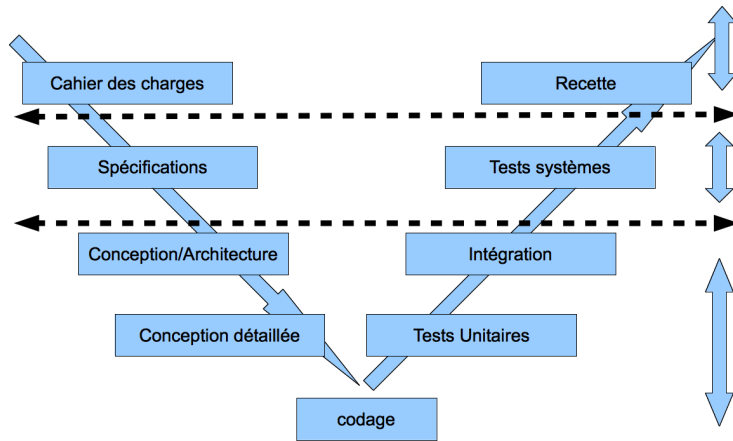




Cours 5 : l'automatisation

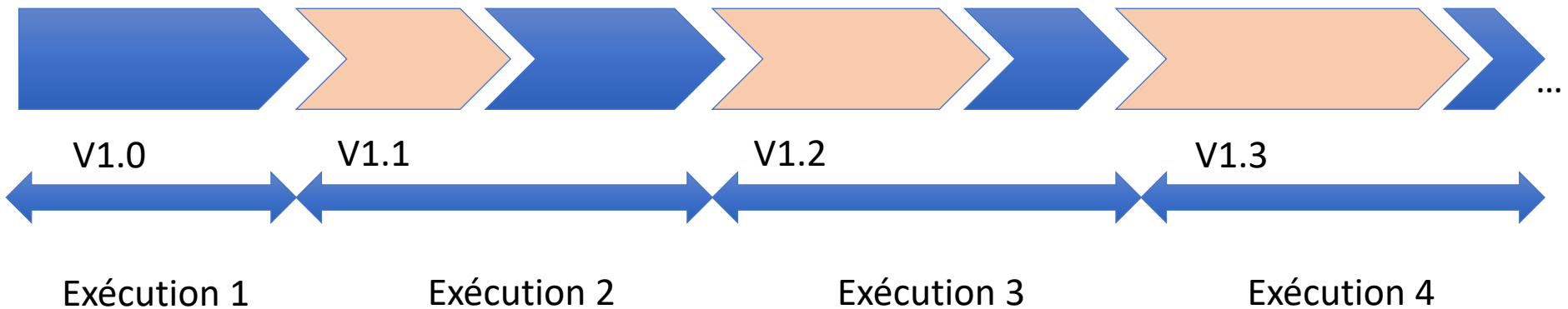
la phase d'exécution



C'est trop long,
le client attend
la version



-  Re-test et non régression
-  Test des nouvelles fonctionnalités



Décision d'automatiser

→ Souvent prise pour réduire en temps d'exécution la non-régression.

Ex: fréquence des versions

→ Nombre de tests ingérables manuellement.

Ex : combinatoires et configurations

→ Fiabilisation des tests.

Vérifications manuelles pénibles.

→ Réduction des coûts

Critères d'automatisation

- Cas de test éligibles à l'automatisation
- Cas de tests importants, critiques.
- Cas de tests répétitifs.
- Cas de test non fiable manuellement.
 - Lassitude du testeur
 - Risque d'erreur important
- Cas de test éligible à la non régression/recevabilité
 - rejoués souvent

Automatisation

Niveaux de test

Qu'automatise-t-on?

- Tests Unitaires
 - Simples à automatiser
 - Nécessitent des bouchons/mocks
 - Facile à intégrer dans un outil de CI
- Tests API, WEB SERVICES (systèmes)
 - Complexes
 - Prérequis
 - Nécessitent un outillage dédié
- Tests d'interface
 - Plus complexes, maintenabilité
 - Prérequis
 - Nécessitent un outillage dédié

Les Tests d'API

ACTIONS de test:

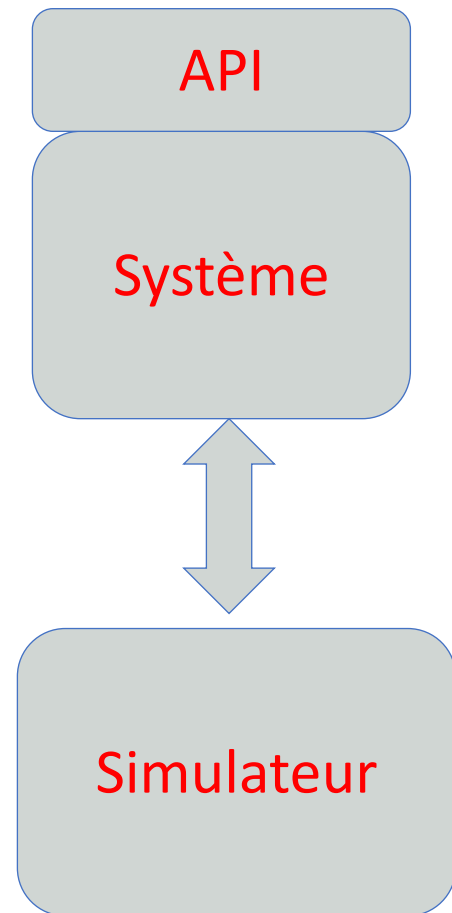
Par programmation

Le Monde extérieur

1. Programmation
2. Un outil + ou – complexe

Contrôles:

1. Vérification de sortie
2. Vérification de protocole
3. Fichiers
4. Base de données
5. Etat de système
6. Actions (mail envoyé)



Les tests d'API

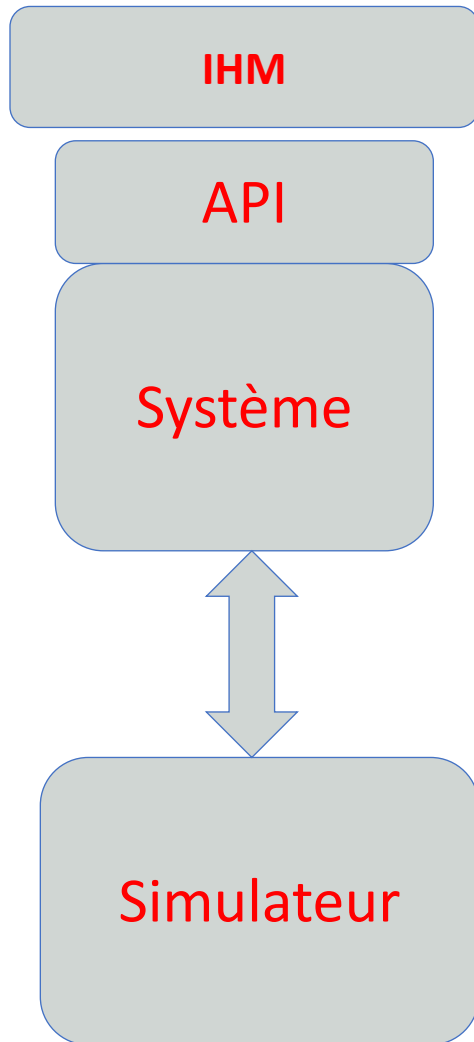
- API:

- Même framework de test que les tests unitaires (partagé entre dev et test),
- Un framework dédié (spécifique à l'équipe de test),
- Outils maison,
- Des langages de programmation type script (python, jython ...).

- WEB Service/API Rest

- Jmeter,
- SOAP UI,
- Etc...

Les Tests d'IHMs



Tester des API nécessite des compétences en programmation.

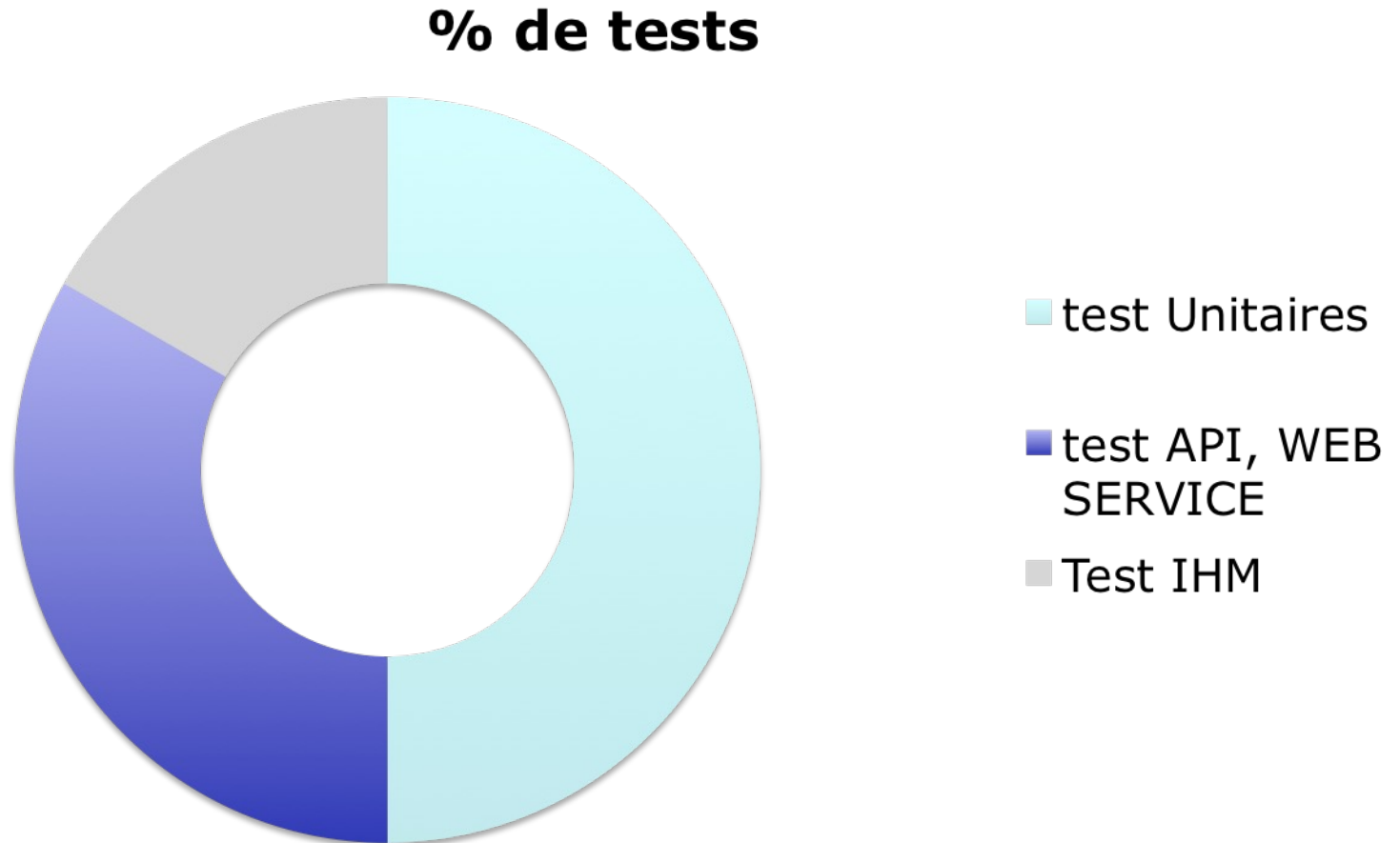
Tester des IHM nécessite

1. Peu de compétences en programmation.
2. Des IHMs stables.
3. Des outils performants pour simuler les actions souris et claviers et pour capturer des éléments graphiques pour des vérifications.

Les Tests d'IHM

- Outils qui manipulent les objets graphiques (mono techno ou multi techno).
 - Auto IT (IHM Windows, langage dédié).
 - Selenium (Web, java, python, etc.).
 - QTP/UFT (multi techno, VB).
- Outils qui se basent sur la position ou l'image
Sikuli (multi techno, langage dédié).
- Tous ces outils possèdent un enregistreur d'actions.

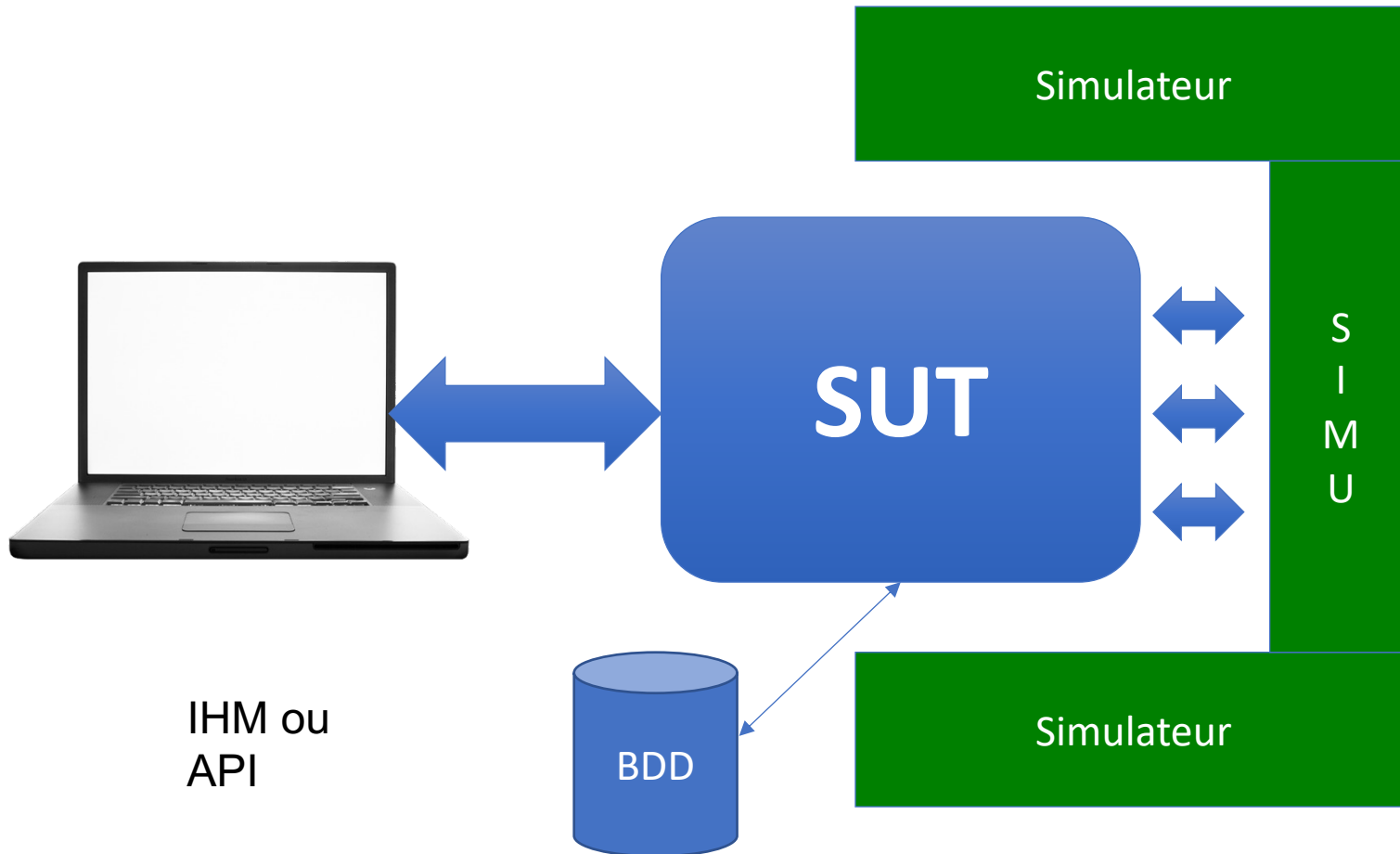
Répartition des tests par type (conseillée)



Automatisation

L'environnement de test

LE SYSTEME SOUS TEST



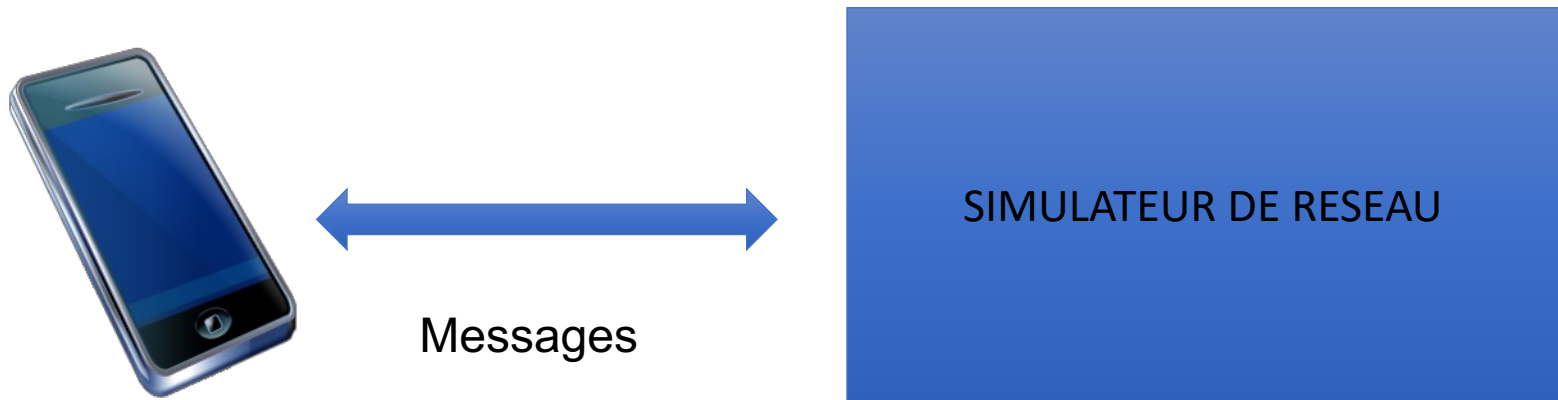
LE SYSTEME SOUS TEST

Pour automatiser nous avons besoin:

- De piloter l'application via API ou IHM avec l'outillage adéquat.
- Un outillage qui permet d'effectuer les contrôles.
 - Même outillage que pour piloter l'application.
 - Des outils spécifiques (comparaison de fichiers, protocoles ...).
- De simulateurs qui approximent les systèmes tiers auxquels l'application sous test est connectée.
 - Pilotage, spécification du comportement ...
- Gérer le contenu de la base de données.

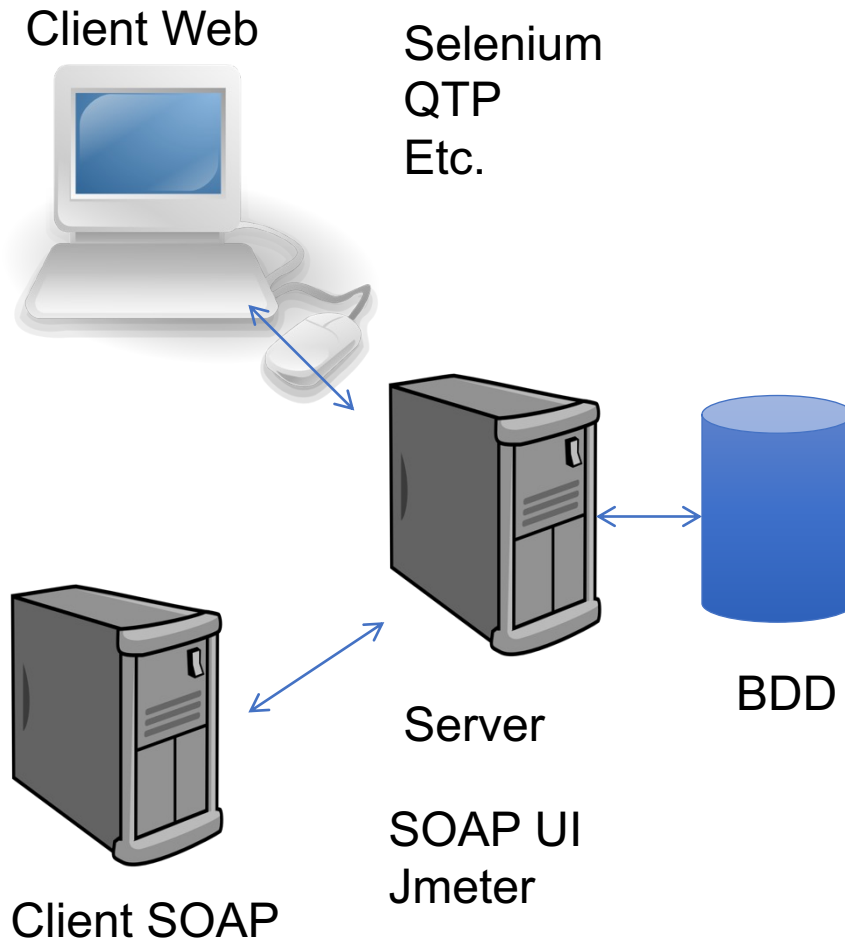
Environnement : le simulateur

- EX: test d'un nouveau téléphone mobile



- Le Simulateur émet et reçoit des messages conformément au protocole GSM.

Exemple: site web



• SCENARIO DE TEST TYPE

1. Mise en place des données de test.
2. Utilisation d'une fct via un formulaire ou une requête SOAP.
3. Vérification en BDD de la mise à jour des informations.
4. Utilisation d'une autre page pour vérifier les infos ou appel à une requête SOAP.
5. Vérification par rapport à une référence.
6. Mise en forme des résultats de test.
7. Stockage des logs de test.

Automatisation

Les différentes approches

Le Projet d'automatisation de test

Définition de la stratégie d'automatisation.

Sélection des tests à automatiser.

Mise en place du framework et codage des tests.

Génération des rapports.

Sauvegarde des scripts de tests et de l'environnement.

Analyse et conception

Implémentation et exécution

Evaluation et information

Clôture

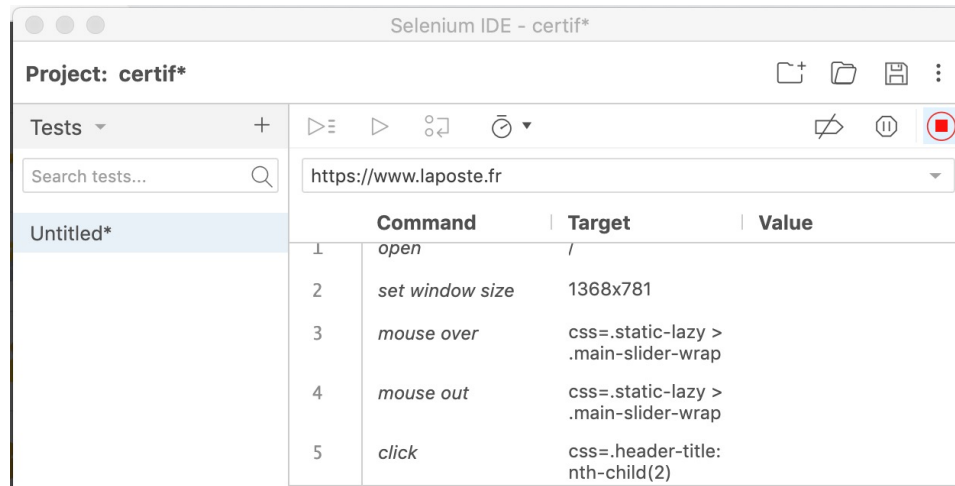
Planification et contrôle

Approche capture-rejeu

- les outils capturent les interactions avec le SUT, les entrées et les sorties peuvent également être capturées.
- Néanmoins les vérifications peuvent être
 - Manuelles, le testeur regarde
 - Complètes : toutes les sorties ont été enregistrées
 - Exactes : toutes les sorties ont été enregistrées selon le niveau de détail sélectionné
 - Checkpoint : le testeur sélectionne les sorties à enregistrer

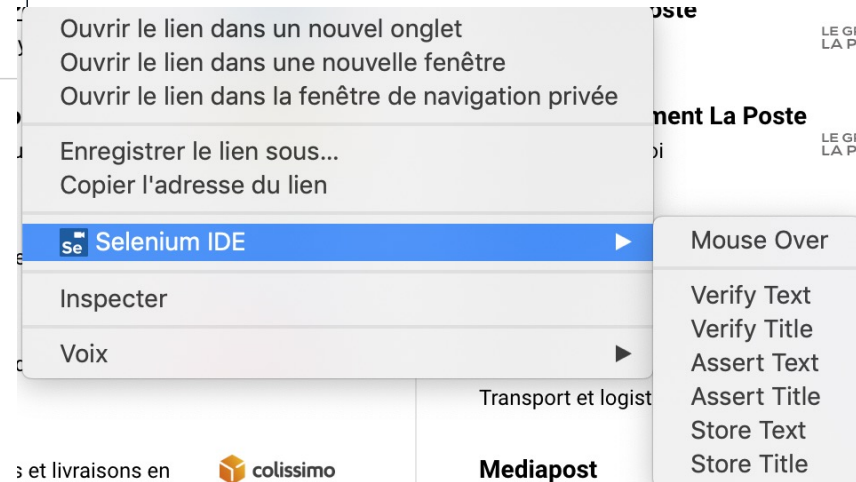
Approche capture-rejeu

- Capture/Rejeu : exemple d'outil: selenium



Toutes les actions utilisateurs sont enregistrées

Le testeur peut sélectionner le contrôle qu'il veut mettre en place.



Approche capture-rejeu

- Capture/Rejeu

Avantages

Facile à mettre en oeuvre

inconvénients

Coût de maintenance important du à la fragilité du script (trop proche de l'implémentation).

Scripting linéaire

- Les procédures de test sont connus mais pas forcément formalisées.
- L'outil enregistre la séquence d'actions quand le testeur exécute manuellement le test
- Les scripts sont ensuite édités pour rajouter des commentaires ou des vérifications.
- Les scripts peuvent être rejoués à l'infini.
- Ces scripts coûtent chers en maintenance, car sensible aux changements.

Scripting linéaire

- Exemple

```
_navigateTo("http://qualifiez.fr/examples/Selenium");  
_setValue(_textbox(0), "dcfsdc");  
  
_setValue(_textbox("maClasse[0]"), "cdfsdcsd");  
_click(_submit(0));  
_assertEqual("Time sheet", _getText(_heading1("Time sheet")));  
  
_assertEqual("", _getValue(_textbox("pw")));  
_assertExists(_paragraph("login / pw invalide"));  
  
_click(_submit("connecter"));  
_click(_paragraph("login / pw invalide"));  
_click(_submit("connecter"));  
_click(_paragraph("login :"));
```

Scripting linéaire

Avantages

Facile à mettre en œuvre

Nécessite peu de compétence en programmation

inconvénients

Coût de développement élevé car pas d'optimisation.

Nécessité de connaître le langage de script qui peut-être propriétaire.

Coût de maintenance important du à la fragilité du script (trop proche de l'implémentation).

Scripting structuré

- Utilisation de la réutilisabilité pour créer des bibliothèques d'actions et de contrôles.
- Exemple:

```
_include("C:/SQUASH-TA/sahi_v50_20141105/userdata/scripts/global_include.sah")
function doTest($name,$pwd)
{
  _navigateTo("http://dgu-PC/timesheet");
  seLogger($name,$pwd)
  controle("login / pw invalide")
}
var $data = _readCSVFile("data.csv");
_dataDrive(doTest, $data)
```

Scripting structuré

- Scripting structuré: exemple

```
@Test
public void testRecherche1() throws Exception {
    pageRecherche p2;
    p2 = p1.rechercher("JUPE");
    assertEquals("1 résultat a été trouvé.", p2.res());
}

@Test
public void testRecherche2() throws Exception {
    pageRecherche p2;
    p2 = p1.rechercher("ROBES");
    assertEquals(7, p2.nbMeilleursVentes());
}
```

Scripting structuré

- Scripting structuré

Avantages

Coût de maintenance réduit.

Coût de développement réduit.

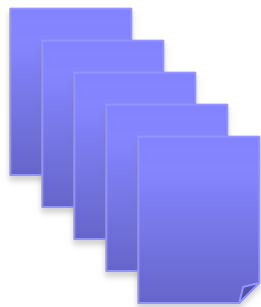
inconvénients

Investissement initial important.

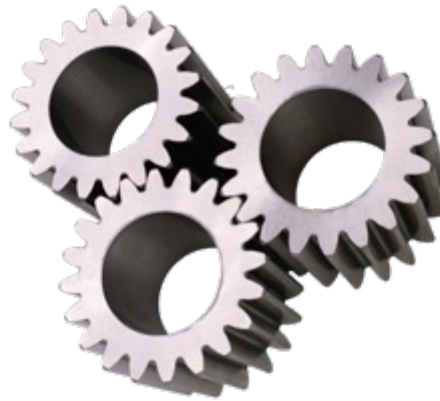
Compétences en programmation requises.

Tests pilotés par les données

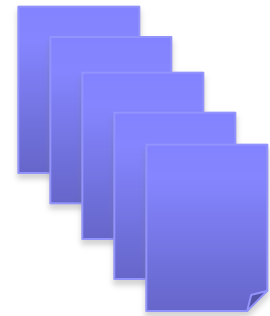
- Basé sur le scripting structuré
- Les données sont gérées via des fichiers



Jeux d'essais :
Entrées



Script de contrôle

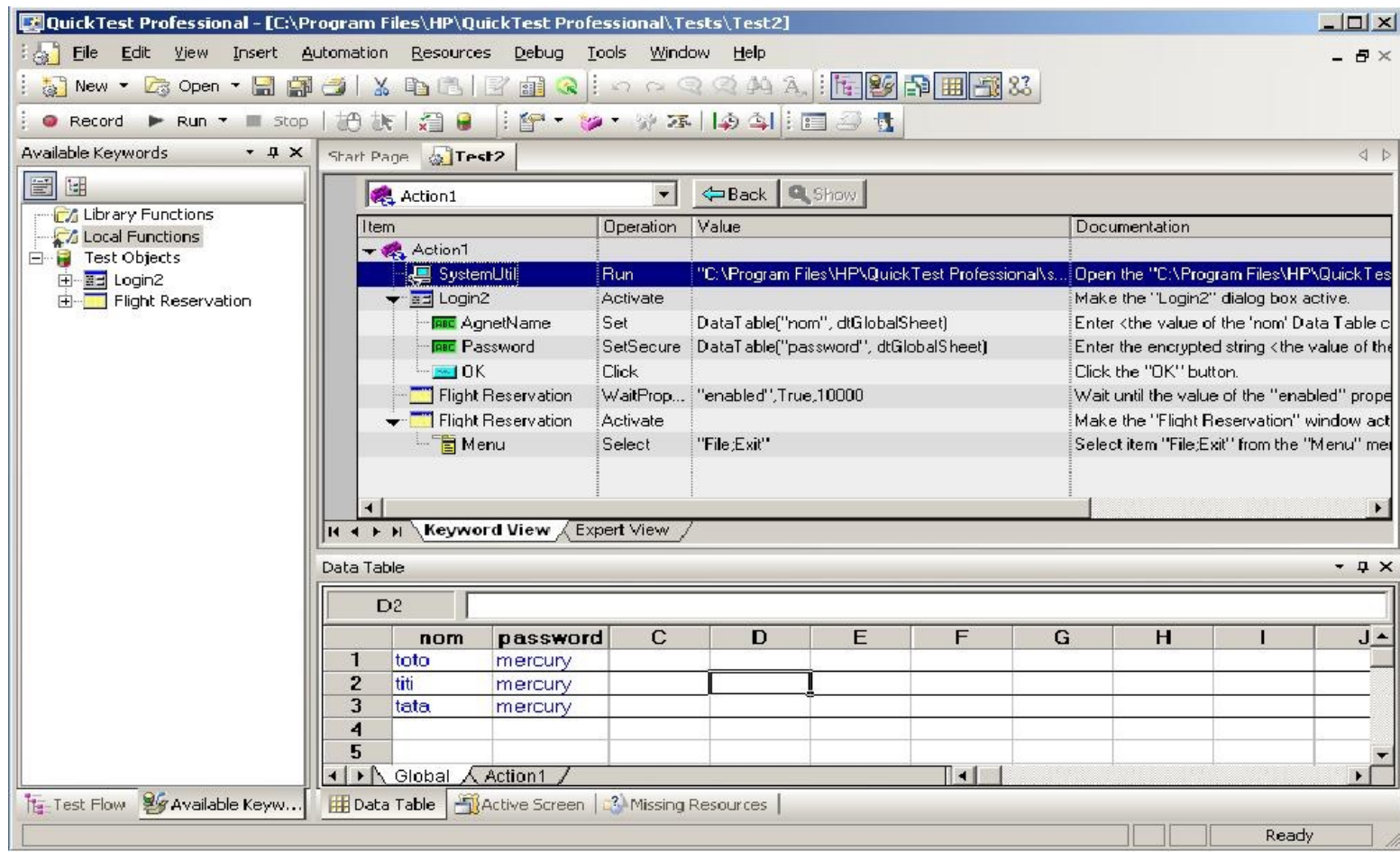


Référence:
Sorties

Fichier de
données, xml,
excel ...

Tests pilotés par les données

- example



Tests pilotés par les données

- Test piloté par les données

Avantages

Augmentation de la profondeur de tests (variation d'un même test)

L'analyste de test peut facilement créer des tests via les jeux de données

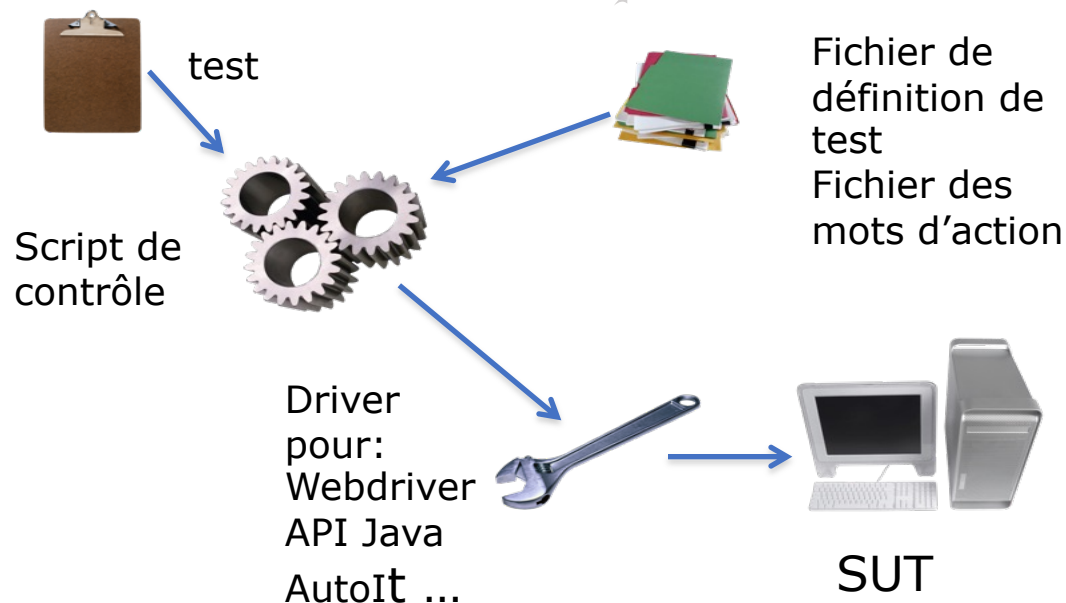
inconvénients

Gestion des fichiers de données via le TAS

Oubli des cas de tests négatifs (cas d'erreur)

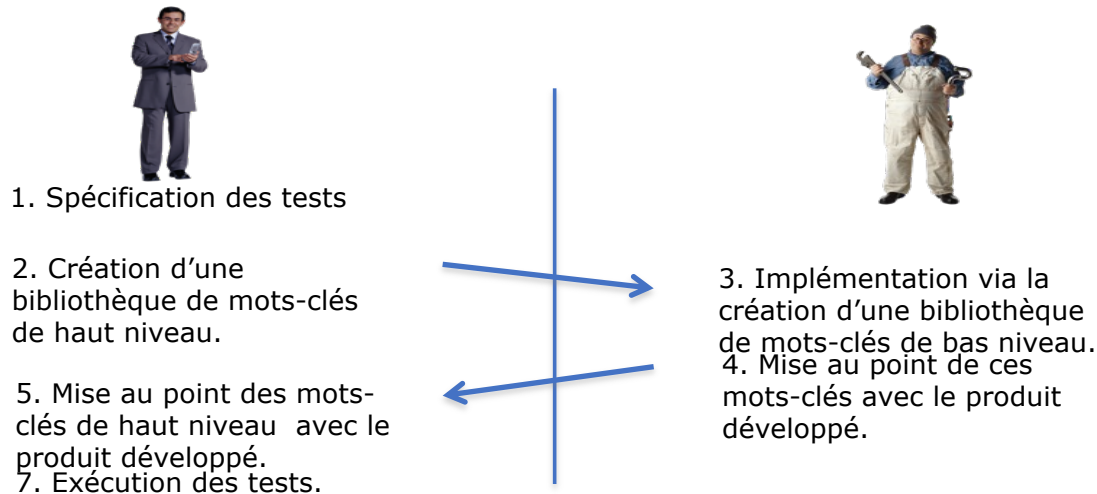
Tests pilotés par les mots-clés

- Tests pilotés par les mots clés
 - Basé sur le test piloté par les données
 - Les fichiers de données contiennent les définitions d'actions + 1 seul script de contrôle.



Tests pilotés par les mots-clés

- Tests pilotés par les mots clés
 - Les mots-clés représentent des actions de haut niveau orientés métier
 - Les analystes de test définissent ces mots-clés
 - Un mot clé représente une suite d'actions basiques



Tests pilotés par les mots clés

- Test piloté par les mots clés

Avantages

Ajout de nouveaux tests peu couteux
(une fois le script de contrôle écrit +
mots clés)
Scripting possible par analyste de test
Actions de haut niveau compréhensible
par tous
Facile à maintenir

inconvénients

L'Implémentation des mots clés reste
une tâche technique
Faire les bons choix des mots clés

Tests pilotés par les processus

- Tests pilotés par les processus
 - Basé sur le test piloté par les mots clés
 - On implémente les cas d'utilisation métier
 - Ex : passer la commande puis vérifier que la commande est passée.

Tests pilotés par les processus

Avantages

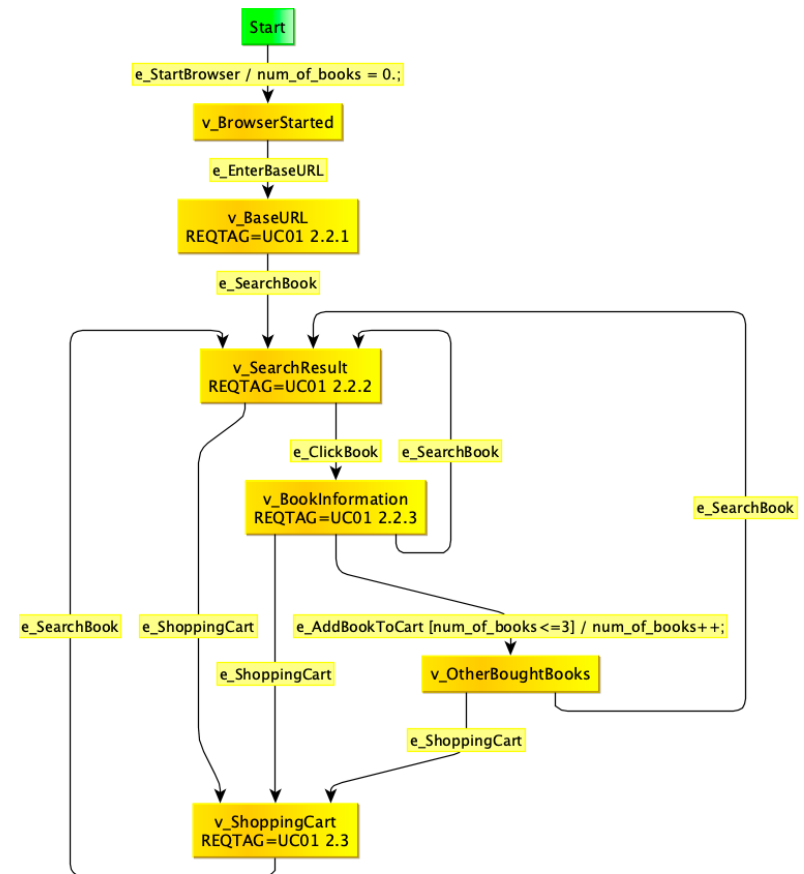
Utilisation des scénarios de cas d'utilisation
Bibliothèques de test dédiées contenant les étapes détaillées

inconvénients

Plus compliqué pour l'analyste de test technique
Vérifier que les processus et mots clés sont correctement implémentés.

Tests pilotés par les modèles

- Tests pilotés par les modèles
 - Les scripts de test sont générés à partir de modèles
 - Indépendants de la technologies de Scripting



Tests pilotés par les modèles

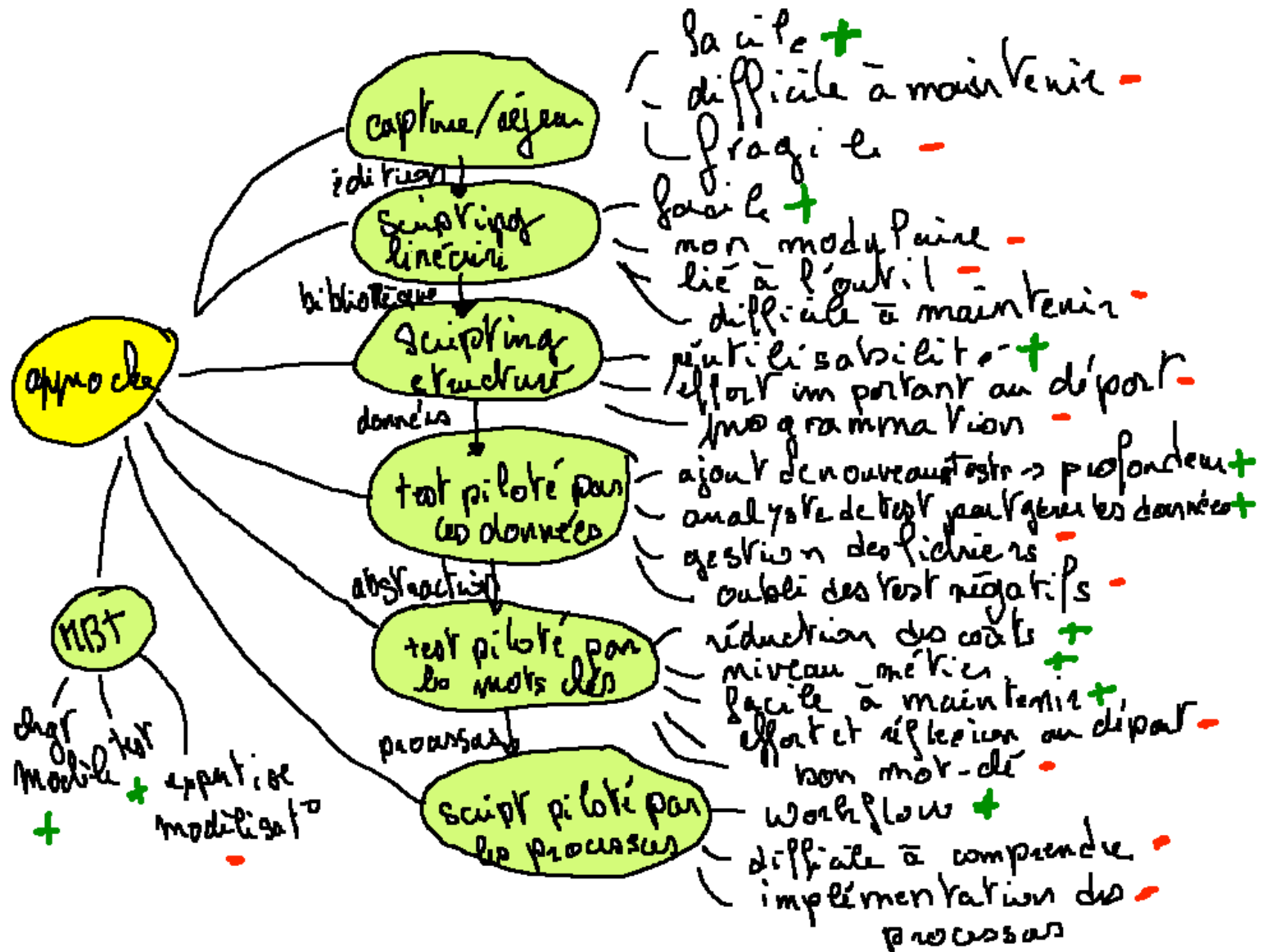
Avantages

L'analyste se concentre sur le test en terme de en termes de logique métier, données, scénarios, configurations.
Les scripts sont générés automatiquement indépendamment de la technologie
En cas d'évolution on ne change que le modèle, la génération de script étant automatisée.

inconvénients

Capacité d'abstraction pour concevoir le modèle (expertise).
Peu d'outils sur le marché.
Les modèles doivent être vérifiés et consolidés.

Approches d'automatisation



Mots-clefs

Une présentation rapide

DEFINITION

- Tests déterminés par mots clé:
- une technique de script utilisant des fichiers de données qui contiennent outre les données de test et les résultats attendus, des mots clé liés à l'application à tester. Ces mots clé sont interprétés par des scripts de support spécifiques, appelés par le script de contrôle du test.
- (définition tirée du glossaire de l'istqb)

DEFINITION

- Technique d'automatisation qui utilise les mots clés.
- Mot-clé: représente une action ou une vérification qui va être appelé par le script d'automatisation.
- Interprétation

ETAPES	
Action 1	Contrôle 1
Action 2	Contrôle 2



Web App



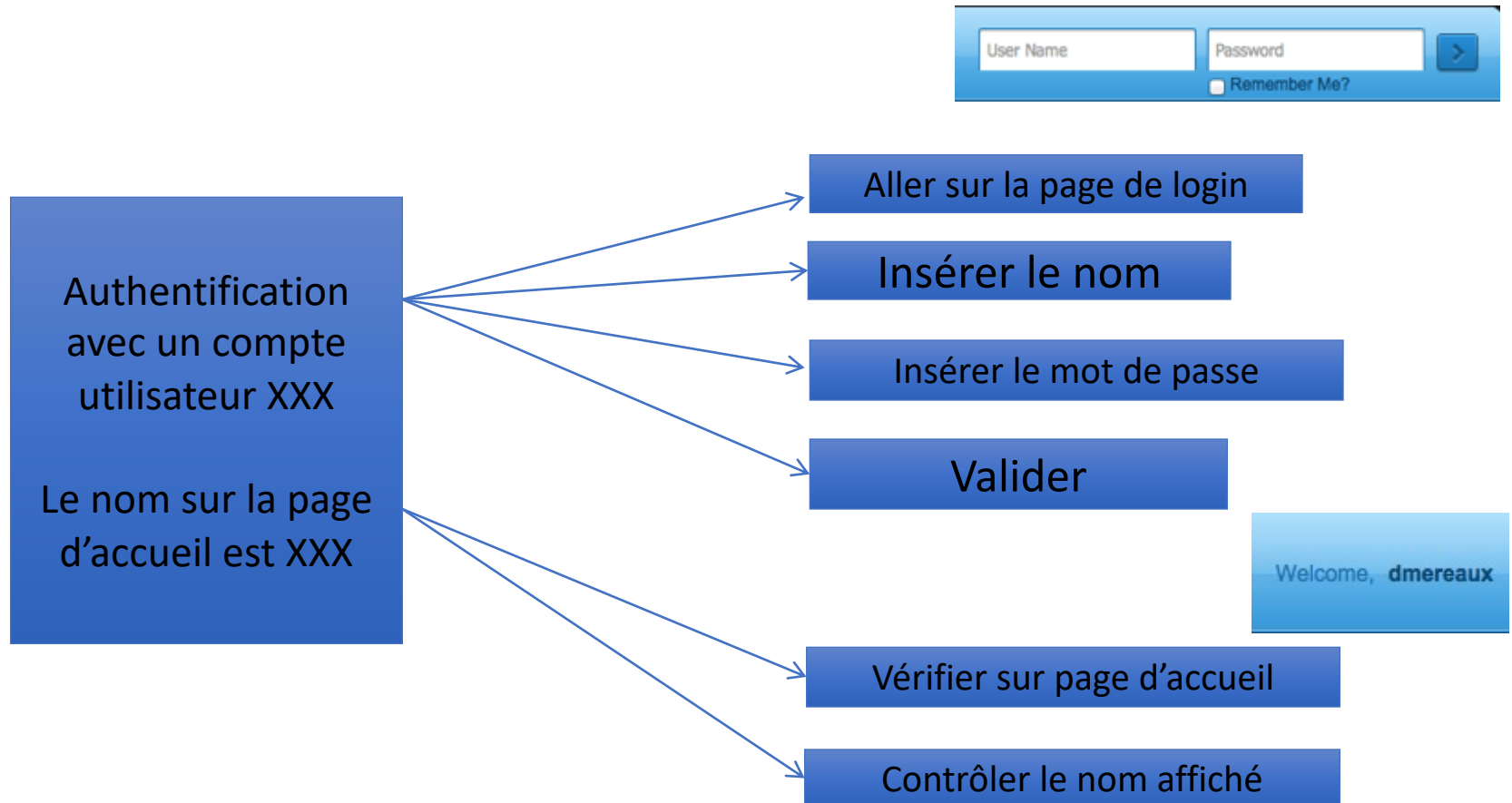
Web App

EXEMPLE

- Soit un test qui permet de vérifier que si je m'authentifie sur le système, alors une page de bienvenue avec mon nom apparaît.
- Pour réaliser ce test je pourrais utiliser les mots-clés:
 - Je m'authentifie sur la page de login
 - Je vérifie que mon nom est affiché sur la page d'accueil dans le message de bienvenue

EXEMPLE

- Mon script pourrait ressembler à:

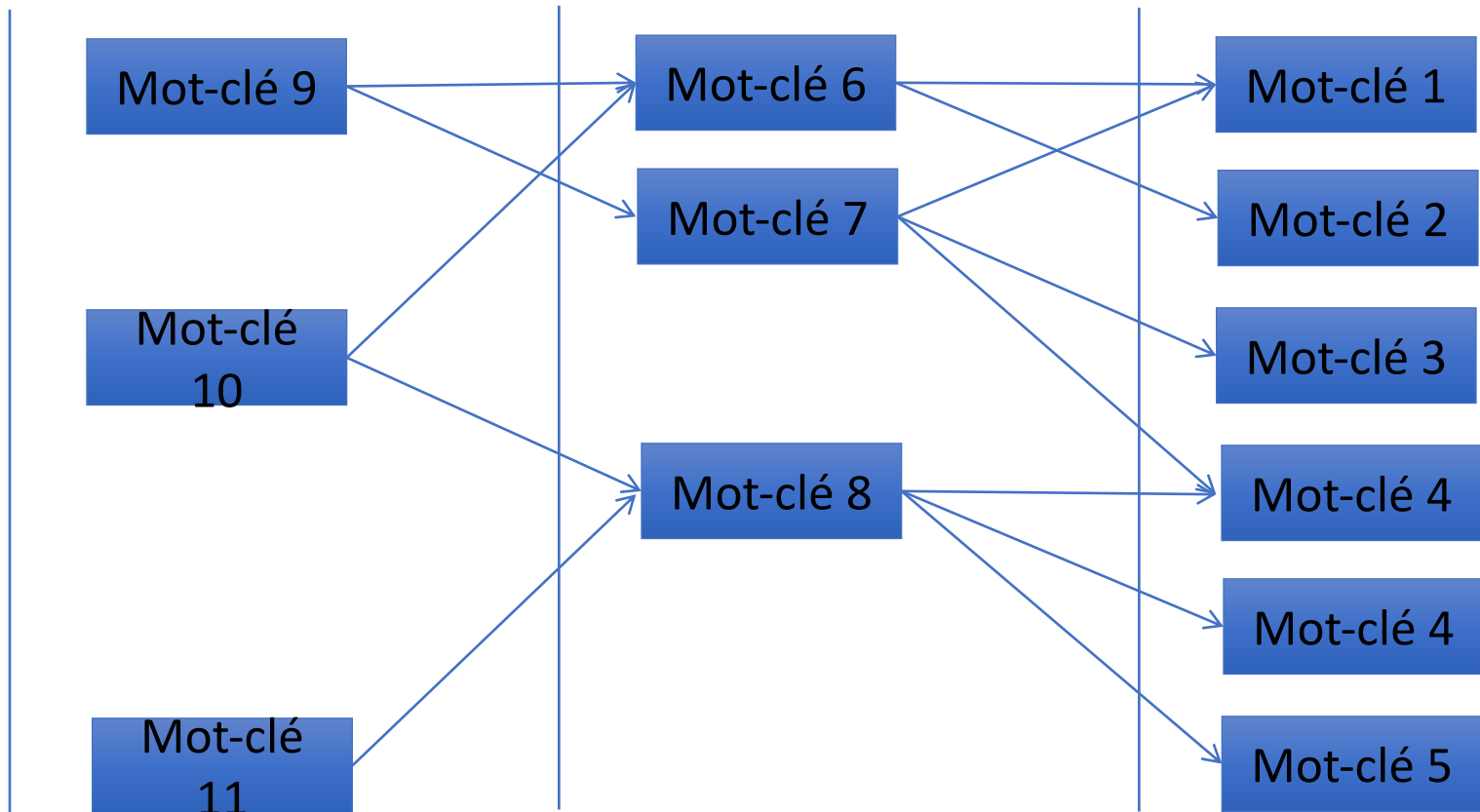


Différents niveaux d'abstraction

2^{ème} Niveau

1^{er} Niveau

Implémentation



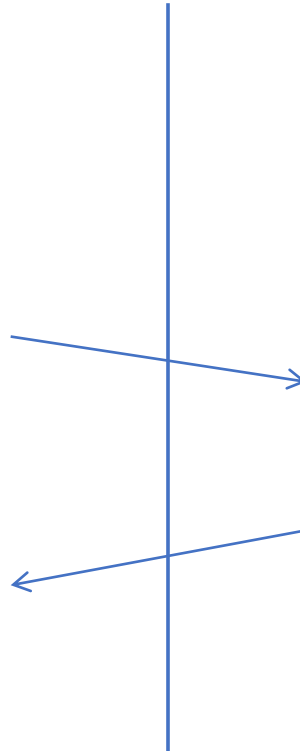
PROCESS de Développement des tests



1. Spécification des tests et sélection des tests à automatiser.
2. Création d'une bibliothèque de mots-clés de haut niveau.
5. Mise au point des mots-clés de haut niveau avec le produit développé.
7. Exécution des tests.



3. Implémentation via la création d'une bibliothèque de mots-clés de bas niveau.
4. Mise au point de ces mots-clés avec le produit développé.



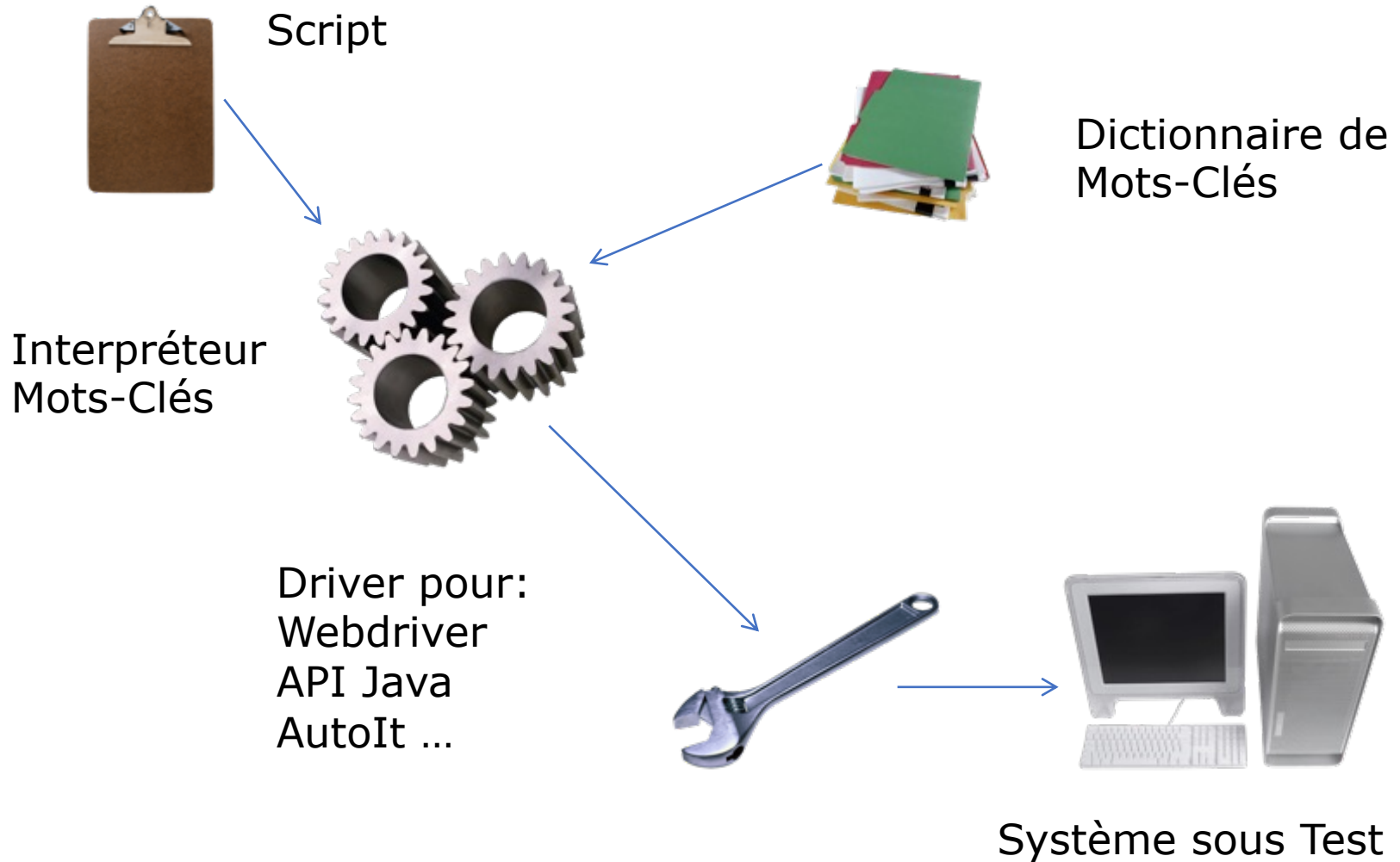
Un MOT-CLE = PHRASE

1. Authentification avec un compte utilisateur XXX
 2. Le nom sur la page d'accueil est XXX
- On pourrait également écrire:
 - **Etant donné** une page de login
 - **Quand** un compte valide est utilisé
 - **Alors** la page d'accueil est affichée avec le nom de l'utilisateur

POINTS CLEFS

- Création de mots-clés à partir de mots-clés
- Différents niveaux d'abstractions (métier aux tests unitaires).
- Les testeurs fonctionnels peuvent scripter des tests.
- Spécifications par les tests (scénarios, cas d'utilisations)
- Le test automatique n'est plus réservé aux tests de non régression.

ARCHITECTURE DE TEST



ROBOT FRAMEWORK

- HOSTE sous google
- LIBRE
- Data driven and behavior test driven
- Test System
- Python, jython, java
- <http://code.google.com/p/robotframework/>
- Un forum actif
<http://groups.google.com/group/robotframework-users>

Robot Framework

- Mots-Clefs ↔ Actions unitaires (code)
- Construction de mots-clefs à partir d'autres mots-clefs.
- Propre librairies ou des librairies existantes comme pour Selenium.

Vocabulaire et syntaxe

- Il existe 4 types de sections dans un fichier robot:
 - *** Settings *** → configuration, librairie
 - *** Test Cases *** → les cas de test à exécuter
 - *** Variables *** → déclaration de variables
 - *** Keywords *** → les mots clefs propriétaires
- Un mot clef peut contenir un espace, le caractère séparateur est donc 2 espaces.

Vocabulaire et syntaxe

- Éléments de syntaxe : exemple

```
*** Settings ***
Library ..... Selenium2Library
Test Setup → Ouvrir Application
Test Teardown → Fermer Application

*** Test Cases *** ....

T1 →
→ [Documentation] > cas nominal
→ [Tags] → NONREG
→ Un utilisateur indecis
→ je fais une recherche avec le mot cle jupes
→ je trouve 7 jupes

*** Keywords ***
Un utilisateur indecis
→ AllerALaPageDaccueil

je fais une recherche avec le mot cle jupes
→ lancerUneRecherche → jupe
```

Les mots clefs

```
*** Keywords ***
```

```
Mot Clef Simple
```

```
log to console      Hello World
```

```
Mot Avec Un argument
```

```
[Arguments]         ${nom}
```

```
log to console      ${nom}
```

```
Mot Avec Un retour
```

```
[Arguments]         ${nom}
```

```
log to console      ${nom}
```

```
[Return]            affiché
```

Les mots clefs sont déclarées dans une section keywords.

Un mot clef simple.

Un mot clef avec un argument.

Un mot clef avec un argument en entrée et un argument en sortie.

Utilisation des mots clefs dans un cas de test

```
]*** Test Cases ***
```

```
]Afficher Hello World
```

```
    [Documentation]    documentation du cas de test
```

```
]    |    log to console    Hello World
```

```
]Utiliser un mot clé propriétaire
```

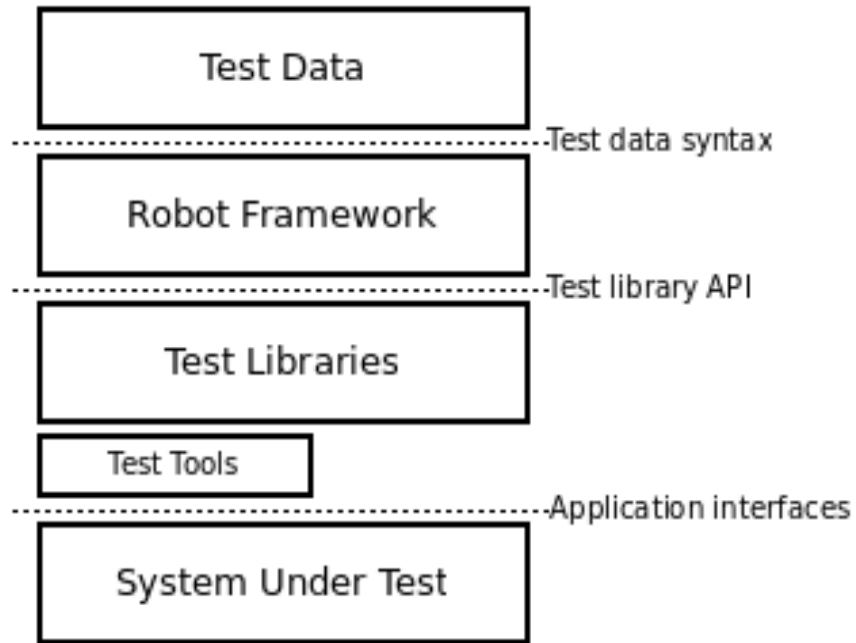
```
    Mot Clef Simple
```

```
    Mot Avec Un argument    Machin
```

```
    ${ret}    Mot Avec Un retour    Truc
```

```
]    log to console    ${ret}
```

Architecture



Robot Framework architecture

Il existe de nombreuses bibliothèques de test pour différentes technologies:

- Selenium
- Autolt
- ...

Combiné à RobotFramework on obtient un framework de test puissant et orienté test.

Librairies standard

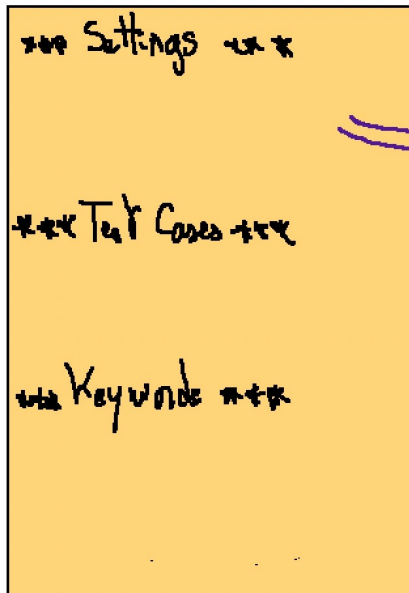
- Builtin: intégrée automatiquement, mot-clé génériques pour les tests.
- OperatingSystem: gestion de fichier, répertoire, var env ...
- Collections: gestion de liste
- DateTime: support date
- Dialogs : tests semi-automatique via boîte de dialogue
- Process: support de l' exécution de processus
- Screenshot: copie écran
- String: manipulation de chaine de caractères
- Telnet: support telnet
- XML: vérification et modification de document XML

Librairies externes

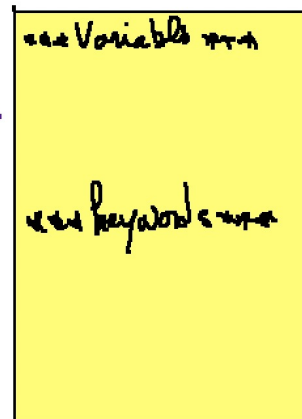
- Librairies pour test d'application mobile:
 - AppiumLibrary
 - AndroidLibrary
 - iOS Library
- Librairies pour le test d'interface Homme-Machine
 - AutoltLibrary
 - SeleniumLibrary,
 - SwingLibrary
 - Watir-robot
 - Browser
- API
 - Sudslibrary
 - HTTP library
- Base de données
 - Database Library

Organisation

Suite de test



Fichier ressource



- Suite de test → fichier robot
- Fichier Ressource → fichier txt
- Les Suites de test contiennent :
 - Une section settings
 - Inclusion de bibliothèques
 - Inclusion de fichier ressource
 - SetUp, TeardDown suite de test
 - Une section Test Cases avec l'ensemble des cas de tests.
 - Une section Keywords avec des keywords
- Les fichiers ressources contiennent
 - Des variables communes à plusieurs cas de test
 - Des Keywords communs à plusieurs cas de test

Cas de test

*** Settings ***

Test Setup	Ouvrir Navigateur
Test Teardown	Fermer Navigateur
Library	SeleniumLibrary
Resource	motclefPresta.resource

librairie externe

fichier ressource avec des
mot-clés personnalisés

*** Test Cases ***

TestRecherche

[Template]	Recherche
Mug	There are 5 products.
T-Shirt	There is 1 product.
notebook	There are 3 products.
xxxx	Sorry for the inconvenience.

Test Recherche MUG) nom du test

Given un acheteur potentiel	
When il fait une recherche avec	MUG
Then il trouve	There are 5 products.

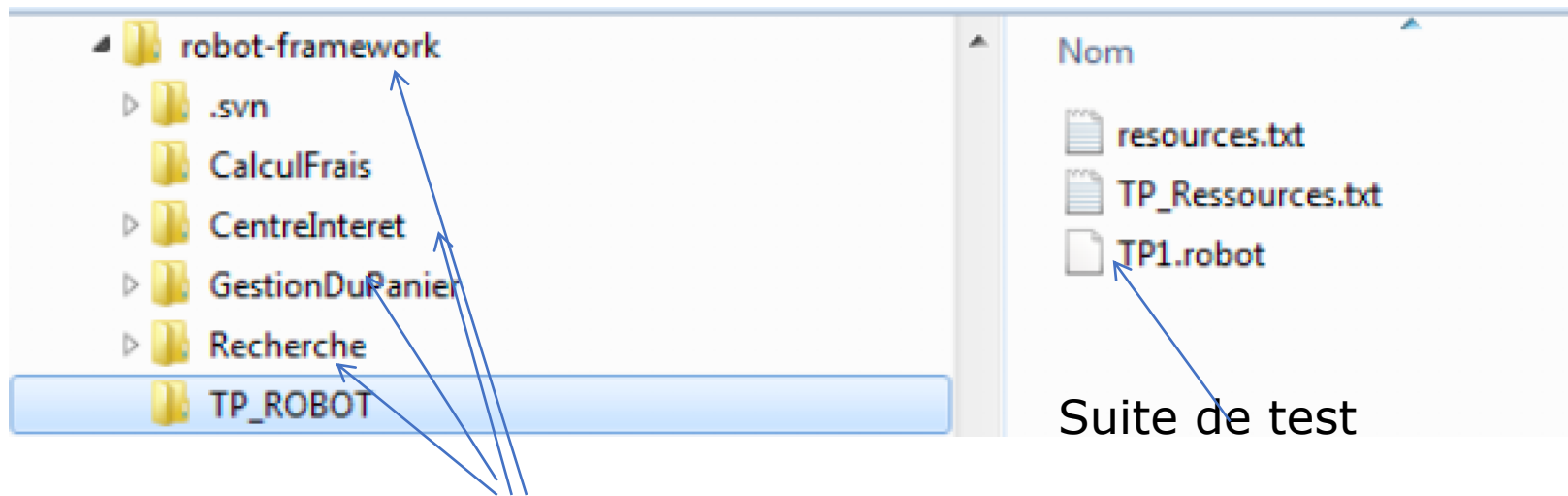
mot de
mot de + valeur

Test Connexion Echec

Given un utilisateur inconnu		
When il saisit	toto@titi.fr	123334444
Then il voit un message d'erreur		Authentication failed.

Suite de test

- Robot utilise la structure arborescente de Windows pour structurer les tests:



Suite de test

Suite de test

Mots-clés et données

```
*** Settings ***  
Resource      MesPremiersMotsCles.robot  
Test Template  Afficher 2 Arguments
```

```
*** Test Cases ***
```

```
T1    marcel    dupont  
T2    louis     dubois  
T3    ernest    le coeur
```

```
Afficher 2 Arguments  
[Arguments]    ${arg1}    ${arg2}  
${ret}    concatenate    ${arg1}    ${arg2}  
log To Console    ${ret}
```

Exemple library HTTP LIBRARY

```
*** Settings ***  
  
Library           RequestsLibrary  
Library           XML  
Library           String  
Library           Collections  
Library           JSONLibrary  
  
Suite Setup       Create Session    maSession    https://api.chucknorris.io    disable_warnings=1  
Suite TearDown    Delete All Sessions  
  
*** Test Cases ***  
  
Chuck  
    Create Session    maSession    https://api.chucknorris.io  
    ${resp}          GET On Session    maSession    /jokes/random    expected_status=200  
    Log To Console    ${resp.content}
```



resp.content



resp.status_code

Exemple library HTTP LIBRARY

- status_code réponse à la requête si succès code 200
- content : contenu de la réponse, il peut être au format:
 - XML (exploitable avec la bibliothèque XML)
 - Ou JSON (exploitable avec la bibliothèque JSONLibrary ou Collections)

Library Requests

