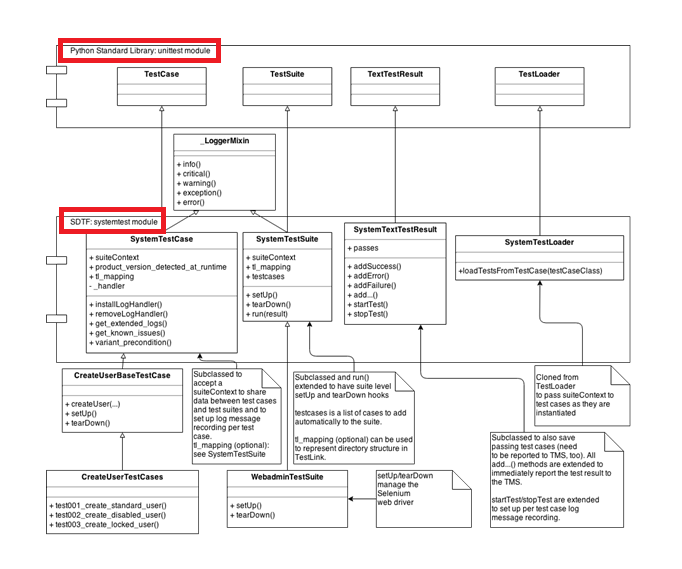
# Samenvatting van het SDTF-document

SDTF is een proof of concept van een unit testing framework, geschreven in python en gebaseerd op de python unittest library.

Er zijn afwijkingen van de standaard unittest library nodig om ervoor te zorgen dat een object informatie kan doorgeven tussen test cases en test suites, en de test resultaten terug te koppelen naar het test management system, in ons geval TestLink.

De link tussen de implementatie van den test case en de test management system wordt gedefinieerd in de docstring van de test cases. Op deze moment is dit een samentrekking van een ID en een korte omschrijving. Dit moet overeenkomen met de entry in de test management system.



Deze figuur geeft de relatie weer tussen de classes en de unittest classes waarvan ze zijn afgeleid

## SystemTest

De SystemTest module bevat alle toevoegingen tot de standaard unittest framework en uitbreidingen op de standaardfuncties zoals:

* Suite fixtures: een suite setUp/tearDown method(?) wordt maar ene keer gecalled per test suite
* Mogelijkheid om een context(?) door te geven naar de TestCase
* Een uitgebreide TextTestResult
* Een aangepaste TestLoader

Dit is gedefinieerd in de framework/core/systemtest.py.

### SystemTestCase

SystemTestCase is een subclass van de unittest.TestCase class, dit voegt een attribuut toe en zorgt ervoor dat het log berichten kan opvangen tijdens het uitvoeren van de testcase. De log berichten worden opgenomen door een custom subclass van de logging.Handler class.

*“All test cases must (indirectly) inherit from SystemTestCase and (for future extensibility) make sure not to override the SystemTestCase setUp and tearDown methods (inherited from unittest.TestCase): they may be extended, but the subclass’ methods should call their parent’s methods.”* => Hier ben ik ni mee

In tegenstelling tot unittest.TestCase (uit de standaardlib) heeft SystemTestCase een extra initialisatie sap: het is soms nodig om elke instance (een voor elke test method in een test case class) (lichtjes) anders te initialiseren.

*“However, a test case’s \_\_init\_\_ method’s signature is fixed and the \_\_init\_\_ method is (in the source code) far away from the different test methods of the same class. The variant\_precondition method is run right after the \_\_init\_\_ method with the method name as parameter and all keyword parameters provided to the with\_variant\_precondition decorator. The default implementation of variant\_precondition turns the keyword parameters into instance attributes (for more details, see* [*doc/variant\_precondition.org*](https://git.vasco.com/wqa/sdtf/raw/master/doc/variant_precondition.org)*, first committed in* [*a5298381e83c5ca1*](https://git.vasco.com/wqa/sdtf/blob/a5298381e83c5ca1cc0eef5173095e95fc1e5966/doc/variant_precondition.org)*).”* => Snap ik ni helemaal, relevant voor de samenvatting?

De get\_known\_issues wordt gecalled voor een test case in alle gevallen die niet als een succes worden gezien. De return waarden worden dan gebruikt om TestLink te updaten met de problemen die zich hebben voorgedaan (via ID’s). Let er wel op

“Care must be taken when implementing get\_known\_issues not to attribute issue IDs when other issues are present for which no issue ID is known.” => da gaat relevant zijn, maar ik begrijp ni echt wat hier wordt bedoelt. Is het dat ge moet opletten dat ge dingen gaat missen als er voor bepaalde issues geen ID bestaat?

*Verder kan ik ook niet echt kort uitleggen wat dit is/doet tbh*

### SystemTestSuite

Subclass van unittest.TestSuite, voegt een setUp en een tearDown method toe en een suiteContext attribuut. De run method wordt uitgebreid om de setUp en tearDown methods te callen van de suite.

De test suites kunnen gebruikt worden om de directory structuur van de test case organisatie in Testlink voor te stellen; als de suite class een lid definieert als tl\_mapping, dan dan is ‘dit’ onthouden op de moment dat de test wordt uitgevoerd. Deze informatie wordt gebruikt bij het rapporteren naar TestLink om onderscheid te maken tussen test cases met dezelfde naam, maar verschillende mappen. Wanneer dit is gedefinieerd gaan de testcase leden in een, wacht hoor

*“When defined, the testcases member in a suite is used to automatically add instantiate test cases and add them to the suite at suite initialization time”* => dees snap ik niet goed

### SystemTextTestResult

Subclass van TextTestResult, voegt ook de geslaagde test cases toe aan het testResulat object, omdat de geslaagde test ook moeten worden geregistreerd in het Test Management System.

Alle addXXX methods zijn uitgebreid zodat ze onmiddellijk test case resultaten registreren in het Test Management System. Voor TestLink is het mogelijk om de build ID automatisch aan te maken in TestLink voor het geselecteerde testplan, indien mogelijk. Maar enkel als de nieuwe build ID bevestigd is door het uitvoeren van de test case.

### SystemTestLoader

De unittest framework werkt op het idee van een suite dat uitgevoerd wordt door een test runner (unittest.testrunner). Het builden van een test suite wordt geautomatiseerd met een testLoader instance; gebaseerd op conventies omtrent naamgeving

## Logging

Voor logging gebruiken we de standaard Python library. Het logger object dat wordt gebruikt is opgeslagen in de config data in de suiteContext en een subset van logging methods is ook gedefinieerd in de test cases en test suites.

Tijdens het uitvoeren van een test case wordt elk bericht dat geprint wordt in de standaard output gecaptured en naar de logger gestuurd. Maar het voordeel van de logger rechtstreeks te gebruiken is dat filenaam en lijnnummer ook wordt opgenomen, zodat je weet van waar het bericht afkomstig is.

## Configuratie

De configuratie wordt geladen uit 2 bronnen, de ene is centraal in het framework, de ander is gelinkt aan het uit te voeren test project door zijn plek in het file system. De framework/core/config.py module definieert de configuratie instellingen die gemeenschappelijk zijn voor alle tests. Nu zijn dit:

* TestLink login info (URL en DevKey)
* Virtualisatie info (virtualisatie systeem URL, username en password,…)
* De centrale logger instantie

De SDTFConfiguration zoekt naar twee optionele parameters: een filename en een arbitraire Python waarde. Deze waarden worden gebruikt om de configuratie te bouwen die wordt toegevoegd aan het configuratie object. Het object zoekt naar een SDTF\_config.py file en stopt bij de eerste hit. (het begint met zoeken in de directory waar de file inzit die hij meekreeg als input parameter, en loopt dan omhoog in de hiërarchie tot aan de root)

## Main: program execution

De code om de config te laden, een test suite te bouwen, runnen en rapporteren naar de Test Management System is geconcentreerd in de Main class gedefinieerd in framework/core/main.py. Deze class kan worden gebruikt zonder parameters aangezien er logische defaults worden voorzien.

Deze class zorgt ook voor de instantiering van de configuration class en het parsen van command line argumenten. “This will give all test scripts a common command line interface:”

<script name> <build> <platform> [suite] [--config.sub.var=value …]

Build en platform info zijn nodig om de link te maken tussen de test resultaten en de TMS. Meer waarden kunnen gespecifieerd worden aan het einde van de command line, zie “special parameters”

### Speciale parameters

* **--help:** If `--help’ is present anywhere in the command line arguments, a help text is printed and nothing is done. The [docstring](http://freeradius.org/radiusd/man/radtest.html) of the script is used in the help text.
* **--dump\_fqtcn:** The optional --dump\_fqtcn parameter will print a list of all testcases in the provided suite along with their (dynamic) index. After printing the list, nothing is done.
* **--dump\_fqsn:** The optional --dump\_fqsn parameter will print a list of all subsuites in the provided suite along with their (dynamic) index. After printing the list, nothing is done.
* **--tests\_to\_run**: The optional --tests\_to\_run parameter allows you to run a specific subset of tests inside a suite. You can separate testcase indexes by a ',' or by a '-' if you want to run a range of tests. Example :
  + --tests\_to\_run="1,2-5,3,10"
    - Will run testcases : 1, 2, 3, 4, 5 and 10
* **--suites\_to\_run**: The optional --suites\_to\_run parameter allows you to run specific subsuites of a suite. Similar usage to the --tests\_to\_run parameter. Indexes out of bound are ignored.

# Vragen

~~Wat is het een test case precies? En wat is het verschil met een test suite?~~

~~Wat is de functie van een test management system, en waarom moet er een link zijn met de test cases?~~

Die system test module, wat moet ik mij daar bij voorstellen, is dat gewoon een verzameling van classes en functies/methods?

“Ability to pass a context to the TestCase” wat houdt da juist in?

“Suite fixtures: a suite setUp / tearDown method” wat doen die methods?

Als er prereq’s zijn voor het uitvoeren van een testcase dan zorgt de setup ervoor dat deze in orde zijn. Bijvoorbeeld: als er een testcase is voor users te deleten, dan gaat de setup users aanmaken hiervoor.

VirtualEnv?