"executable"`, `"static-library"` or `"dynamic-library"`, default: `"executable"`).

The framework shall recognize optional array parameter `profiles` for the build profiles supported by the product (default `["debug", "release"]`).
The framework shall recognize optional array parameter `features` for the features supported by the product (default `[]`).
The framework shall allow any other parameter to be defined by the user, but these will not be used by the framework itself.

RQ-106 Build related configuration parameters
The framework shall require build related configuration parameters to be defined in file `product.toml` in the product's subdirectory under configuration root directory or in files named `issai.toml` attached to test cases or plans.
The framework shall require all build configuration parameters to be specified under top level key [`build`].
The framework shall recognize optional boolean parameter `build-before-test` as an indicator whether to build the product before test execution (default: `false`).
The framework shall recognize optional string parameter `target-path` for the target path of compiled sources (default: `"${runner.working-path}/target"`).
The framework shall recognize optional string parameter `profile` for the build profile to use (default: `"debug"`).
The framework shall recognize optional array parameter `features` for the product features to use for build (default: `[]`).

RQ-107 TCMS related configuration parameters
The framework shall require Kiwi TCMS related configuration parameters to be defined in file `product.toml` in the product's subdirectory under configuration root directory or in files named `issai.toml` attached to test cases or plans.
The framework shall require all TCMS related configuration parameters to be specified under top level key [`tcms`].
The framework shall recognize optional string parameter `label-scheme` for the labeling scheme used for the product (one of `"none"`, `"version"` or `"build"`, default: `"version"`).
The framework shall recognize optional string parameter `spec-management` for the management of test specifications (`"auto"` or `"manual"`, default: `"manual"`).
The framework shall recognize optional string parameter `result-management` for the management of test results (`"auto" or "manual"`, default: `"auto"`).
The framework shall recognize optional array parameter `relevant-attachments` for the attachment file names to be downloaded for tests (default: `[]`).
The framework shall recognize optional array parameter `permuting-properties` for the property names triggering repeated execution of test cases for all value items of the property (default: `[]`).

RQ-108 Runner related configuration parameters
The framework shall require test runner related configuration parameters to be defined in file `product.toml` in the product's subdirectory under configuration root directory or in files named `issai.toml` attached to test cases or plans.
The framework shall require all test runner related configuration parameters to be specified under top level key [`runner`].
The framework shall recognize mandatory string parameter `working-path` for the working directory to use for tests.
The framework shall recognize optional string parameter `test-driver-executable` for the name of test driver executable including path (default: parameter not defined).

The framework shall recognize optional string parameter `product-executable` for the name of product executable including path (default: parameter not defined).
The framework shall recognize optional string parameter `locale` for the default locale to use for tests (default: `"en"`).
The framework shall recognize optional string parameter `custom-script-path` for the path of custom scripts for test execution (default: parameter not defined).
The framework shall recognize optional string parameter `custom-module-path` for the path of the python module with custom functions for test execution (default: parameter not defined).
The framework shall recognize optional string parameter `test-init-script` for the name of the custom script to be executed at the beginning of a test (default: parameter not defined).
The framework shall recognize optional string parameter `test-finish-script` for the name of the custom script to be executed at the end of a test (default: parameter not defined).
The framework shall recognize optional string parameter `test-plan-init-script` for the name of the custom script to be executed at the beginning of a test plan (default: parameter not defined).
The framework shall recognize optional string parameter `test-plan-finish-script` for the name of the custom script to be executed at the end of a test plan (default: parameter not defined).
The framework shall recognize optional string parameter `test-case-init-script` for the name of the custom script to be executed at the beginning of a test case (default: parameter not defined).
The framework shall recognize optional string parameter `test-case-finish-script` for the name of the custom script to be executed at the end of a test case (default: parameter not defined).
The framework shall recognize optional string parameter `test-init-function` for the name of the custom python function to be executed at the beginning of a test (default: parameter not defined).
The framework shall recognize optional string parameter `test-finish-function` for the name of the custom python function to be executed at the end of a test (default: parameter not defined).
The framework shall recognize optional string parameter `test-plan-init-function` for the name of the custom python function to be executed at the beginning of a test plan (default: parameter not defined).
The framework shall recognize optional string parameter `test-plan-finish-function` for the name of the custom python function to be executed at the end of a test plan (default: parameter not defined).
The framework shall recognize optional string parameter `test-case-init-function` for the name of the custom python function to be executed at the beginning of a test case (default: parameter not defined).
The framework shall recognize optional string parameter `test-case-finish-function` for the name of the custom python function to be executed at the end of a test case (default: parameter not defined).

RQ-109 Runtime configuration parameter precedence
The framework shall determine the value of a configuration parameter using the first match found in following sequence:
1. parameter is contained in test case attachment file `issai.toml`
2. parameter is contained in test plan attachment file `issai.toml`, for nested test plans starting with the plan containing the test case and then the hierarchy up
3. parameter is contained in product specific configuration file
4. parameter is contained in master configuration file

The framework shall immediately abort operation, if an undefined configuration parameter is referenced.

Test execution

RQ-201 Test plan execution
The framework shall provide the possibility to execute a test plan through script or GUI.
The framework will execute all test cases contained in the test plan itself and all its sibling test plans.

RQ-202 Test case execution
The framework shall provide the possibility to execute a single test case through script or GUI.
The framework shall exclude test cases not having test case status CONFIRMED from execution.
The framework shall exclude test cases not marked as automated from execution.

RQ-203 Offline execution
The framework shall provide the possibility to execute a test without connection to Kiwi TCMS.
The framework will read all necessary data from a file instead in that case.
The framework will not store any results in Kiwi TCMS in that case.

RQ-204 Execution without result storage
The framework shall provide the possibility to run a test, but not store the results in Kiwi TCMS.

RQ-205 Read-only execution
The framework shall provide the possibility to run a test without changing any test specifications in TCMS.
The framework will read all data needed from Kiwi TCMS, but not create test runs or executions.
The framework will skip execution of a test plan in read-only execution, if it would need to create a test run for the plan.
The framework will skip execution of a test case in read-only execution, if it would need to create a test execution for the case.

RQ-206 Simulated execution
The framework shall provide the possibility to simulate a test execution (dry run).
The framework will try to gather all data needed to run a test in a simulated execution.
The framework will log all essential actions to console in a simulated execution.
The framework will not change any data in Kiwi TCMS in a simulated execution.
The framework will not actually execute a test in a simulated execution.
The framework will not produce any result file in a simulated execution.

RQ-207 Test plan tags
The framework shall allow test plans to be excluded from execution by using tags.
The framework will consider tags starting with "os_" and "arch_" as reserved.
The framework shall treat test plans tagged with a name starting with "os_" as operating system specific and skip all test cases under the test plan, if the platform’s operating system doesn’t match the tag.
The framework shall treat test plans tagged with a name starting with "arch_" as machine architecture specific and skip all test cases under the test plan, if the platform’s machine architecture doesn’t match the tag.

RQ-208 Test case tags
The framework shall allow test cases to be excluded from execution by using tags.

The framework will consider tags starting with "os_" and "arch_" as reserved.
The framework shall treat test cases tagged with a name starting with "os_" as operating system specific and skip the test case, if the platform's operating system doesn't match the tag.
The framework shall treat test cases tagged with a name starting with "arch_" as machine architecture specific and skip the test case, if the platform's machine architecture doesn't match the tag.

RQ-209 Permuting properties
The framework shall allow all test cases of a test plan or single test case to be repeatedly executed by using properties.
The framework will recognize such properties by runtime configuration parameter `permuting-properties`.
The framework shall run affected test cases repeatedly with all value items of the property.

RQ-210 TCMS attachments
The framework shall allow test plan or test case attachments to be used for test execution.
The framework shall download execution relevant attachments before test execution to local disk.
The framework shall recognize execution relevant attachments by runtime configuration parameter `tcms.relevant-attachments`.
The framework will always download attachment `issai.toml` for specific runtime configuration, if present.

RQ-211 Custom scripts
The framework shall allow custom scripts to be integrated with the framework.
The framework will require custom scripts to be stored in the directory referenced by configuration parameter `runner.custom-script-path`.
The framework will require references to custom scripts in TCMS test cases to be prefixed with `script:`.
The framework shall allow a custom script to be called at the start of a test, it must be referenced by configuration parameter `runner.test-init-script`.
The framework shall allow a custom script to be called at the end of a test, it must be referenced by configuration parameter `runner.test-finish-script`.
The framework shall allow a custom script to be called at the start of a test plan, it must be referenced by configuration parameter `runner.test-plan-init-script`.
The framework shall allow a custom script to be called at the end of a test plan, it must be referenced by configuration parameter `runner.test-plan-finish-script`.
The framework shall allow a custom script to be called at the start of a test case, it must be referenced by configuration parameter `runner.test-case-init-script`.
The framework shall allow a custom script to be called at the end of a test case, it must be referenced by configuration parameter `runner.test-case-finish-script`.

RQ-212 Custom python functions
The framework shall allow custom python functions to be integrated with the framework.
The framework will require custom python functions callable by the framework to be placed in the python module referenced by configuration parameter `runner.custom-module-path`.
The framework will require references to custom python functions in TCMS test cases to be prefixed with `function:`.
The framework shall allow a custom python function to be called at the start of a test, it must be referenced by configuration parameter `runner.test-init-function`.
The framework shall allow a custom python function to be called at the end of a test, it must be referenced by configuration parameter `runner.test-finish-function`.

The framework shall allow a custom python function to be called at the start of a test plan, it must be referenced by configuration parameter `runner.test-plan-init-function`.
The framework shall allow a custom python function to be called at the end of a test plan, it must be referenced by configuration parameter `runner.test-plan-finish-function`.
The framework shall allow a custom python function to be called at the start of a test case, it must be referenced by configuration parameter `runner.test-case-init-function`.
The framework shall allow a custom python function to be called at the end of a test case, it must be referenced by configuration parameter `runner.test-case-finish-function`.

Auxiliary operations

RQ-301 Product export
The framework shall provide the possibility to export all test specification items of a product into a file.
The framework shall include any metadata referenced by the specification items in the export file.
The framework shall include any result items in the export file.
The framework shall use TOML as file format.

RQ-302 Test plan export
The framework shall provide the possibility to export a test plan into a file.
The framework shall include all data needed for offline execution of the test plan in the export file.
The framework shall exclude result items from the export file.
The framework shall use TOML as file format.

RQ-303 Test case export
The framework shall provide the possibility to export a test case into a file.
The framework shall include all data needed for offline execution of the test case in the export file.
The framework shall exclude result items from the export file.
The framework shall use TOML as file format.

RQ-304 File import
The framework shall provide the possibility to import a file in Kiwi TCMS.
The framework will require the product and referenced users to exist in TCMS prior to the import.
The framework shall require the file to be formatted in TOML.

RQ-305 Test result import
The framework shall provide the possibility to import the result of a previous test run into Kiwi TCMS.

User interface

RQ-401 Command line interface
The framework shall provide a command line interface to execute tests, mainly to support usage within CI tools.
The framework shall provide a command line interface to export data from Kiwi TCMS, mainly to support automated backup.
The framework shall provide a command line interface to import test specification or result data from a file into Kiwi TCMS, mainly to support automated test data creation for subsequent tests.

RQ-402 Graphical user interface
The framework shall provide a GUI to execute tests, export data from Kiwi TCMS and import test specification or result data from file into Kiwi TCMS.
The GUI shall be implemented with PyQt Version 5.

RQ-403 GUI startup options
The GUI shall recognize an option to specify the desired Issai product name.
The GUI shall recognize an option to specify the desired product software version.
The GUI shall ignore the option for the desired product software version, if the option for the product name is not specified.
The GUI shall recognize an option to specify the desired product build.
The GUI shall ignore the option for the desired product build, if the GUI cannot determine the product software version from the other startup options.
The GUI shall recognize an option to specify the desired test plan name.
The GUI shall ignore the option for the desired test plan name, if the GUI cannot determine the product software version from the other startup options.
The GUI shall recognize an option to specify the desired test plan ID.
The GUI shall recognize an option to specify the desired test case summary.
The GUI shall ignore the option for the desired test case summary, if the GUI cannot determine the product software version from the other startup options.
The GUI shall recognize an option to specify the desired test case ID.

RQ-404 GUI product selection
The GUI shall determine all available products and display them in a selection list.
The GUI shall automatically select a product, if a valid product name or test plan/case ID has been specified as startup option or there is only one product available.

RQ-405 GUI product software version selection
The GUI shall determine all available software versions and display them in a selection list, as soon as a product has been selected.
The GUI shall automatically select a software version, if a valid software version or test plan/case ID has been specified as startup option or there is only one software version available.

RQ-406 GUI test plan selection
The GUI shall supply a widget group for the selection of test plans.
The GUI shall supply input fields for filtering test plans for name and ID.
The GUI shall supply a button to fetch test plans from TCMS.
The GUI shall supply a selection list to display the test plan IDs and names from TCMS.
The GUI shall supply a context menu to execute or export the selected test plan in the selection list.

The GUI shall automatically display and select a test plan, if a valid test plan ID has been specified as startup option.

RQ-407 GUI test case selection
The GUI shall supply a widget group for the selection of test cases.
The GUI shall supply input fields for filtering test cases for summary and ID.
The GUI shall supply a button to fetch test cases from TCMS.
The GUI shall supply a selection list to display the test case IDs and summaries from TCMS.
The GUI shall supply a context menu to execute or export the selected test case in the selection list.
The GUI shall automatically display and select a test case, if a valid test case ID has been specified as startup option.

RQ-408 GUI test plan execution
The GUI shall supply a dialog window for the execution of test plans.
The GUI shall supply fields to display ID and name of the selected test plan.
The GUI shall supply a list to select the desired product build.
The GUI shall automatically select a build, if a valid build has been specified as startup option or there is only one product build available.
The GUI shall supply a drop-down list for the labeling scheme, preset according to the setting of configuration parameter `tcms.label-scheme`.
The GUI shall supply a checkbox for automatic creation of test runs and executions if needed, preset according to the setting of configuration parameter `tcms.spec-management`.
The GUI shall supply a checkbox for ignoring test results, preset according to the setting of configuration parameter `tcms.result-management`.
The GUI shall supply an initially unchecked checkbox for a dry run.
The GUI shall supply an initially unchecked checkbox for a read-only run.
The GUI shall supply buttons to start the test plan execution or to close the window.

RQ-409 GUI test case execution
The GUI shall supply a dialog window for the execution of test cases.
The GUI shall supply fields to display ID and summary of the selected test case.
The GUI shall supply a list to select the desired product build.
The GUI shall automatically select a build, if a valid build has been specified as startup option or there is only one product build available.
The GUI shall supply a drop-down list for the labeling scheme, preset according to the setting of configuration parameter `tcms.label-scheme`.
The GUI shall supply a checkbox for automatic creation of a test execution if needed, preset according to the setting of configuration parameter `tcms.spec-management`.
The GUI shall supply a checkbox for ignoring test results, preset according to the setting of configuration parameter `tcms.result-management`.
The GUI shall supply an initially unchecked checkbox for a dry run.
The GUI shall supply an initially unchecked checkbox for a read-only run.
The GUI shall supply buttons to start the test case execution or to close the window.

RQ-410 GUI execution status dialog
The GUI shall supply a dialog window to show the execution status of a running test.
The GUI shall show the item type in the dialog window title (Test plan or Test case).
The GUI shall show a status bar to display test progress.
The GUI shall show a scrollable text field for messages.
The GUI shall display overall result, statistics and major problems in the message field, when the test has finished.
The GUI shall provide a button to abort test execution and close the window.

RQ-411 GUI test plan export
The GUI shall supply a dialog window for the export of test plans.
The GUI shall supply fields to display ID and name of the selected test plan.
The GUI shall supply a list to select the desired product build.
The GUI shall automatically select a build, if a valid build has been specified as startup option or there is only one product build available.
The GUI shall not require a build to be selected.
The GUI shall supply an initially checked checkbox to include sibling test plans in the export.
The GUI shall supply an initially checked checkbox to include test runs and executions in the export.
The GUI shall supply a field to select the output directory, preset with the value of configuration parameter `runner.working-path`.
The GUI shall supply a field to select the output file name, preset with the value `testplan_<id>.toml` (where <ID> denotes the test plan ID from TCMS).
The GUI shall supply buttons to start the test plan export or to close the window.

RQ-412 GUI test case export
The GUI shall supply a dialog window for the export of test cases.
The GUI shall supply fields to display ID and name of the selected test case.
The GUI shall supply a list to select the desired product build.
The GUI shall automatically select a build, if a valid build has been specified as startup option or there is only one product build available.
The GUI shall not require a build to be selected.
The GUI shall supply an initially checked checkbox to include test executions in the export.
The GUI shall supply a field to select the output directory, preset with the value of configuration parameter `runner.working-path`.
The GUI shall supply a field to select the output file name, preset with the value `testcase_<id>.toml` (where <ID> denotes the test case ID from TCMS).
The GUI shall supply buttons to start the test case export or to close the window.

RQ-413 GUI product export
The GUI shall supply a “File/Export product” menu item to export an entire product.

RQ-414 GUI export status dialog
The GUI shall supply a dialog window to show the status of a running export.
The GUI shall show the item type in the dialog window title (Product, Test plan or Test case).
The GUI shall show a status bar to display export progress.
The GUI shall show a scrollable text field for messages.
The GUI shall display overall result and major problems in the message field, when the export has finished.
The GUI shall provide a button to abort the export and close the window.

RQ-415 GUI test specification data import
The GUI shall supply a “File/Import/Test specification” menu item to import data from a file.

RQ-416 GUI test result import
The GUI shall supply a “File/Import/Test result” menu item to import a test result from a file.

RQ-417 GUI import status dialog
The GUI shall supply a dialog window to show the status of a running import.
The GUI shall show a status bar to display import progress.
The GUI shall show a scrollable text field for messages.

The GUI shall display overall result and major problems in the message field, when the import has finished.
The GUI shall provide a button to abort the import and close the window.

RQ-418 GUI configuration editor
The GUI shall supply an editor to edit TCMS XML-RPC credential file, and Issai master and product specific configuration files.
The GUI shall supply a “File/Settings” menu item to access the configuration editor.

Glossary

Labeling scheme

Kiwi TCMS uses software versions and builds to be used for the labeling of the product. Issai offers three labeling schemes which control the creation resp. selection of builds, test runs and test executions.
Scheme ***none*** will use the software version and build, which are automatically created by Kiwi TCMS for every product.
Scheme ***version*** will use the build, which is automatically created by Kiwi TCMS for every software version.
Scheme ***build*** will use the build name specified by the user.

Product

A product is the piece of software, that shall be tested.

Product build

A build is a name assigned to the state of a software on a fine grained level, usually for an incremental build. A build is always specific to a certain software version.

Result management

Kiwi TCMS uses test runs and test executions to store the test results. Issai can be configured for two modes regarding test results:
In ***automatic*** mode, Issai will store results in the appropriate test runs and executions by itself and even create the test runs and executions, if missing.
In ***manual*** mode, Issai will keep all results in local disk files, but not update Kiwi TCMS.

Software version

A software version is a name assigned to the state of a software on a higher level, usually for a certain set of features implemented by that version.

Specification management	Kiwi TCMS requires test runs and test executions to be used for running tests. Issai can be configured for two modes regarding test runs and test executions: In ***automatic*** mode, Issai will create the test runs from test plans and test executions from test cases, if they don't exist for the build. In ***manual*** mode, Issai will not create missing test runs or test executions, the test will fail if they don't exist.
TCMS	Test case management system.
Test case	In Kiwi TCMS, a test case describes how to perform a specific scenario. Test cases are specific to a certain product software version. Since tests are always executed against a product build, Kiwi TCMS requires test runs to be used for execution. In Issai, a test case holds informations from both Kiwi TCMS test cases and test executions.
Test execution	In Kiwi TCMS, a test execution is more or less a clone of a test case plus attributes to store execution results. In contrast to a test case, a test execution is specific for a product build. Issai doesn't use separate test execution items, Issai always runs tests for a specific product build and merges test execution data into the test case.
Test plan	In Kiwi TCMS, a test plan is a high-level container object which is used to describe testing activities. It holds a list of test cases and may also contain other test plans. In Issai, a test plan holds informations from both Kiwi TCMS test plans and test runs.
Test run	In Kiwi TCMS, a test run is more or less a clone of a test plan plus attributes to store execution results. In contrast to a test plan, a test run is specific for a product build. Issai doesn't use separate test run items, Issai always runs tests for a specific product build and merges test run data into the test plan.

Test specification item	General term for a Kiwi TCMS item specifying a test, i.e. a test plan, test run, test case or test execution.
TOML	Tom's obvious minimal language, a configuration file format both easy to read by humans and process by machines.