



---

## Contrat de professionnalisation

---

Titre de la mission

---

Sous la direction de Le Tuteur.

Date Visa du tuteur industriel	Date Visa du tuteur pédagogique	Date Visa du service formation continue



### **Remerciements**

Merci à Namrod pour toute la partie sur la bibliographie. Retrouvez ses question FAQ qui ont permis la rédaction de cette partie.

Merci à f-leb, LittleWhite et Metalman pour leurs conseils et la relecture.

Merci à ced et jacques\_\_jean pour la correction orthographique et typographique.



# Sommaire

<b>Remerciements</b>	<b>ii</b>
<b>Sommaire</b>	<b>v</b>
<b>Introduction</b>	<b>1</b>
<b>1 Présentation de l'entreprise</b>	<b>3</b>
<b>2 Software tester</b>	<b>5</b>
<b>3 Migration Jira/Java Correction WorkBench</b>	<b>19</b>
<b>4 Plugin Jenkins</b>	<b>23</b>
<b>Conclusion</b>	<b>35</b>
<b>A Les niveaux de test</b>	<b>37</b>
<b>B Résultats quotidien des tests</b>	<b>39</b>
<b>C Tous les tests implémentés</b>	<b>43</b>
<b>D Les bases pour comprendre Jenkins</b>	<b>49</b>
<b>E Maven</b>	<b>51</b>
<b>Table des figures</b>	<b>53</b>
<b>Liste des tableaux</b>	<b>55</b>
<b>Table des matières</b>	<b>57</b>



# Introduction

La 1<sup>ème</sup> partie de mon contrat s'est étalée du 25 juillet au 29 septembre 2015, période durant laquelle j'ai pu, d'une part, me familiariser avec les différents outils d'un employé SAP. D'autre part, étudier le document d'architecture de WebI et apprendre à utiliser ses caractéristiques de base : , , , , .





# Chapitre 1

## Présentation de l'entreprise

### 1.1 SAP dans le monde

### 1.2 Contexte actuel

La structure actuelle est le fruit de la fusion entre SAP France (+/- 500 salariés) et Business Objects (BOBJ - +/- 1000 salariés) dont la date légale est le 1er janvier 2010. Précédemment, BOBJ avait absorbé une autre entité, CARTESIS. D'autres petites entreprises (60/100 salariés) sont en cours d'intégration avec salariés français en 2013 : Sybase (au 01/04/2013), Success Factors et Ariba. avec crystal object ils ont intégré les deux en même temps => la 4.0

### 1.3 Présentation de l'équipe

Nom de l'équipe : P&I BIT BI Suite Paris ST Transverse team

Diagramme présentant l'équipe

Les tickets : IT ticket, hrTicket, Ticket CSS



# Chapitre 2

## Software tester

### 2.1 Généralités sur le test

Interview avec Fabien

### 2.2 Le test à SAP

!!!!!!!Présenter l'architecture de test et les différents process!!!!!!!

L'équipe Transverse est particulièrement focalisée d'une part, sur le test coverage, les tests de régression, les tests de performance ; et d'autre part sur le principe « un bug un test ». Ce principe consiste, comme son nom l'indique, que chacun des bugs fixés par un développeur soit systématiquement testé par un ST

SAP utilise un grand nombre d'outils pour gérer le code de ses nombreux produits, possédant chacun plusieurs branches, ayant chacune une suite de tests exécutée quotidiennement. Globalement les codes, quels qu'ils soient, sont hébergés sur le gestionnaire de code source Perforce. La compilation journalière est exécutée par Jenkins ou ASTEC, dépendamment des équipes et des produits, dont les résultats sont automatiquement envoyés aux personnes concernées (cf. annexe B page 39 qui présente l'un de ces mails automatiques).

Les ST sont régulièrement tenus au fait des tests à implémenter aussi bien par Jira ou Java Correction WorkBench que par liste de distribution de mails (cf. figure 2.1 page 6 pour le process complet)

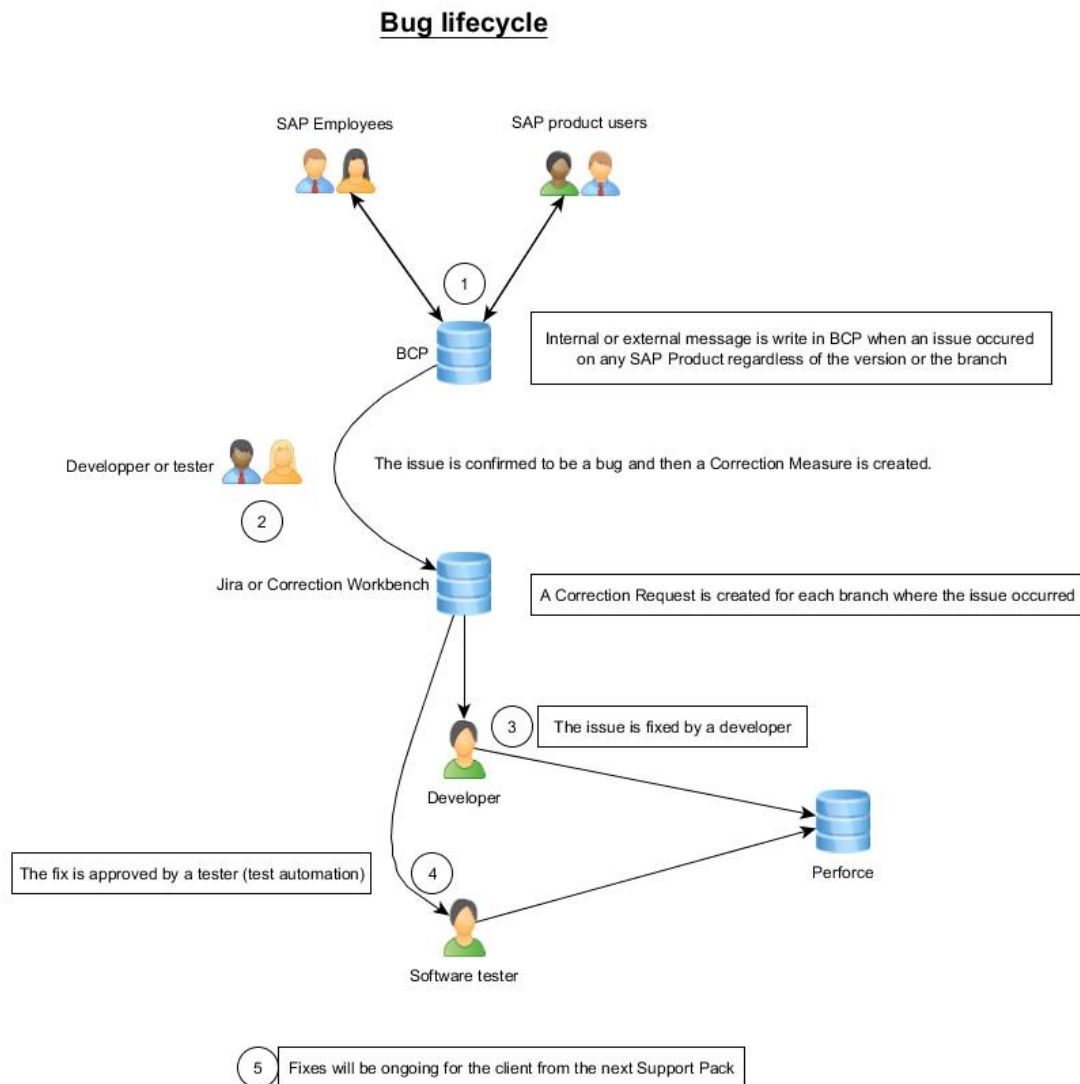


FIGURE 2.1 – Le process de test

## 2.3 Présentation du produit testé : WebI

WebI est un logiciel de BI permettant d'accéder à des données stockées en ligne dans un « univers » (anciennement des fichiers aux extensions .unv, maintenant .unx). L'accès aux données peut se faire via rich client via applet ou via le client dhtml.

## 2.4 Préparation aux tests

Mes 1<sup>er</sup> jours à SAP se sont déroulés de la manière suivante :

- Présentation à l'équipe et visite des locaux
- Réunion avec mon tuteur et mon manager pour une description de la mission
- Récupération des différents droits d'accès aux serveurs
- Familiarisation avec les outils internes (tickets HR, CSS, IT, ...)
- Installation des logiciels nécessaires au développement (IDE, SCM, éditeur de texte)
- Mise en place du framework de test (création workspace perforce) à l'aide de Christophe DOLIMONT
- Au terme de la 1<sup>ère</sup> semaine, j'ai pu commencer à utiliser le framework de test en me basant sur les tests déjà existants

Une fd

## 2.5 Déroulement de la 1<sup>ème</sup> mission

!!!!cf journal de bord tests!!!!!!!!!!!!

!!!! caractéristique de base de WebI!!!!!!!!!!!!

L'objectif principal de cette mission était d'être, à terme, capable d'implémenter seul un test automatique. La difficulté majeure rencontrée au début de cette période a été de comprendre la logique de test du framework de test. Les points suivants sont importants pour pouvoir utiliser le framework correctement :

- L'intégration du test à la suite de tests
- Quel type de test mettre en place, statique ou dynamique ?
- Comment initialiser un test, les constructeurs des tests dynamiques peuvent se baser sur un lcmbiar, un wid ou une querspec
- Les tests ne se basant pas sur les mêmes versions des produits, les JARs ne sont jamais les mêmes
- Où trouver les fichiers nécessaires

### 2.5.1 Tests statiques ou dynamiques

Le test, qu'il soit statique ou dynamique, sera exécuté avec tous les autres dès lors que le nom du test plan est inscrit dans la test suite.

La différence fonctionnelle entre ces deux types de tests est que l'un ne fait que comparer deux doc alors que l'autre fait des modifications sans nécessairement le comparer à une référence.

La différence horaire est très importante, pour un test statique il n'est pas nécessaire d'implémenter de testcase ! Il suffit de générer la référence, de la mettre dans le dossier dédié, compléter le script.xml et la test suite, implémenter le test plan et commit.

## Tests statiques

Le test statique

- ne fait que comparer un document par rapport à une référence, aucun testcase n'est donc à implémenter
- ne demande que très peu de connaissances technique, que ce soit en Java ou sur WebI

Lors de l'exécution d'un test statique, un fichier est généré à partir d'un wid, son format (txt\_1, txt ou doc) est précisé dans le fichier script.xml.

Ensuite, à la fin de l'exécution du test, ce fichier généré est comparé avec un fichier de référence (ce fichier de référence étant généré par le ST lors de l'implémentation de son test en mettant l'option savingtxt à true).

Si les deux fichiers sont identiques le test est correct, sinon il échoue.

Pour l'exécution d'un test statique il ne faut qu'implémenter un testplan qui respecte la structure suivante :

```

1 package rebean_wi.customers.Aerospatiale_Matra_Airbus.ID_187739;

3 import model.filters.Mode;
  import model.filters.Severity;
5 import model.filters.Type;
  import model.filters.testplan.Suite;
7 import model.filters.testplan.TestPlanType;
  import model.filters.testplan.features.Features_REBEAN;

9
10 import org.junit.Test;
11
12 import tests.exported.annotations.BOTest;
13 import tests.exported.annotations.BOTestPlan;
14 import extensions.toolbox.WIStaticTestPlanDefinition;
15
16 @BOTestPlan(Type = TestPlanType.FEATURE_VERIFICATION, Suite = Suite.AURORA,
17             Feat = Features_REBEAN.WI)
18 public class CM_{cmNumber} extends WIStaticTestPlanDefinition{
19     private static String _scriptId = "CM_{cmNumber}_{short
20     text}";
21     public CM_{cmNumber} () {
22         super(_scriptId);
23     }
24
25     @Test
26     @BOTest(Objective = "Test CM_{cmNumber}_{short
27     text}' functionality", Severity = Severity.CRITICAL, Author = "", Mode = Mode.
28     PROD, Type = Type.FUNCTIONAL)
29     public void CM_{cmNumber}_{short text} () {
30         logNumericResults(launchTC(getTestcase("CM_{cmNumber}_{short text}")));

```

---

1. type personnalisé interne à SAP

```
29 }
}
```

## Tests dynamiques

À la différence du test statique, le test dynamique va modifier le document après ouverture. Ceci permettant de s'assurer que la mécanique interne de WebI produit l'effet escompté sur le document. La comparaison avec un document de référence est bien évidemment possible.

Un testplan respecte l'implémentation suivante :

```
package rebean_wi.customers.{customerName}.{customerID};
2 //imports
@BOTestPlan(Type = TestPlanType.FEATURE_VERIFICATION, Suite = Suite.AURORA41,
    Feat = Features_REBEAN.WI)
4 public class CM_{CMNumber} extends WIDynamicTestPlanDefinition {
    private static String _scriptID = " CM_{CMNumber}_{text}";
6    public CM_{CMNumber}() {
        super(_scriptID);
8    }
    @Test
10    @BOTest(Objective = "Test 'CM_801959_2014_ValuesMissing' functionality",
        Severity = Severity.CRITICAL, Author = "ptaquet", Mode = Mode.PROD, Type =
        Type.FUNCTIONAL)
    public void CM_{CMNumber}_{text}() {
12        logNumericResults(launchTC(getTestCase("aurora_customers. .{customerName}.CM_
            {CMNumber}_{text}"));
    }
14 }
```

Et son testcase respecte l'implémentation suivante :

```
package aurora_customers.{ customerName};
2 public class CM_{CMNumber}_{text} extends MonoDocTestcase {
    MonoDocTestcaseConfigInfo _tccInfo;
4    //private final static String _docPath = "";
    MSGStep _step = null;
6
    Public CM_{CMNumber}_{text} (MonoDocTestcaseConfigInfo tccInfo,String docName
    ) {
8        super(tccInfo, docName, "corporate", _docPath, true);
        _tccInfo = tccInfo;
10    }
    @Override
12    protected void run() throws Exception {
        startTestcaseStep("");
14    //
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```

16      //
      _step = _msgHelper.startStep("");
      //Here, the implementation of the code
18      _msgHelper.stopStep(_step);

20
      stopTestcaseStepWithoutActionWI();//Stop without compare
22  }
  }

```

À la lecture du test case on peut observer le constructeur de la super classe, celui-ci prends un certain nombre d'arguments. Il existe 10 constructeurs différents pour la super classe `MonoDocTestCase` mais 2 sont particulièrement important. L'un permet de baser son test sur un document généré à partir d'une `querspec`, l'autre sur un document `.wid`. Par exemple le constructeur se basant sur un `wid` :

```

1 MonoDocTestCase(MonoDocTestCaseConfigInfo tccInfo, String sDocumentName, String
    sCategoryType, String sCategory, Boolean useAuroraCtx);

```

où

**tccInfo**

Objet contenant les informations générales relatives au test case

**sDocumentName**

Nom du document `.wid` qui sera chargé lors de la construction de l'objet

**sCategoryType**

Nom de la catégorie, en général : « corporate »

**sCategory**

Path du dossier dans lequel est stocké le `.wid`. Doit respecter la nomenclature suivante : `/auto/{scriptname}/wid`

**useAuroraCtx**

Le test en question porte t-il sur aurora ?

Dans tous les cas, lorsque l'objet `testcase` est instancié, nous pouvons implémenter un code manipulant le document au niveau SDK ; ce qui signifie que l'on ne simule pas un clic sur un bouton mais que l'on appelle l'une des méthodes « derrière » ce bouton, reproduisant ainsi le comportement d'une manipulation au niveau GUI

## Exécution d'un test

Lors de l'exécution d'un test plusieurs fichiers sont générés en divers endroits, il y a par exemple :

**Des fichiers de log**

sd



**La sortie console**

Enregistrée dans

... \Workspace\_Aurora\rebean\logs\{scriptname}

**Les fichiers générés**

Enregistrés dans

... \Workspace\_Aurora\rebean\results\Your Build Number\res\{ scriptName}\CM\_{cmNumber}\_{short text}\

uniquement si l'enregistrement des ressources est spécifié dans le script.xml

Si savingDoc est à True le document de référence est enregistré sur le CMS dans le dossier Folders/Public Folders/auto/, voir l'illustration 2.2 page 11, et le document généré est enregistré dans My Documents/My Favorites/Personnals Documents, voir l'illustration 2.3 page 12

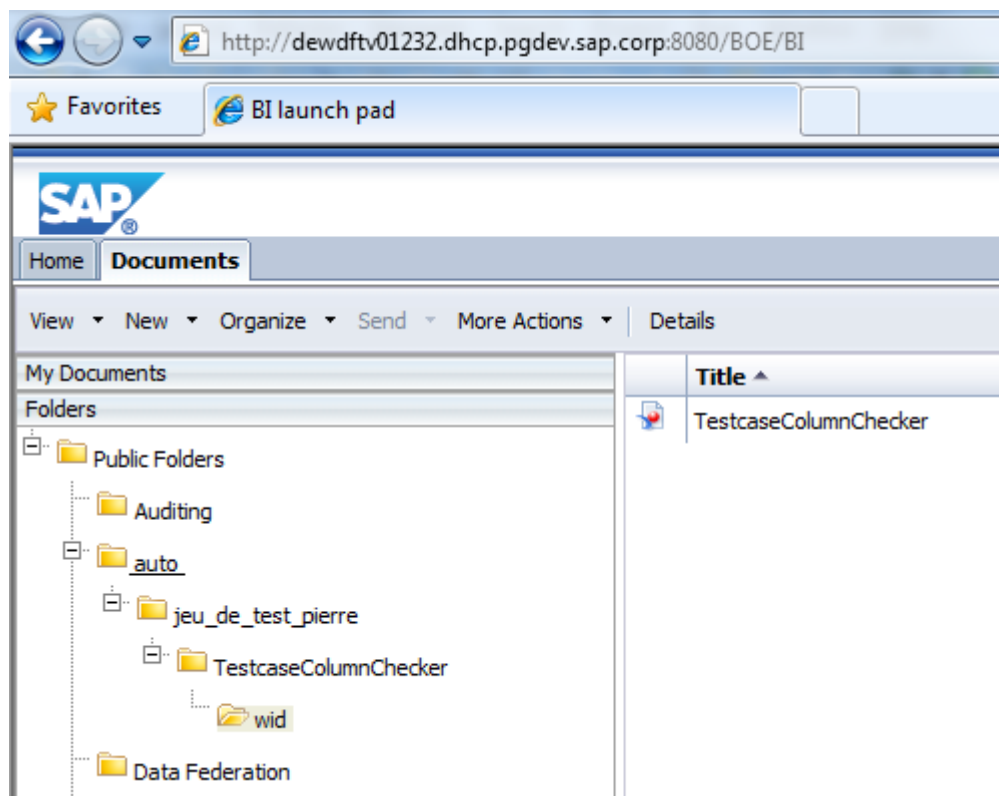


FIGURE 2.2 – Capture d'écran de WebI de l'arborescence du document de référence

**Les fichiers de référence**

Les .wid sont enregistrés dans

... \Resources\_AURORA\storage\auto\{ scriptName}\wid

et les autres documents de références (txt, html, etc.) sont dans

... \Resources\_AURORA\storage\auto\{scriptName}\CM\_{cmNumber}\_{short

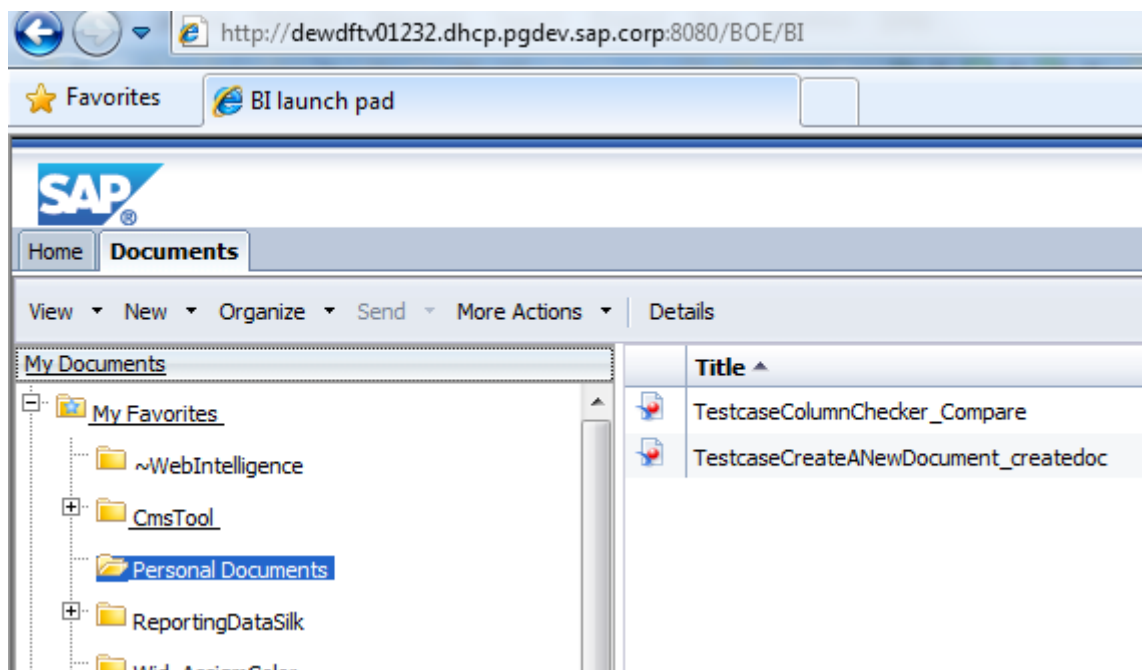


FIGURE 2.3 – Capture d'écran de WebI de l'arborescence du document généré

text}}\

voir une illustration de cette arborescence figure 2.4 page 12

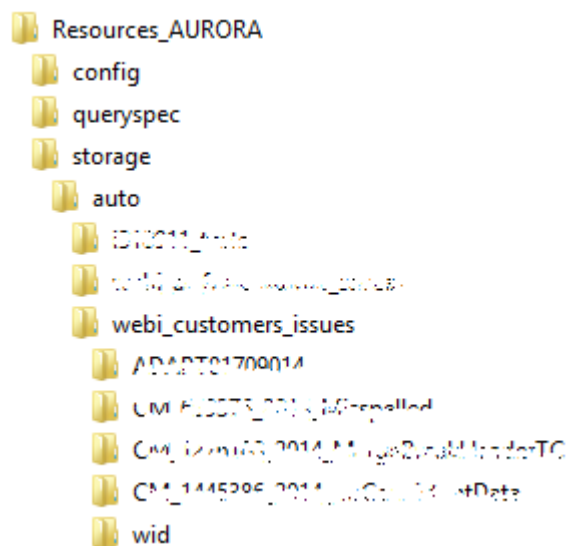


FIGURE 2.4 – Capture d'écran de WebI de l'arborescence des références locales

### Les logs du serveur tomcat

Disponible dans le dossier d'installation du serveur ces logs sont trop verbeux pour pouvoir être utiliser, mais sont cependant utiles dans certains cas.

### 2.5.2 De l'étude à l'intégration

Tout d'abord, les requêtes de test auto ne sont pas toujours réalisables à notre niveau, il faut donc d'abord vérifier la faisabilité du test. Une fois que l'on a validé que la réalisation est possible et que l'on a choisie une stratégie de test, il faut préparer l'environnement de travail, c'est-à-dire construire la coquille vide du test désiré.

#### Analyse du test à implémenter

**Étudier le bug** Au préalable, toutes les informations que l'on a sur le bug à tester se trouvent sur JCWB, étudier le workflow provoquant le bug, la description du bug, la/les branche(s) sur laquelle/lesquelles il survient (voir figure 2.5 page 13) Ceci fait, il est intéressant d'aller étudier le code qui a été modifié pour corriger

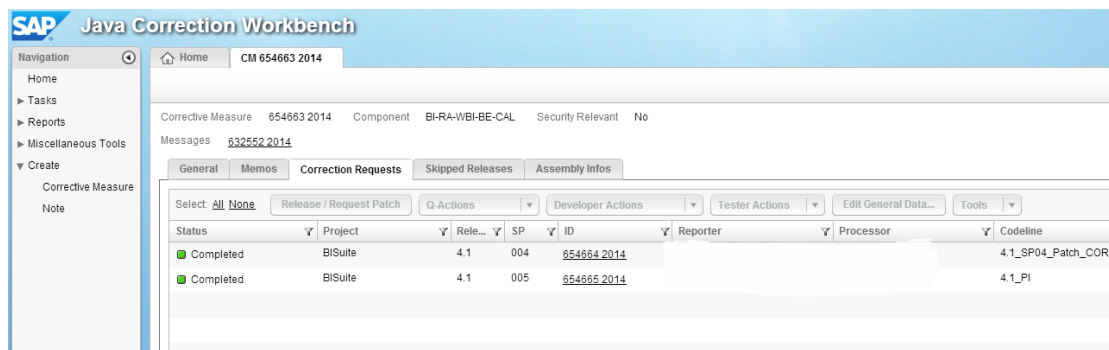


FIGURE 2.5 – Écran de JCWB propre à une CM

le bug. Pour cela il suffit d'utiliser la fonction « diff against previous revision » du SCM pour obtenir la liste exhaustive des fichiers modifiés ainsi qu'un vis-à-vis sur les versions pré-correctif et post-correctif (voir figure 2.6 page 14)

**Reproduire le problème** Une fois que le bug est bien compris il nous incombe de reproduire à la main le workflow et de valider l'existence du bug sur la version buggée ainsi que le bon fonctionnement de la version corrigée. À cette étape, si le bug survient sur le CMS (client léger), il est intéressant d'utiliser le debugger du navigateur pour observer les données transitant.

**Définir la stratégie de test** Maintenant que le problème est bien compris et localisé, nous pouvons savoir s'il est possible de le tester. Si non, soit le problème ne peut pas être testé soit nous redirigeons la correction vers l'équipe de testeurs concernée (plus ou moins haut ou bas niveau).

Si l'implémentation du test est possible, il faut choisir la meilleure manière de

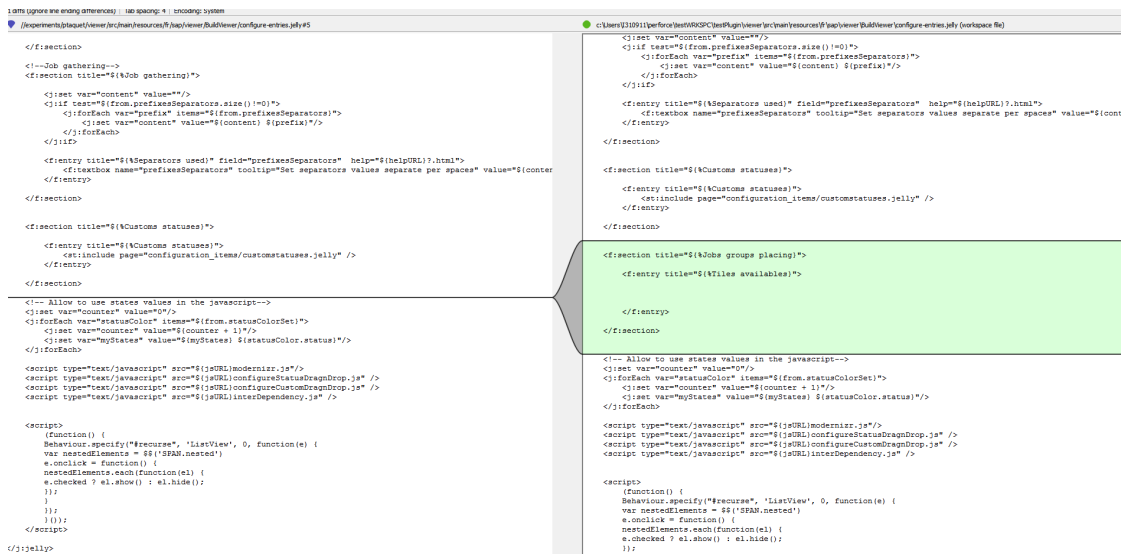


FIGURE 2.6 – Écran de comparaison des 2 versions d'un même fichier (avant et après correctif)

tester l'existence et la non-existence du bug ainsi que le moyen le plus rapide d'arriver à reproduire le problème. Par exemple faut-il un test statique ou dynamique ? Partir d'une querspec ou d'un document .wid ?

## Implémentation de la coquille vide

L'intérêt de coder d'abord la coquille vide est d'avoir un code qui compile mais qui ne fait encore rien. Ce qui garantit que tout se passe correctement au niveau de la nomenclature, la création du test case et, au besoin, la génération sur serveur des .wid. Comme schématisé figure 2.7 page 14, chaque test automatique est intégré à un test plan, lui-même étant intégré à une test suite.



FIGURE 2.7 – Diagramme UML des suites de tests

Ci-dessous les fichiers à implémenter :

1. **Test plan** Créer le nouveau test plan dans le package correspondant au client qui a remonté le bug (dans `testplan.srebean_wi.customers.{clientId}`). Renseigner

toutes les informations relatives au test, si ce fichier est correctement implémenté il ne sera plus nécessaire de le modifier par la suite.

2. **Test case** Créer le test case dans le package correspondant au client (dans `test-cases.aurora_customers.{clientID}`). Attention à respecter le pattern de nommage « `CM_{CM_id}_shortText` ».
3. **Test suite** Dans le package des tests plan, on peut trouver la test suite qu'il faut modifier. Il suffit de rajouter le nom du test plan à la liste de test plan que la test suite exécute.
4. **script.xml** Ajouter les différents paramètres correspondants au test.
5. **ressources** Générer les documents ressources nécessaires (`queryspec`, `.wid`, etc.) et les ajouter au dossier portant le nom de la CM
6. **parameters.xml** Renseigner l'url du CMS ciblé et mettre à jour les extracted JARs permettant au test vide de compiler
7. Vérifier l'intégrité de la change list de perforce et finir par push les modification.

D'une manière générale, la structure générale à connaître pour implémenter correctement la coquille vide est illustrée figure 2.8 page 17

### Les ressources

Les ressources sont très importantes dans le contexte du test, celles-ci se présentent sous plusieurs jours différents :

1. un document `.wid` créé en suivant à la lettre le workflow à tester mais dont la construction à été arrêtée juste avant que ne survienne le bug.  
Cette ressource sert à utiliser un document déjà construit et permet au ST de n'automatiser que la partie à tester.
2. un fichier (`txt`, `txt_`, `pdf`, `doc`, `xls`, `xls`, ...) considéré comme une référence  
Cette ressource est comparée au document obtenu à la fin du workflow pour garantir la similitude.
3. une `queryspec`, c'est un fichier `xml` représentatif du document `.wid`

### Obtenir un fichier wid

**Via le CMS** Il suffit de parcourir l'arborescence du CMS pour arriver à l'emplacement du `.wid`. Dans les propriétés du fichier il y a son nom complet (différent du nom dans WebI) avec son arborescence à partir du dossier Input (figure 2.9 page 18 ). Ensuite, dans le système de fichiers du serveur (par exemple : « `\\dewdftv01634.dhcp.pgdev.sap.corp\c$` ») aller dans « `\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\FileStore\Input` » et copier/coller le chemin d'accès au fichier. Le document `.wid` se trouve dans le répertoire en question.

**Via le Rich Client** Attention à la version du rich client qui doit être la même que celle du CMS. Ouvrir une connexion pointée sur le bon CMS, ouvrir le document désiré et enregistrer sous le document `.wid`.

## 2.6 Bilan de mes 1<sup>ères</sup> missions

Tout au long des mois de juillet, août et septembre j'ai implémenter beaucoup de tests<sup>2</sup> pour des bugs divers et à l'impact plus ou moins critique. J'ai beaucoup appris de mes erreurs, surtout lorsque plusieurs jours de travail s'avéraient inutiles parce qu'une meilleure solution, évidente pour qui sait, existait.

### 2.6.1 Les 1<sup>ères</sup> erreurs

Les principales pertes de temps : coder des choses inutiles ou déjà existantes. réinventer la roue. Beaucoup de choses que j'ai codé se sont avérés être déjà existantes dans le framework

perdre du temps à implémenter des choses inutiles (exemple : création d'un doc niv sdk alors que l'on peut juste le récupérer)

Casser la build : ne jamais push le vendredi soir au risque de casser la build du week-end.

### 2.6.2 Les acquis

ce qu'il faut faire pour tester

comprendre le test (étudier le workflow et cibler exactement la partie du workflow à tester) mettre en place la coquille vide du test et générer et/ou récupérer les fichiers utiles au développement et/ou à l'exécution du test. Tester le test ! ie exécuter sur 2 versions, l'une fixée l'autre non. push le test

ne jamais push le vendredi soir

la composition d'un document WebI au niveau SDK

---

2. cf. annexe C page 43 pour la liste complète des tests implémentés

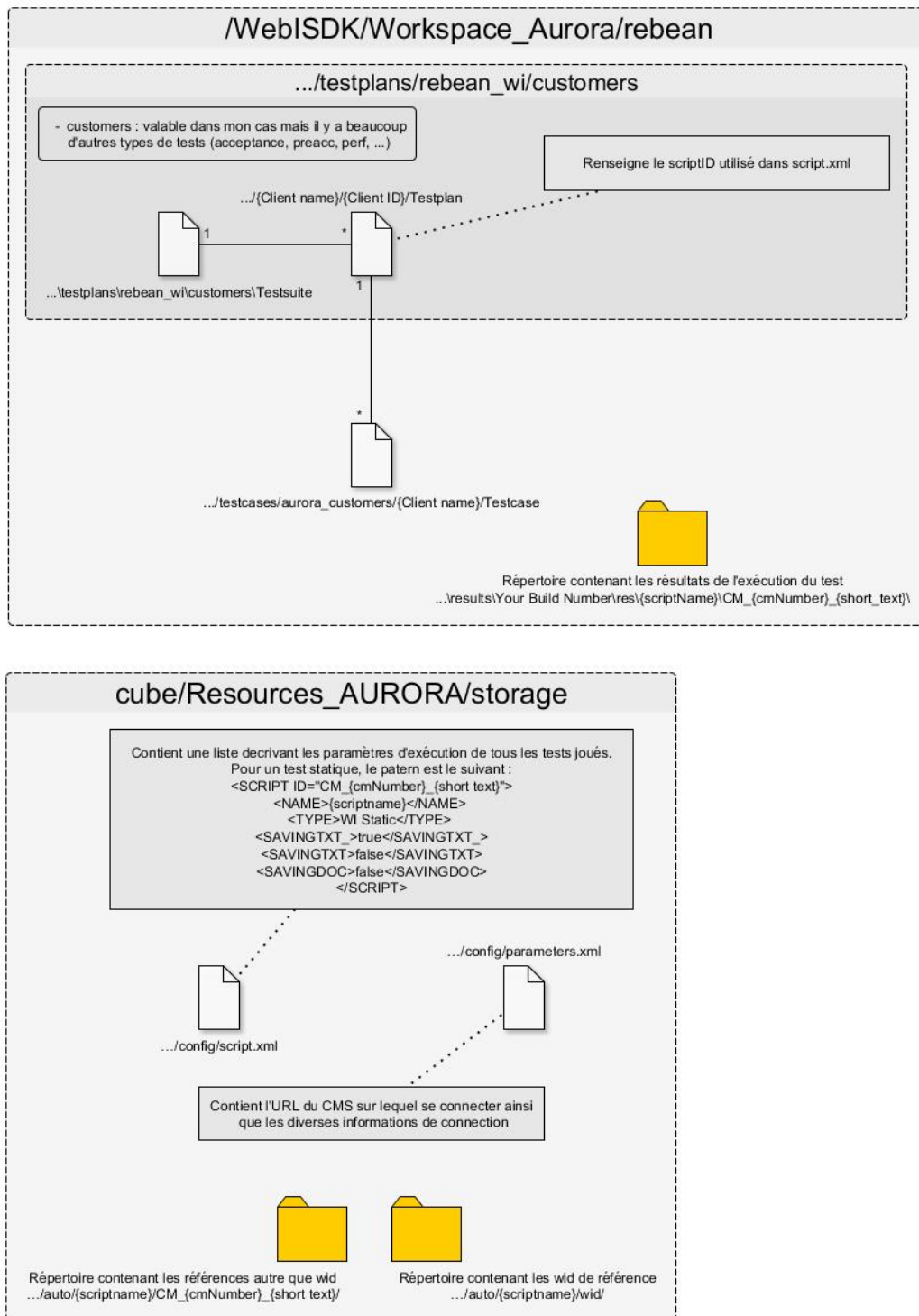


FIGURE 2.8 – Diagramme représentant les différents éléments qui compose la coquille vide d'un test dynamique

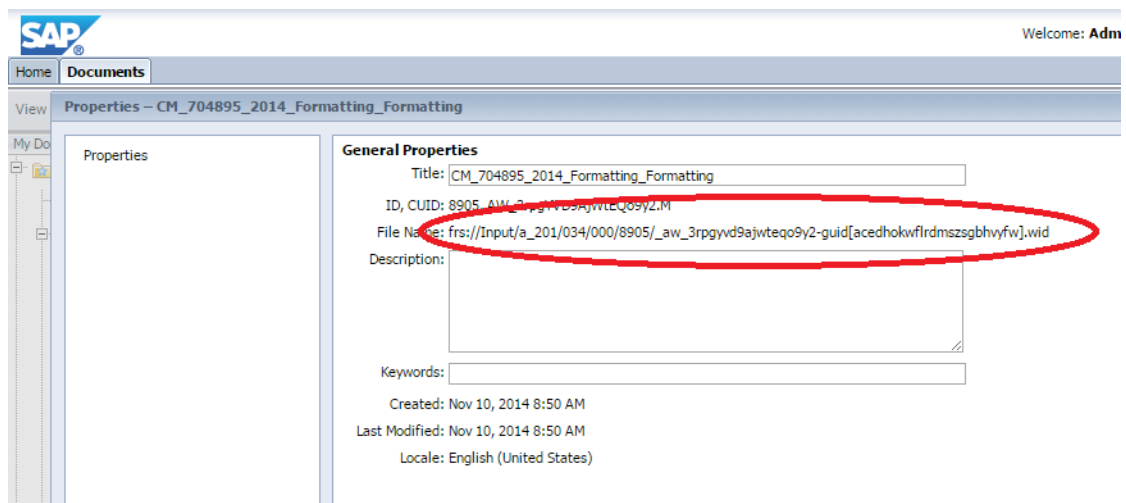


FIGURE 2.9 – Capture de l'écran de propriété d'un document WebI



# Chapitre 3

## Migration Jira/Java Correction WorkBench

### 3.1 Qu'est-ce que Jira et Java Correction WorkBench

Jira est un logiciel de traçabilité des problèmes développé par Atlassian dont l'usage est gratuit pour les organisations à but non lucratif, de charité ou les projet open-source. SAP désire migrer de système de traçabilité pour utiliser maintenant Java Correction WorkBench (JCWB ou CWB).

JCWB est un logiciel..... voir Fabien

### 3.2 Présentation du contexte

Comment cela se passe actuellement à SAP à SAP tous les codes des différents projets sont hébergés sur le gestionnaire de version Perforce (SCM : Source Code Management) et ceux-ci sont compilés plusieurs fois chaque jour grâce à Jenkins ou ASTEC (CIS : Continuous Integration Software).

Chaque projet se compose d'une part son code source mais aussi d'une grande quantité de tests qui sont joués pour garantir<sup>1</sup> le bon fonctionnement du produit. Lorsqu'il y a un problème sur la build, que ce soit une erreur de compilation ou un test qui échoue, le statuts de la build change en fonction du problème. Dès lors que quelqu'un c'est aperçu du problème celui-ci inscrit un defect dans Jira<sup>2</sup>.

---

1. La valeur de la garantie dépend directement du test coverage

2. L'utilité de Jira est bien plus large que la simple déclaration d'un test échoué, il sert à décrire n'importe quel problème quelle que soit la version, la branche ou le produit

### 3.2.1 Plugin de reporting

Pour leur permettre d'avoir un rapport visuel sur l'état des builds, ils utilisent le plugin Radiator. Ce plugin permet l'affichage, sur un seul écran divisé, des groupes de jobs<sup>3</sup>. Un groupe de jobs est un carré coloré dont la couleur représente l'état des jobs qui le compose.

Les statuts pris en compte sont les statuts Jenkins<sup>4</sup> ainsi qu'un statuts supplémentaire issue du plugin Claim. Ce plugin permet à un développeur de claim une erreur, c'est-à-dire que le projet comporte une information supplémentaire qui est un nom d'utilisateur et le message que celui-ci a laisser.

En résumé Radiator :

- permet de rassembler les jobs en un seul groupe de jobs
- où chacun des jobs a un statuts<sup>5</sup>
- le groupe de projet arborant la couleur jugée la plus importante
- de prendre en compte les informations supplémentaires du plugin claim

Par exemple, si nous avons deux groupes composés chacun de trois jobs, tous différents. Supposons que, sur les 6 jobs, l'un soit en échec les autres étant en succès. Nous aurons donc une vue colorée en verte (parce que tous les projets sont en succès) et la deuxième vue apparaissant en rouge (l'un des jobs ayant échoués) ou bien en orange si l'échec est investigué (échec claim par un développeur).

#### Problème

Cette solution permet de visualiser les statuts des jobs mais ne fait pas de distinction parmi les jobs échoués. Parmi ceux-ci on peut distinguer :

- les jobs échoués depuis très longtemps
- les jobs échoués avec un defect enregistré

L'inconvénient est que les tests échoués récemment et non reconnus apparaissent de la même manière que les tests échoués et reconnus. La solution provisoire mais permettant de ne plus polluer l'affichage avec du rouge, consiste à faire passer les jobs en échecs avec defect (ie. reconnus et enregistrés comme échecs dans Jira ou CWB) sont passés en vert. De cette manière, seul les nouveaux échecs apparaissent en rouge. En grande partie des régressions.

Le revers de la médaille est que l'affichage n'est plus représentatif de la réalité, une partie de l'information est perdue puisqu'un test apparaissant en succès peut en fait ne pas l'être.

Sauf qu'actuellement seuls les defects rentrés sur Jira sont pris en compte, la mécanique en place ne permet pas encore de récupérer ces mêmes informations de CWB. Puisque de plus en plus de defects seront enregistrés sur CWB, et dans un soucis d'homogénéisation du process actuel, il faut rationaliser le process actuel et permettre au statuts d'être ajustés quelque soit l'origine du defect (Jira ou CWB).

---

3. Typiquement, un job correspond à un projet logiciel

4. Error, Failure, Unstable, Aborted, Not built, Success

5. statuts Jenkins

### 3.3 Process actuel

1- Investigation du process actuel

la méthode apply de JUnit est surchargée  
il y a une annotation qui permet de checker le status  
l'annotation est prise en compte en fonction d'un paramètre -D

même résultat pour acceptance et GTP ???? -» fabien

Dans l'état actuel des choses le process utilisé est illustré figure 3.1 page 21

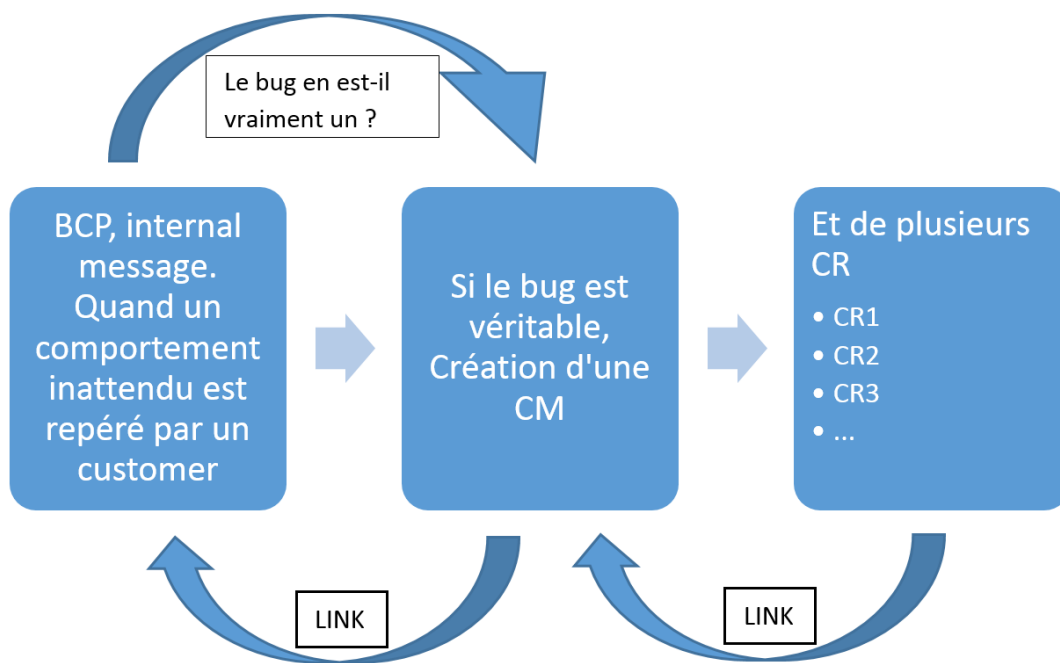


FIGURE 3.1 – Le process utilisé lors de gestion d'un bug

#### En résumé

- JUnit est surchargé par la classe JiraIssueWatcher.java
- Je vais devoir utiliser le système choisi par SAP pour fournir les credentials : prod-pass-access
- Il existe une API permettant d'obtenir les informations que je désire : CWB HTTP API

### 3.4 Travail effectué

La première partie du travail s'est cantonnée à des recherches et des échange de mails internes à SAP.

!!!! Ici je décris mes différents mails avec marco et Fabien!!!!

#### 3.4.1 Fonctionnement et utilisation de prod-pass-access

### 3.5 Résultats obtenus

Maintenant la mission précédente achevée, nous pouvons récupérer les infos de status quelque soit son emplacement (Jira ou CWB) depuis test-defects.xml (cf Migration)

-> connaissances Java J'ai approfondi mes connaissances sur les annotations et la manière de les implémenter. J'ai implémenté ma première classe abstraite

# Chapitre 4

## Plugin Jenkins

### 4.1 Contexte initial

Jenkins est le logiciel d'intégration continue, il est utilisé à SAP dans le contexte des tests. Son rôle est d'intégrer les différents projets, tels que configuré par l'utilisateur, et met à disposition les résultats obtenus. Il permet d'avoir un retour régulier sur l'état des builds et du résultat de leurs tests.

Un plugin permettant une vision globale sur l'état des builds est déjà en place, Radiator. La figure 4.1 page 24 présente son environnement d'utilisation. Nous pouvons y voir les développeurs et les ST qui alimentent le code hébergé sur Perforce que Jenkins l'utilise dans ses diverses intégrations. Au centre, le plugin permet un retour visuel rapide des livraisons Perforce.

Ce plugin est aussi conçu pour fonctionner avec un autre plugin, le plugin Claim. Il permet à quelqu'un visitant la page d'un job en échec de le « claimer », inscrivant son identifiant ainsi qu'un message à destination de quiconque visitera cette même page. Ceci permet de signaler que l'investigation est en cours.

Pour résumer ses caractéristiques principales, c'est un plugin qui permet :

- d'afficher un écran où chaque couleur correspond à un statut de build
- de rassembler les jobs par leurs préfixes communs
- de prendre en compte les claims

L'inconvénient est que le nombre de status existant ne permet pas la précision nécessaire pour définir l'état d'un job. Le plugin Radiator n'offre pas la malléabilité nécessaire. Voici la liste des différents états gérés par ce plugin :

**SUCCESS** Le projet compile correctement et tous les tests sont passés

**ABORTED** La compilation a été arrêtée en cours d'exécution

**FAILURE** Le code ne compile pas

**ERROR**

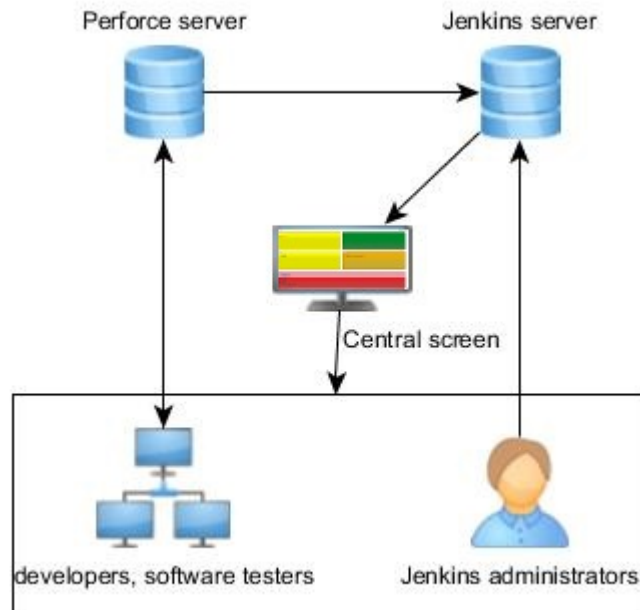


FIGURE 4.1 – L’environnement d’utilisation du plugin

**NOT BUILT** Le projet n’a pas été compilé

**UNSTABLE** Des tests JUnit ne sont pas passés mais le code compile

**CLAIM** Le job en échec a été claim par quelqu’un

Tel qu’il se présente le plugin ne permet pas de donner des informations autres que celles issues de Jenkins ou du plugin Claim. Il ne permet donc pas de donner de renseignements sur le defect associé à ce job. Tenir compte du defect permettrait de faire ressortir les régressions et autres erreurs, séparant ainsi les échecs pris en charge de ceux qui ne le sont pas.

Suite à la mission précédente nous avons maintenant accès aux informations relatives aux defects (celle-ci sont enregistrées dans les archives des builds sur le serveur Jenkins), dans un fichier XML que nous avons nommé test-defects (exemple page 25).

### La solution proposée

Dans un 1<sup>er</sup>, et en tant qu’exercice, il faudra réaliser un plugin offrant les mêmes possibilités d’affichage des builds. C’est-à-dire rassembler les jobs par préfixe et afficher le résultat général de tous ces groupes ainsi créés sur un unique écran. Ajouté aux caractéristiques, ce plugin doit pouvoir prendre en compte les informations du plugin Claim.

Dans un second temps, il faudra pouvoir ajouter un ou plusieurs statut(s) supplémentaire(s) ainsi que de ré-ajuster leur ordre de priorité. L’ajout de ce nouveau statuts doit être générique de sorte qu’un status quelconque puisse être ajouté.

Le plugin est donc capable de recueillir des informations dans des fichiers sans savoir, au

préalable, comment seront structurées ses données. Rappelons que lors de l'exécution des tests, un fichier test-defect est généré contenant la liste<sup>1</sup> des jobs avec defect (exemple page 25).

```

1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
  <testcases>
3    <testcase classname="test.sap.sl.sdk.authoring.IssueWatcherTest" name="
      failingTestNoDefect" status="failed"/>
    <testcase classname="test.sap.sl.sdk.authoring.IssueWatcherTest" name="
      successTestNoDefect" status="passed"/>
5    <testcase actualStatus="failed" classname="test.sap.sl.sdk.authoring.
      IssueWatcherTest" message="Error" name="failingTest3_ClassDefect" status="
      passed">
      <defect expectedMessage="Error" link="https://sapjira.wdf.sap.corp/browse/
        BITBIWEBISL2-1542" status="Open" type="jira"/>
7    </testcase>
    <testcase actualStatus="failed" classname="test.sap.sl.sdk.authoring.
      IssueWatcherTest" message="Error" name="failingTest2_ClassDefect" status="
      failed">
9      <defect expectedMessage="Failing test" link="https://css.wdf.sap.corp/sap/
        bc/bsp/spn/jcwb?crPointer=012006153200001159182015" status="In Process"
        type="cwb"/>
    </testcase>
11    <testcase actualStatus="failed" classname="test.sap.sl.sdk.authoring.
      IssueWatcherTest" message="Failing test" name="
      failingTest_CWB_MatchingMessage" status="passed">
      <defect expectedMessage="Failing test" link="https://css.wdf.sap.corp/sap/
        bc/bsp/spn/jcwb?crPointer=012006153200001159182015" status="In Process"
        type="cwb"/>
13    </testcase>
    <testcase actualStatus="failed" classname="test.sap.sl.sdk.authoring.
      IssueWatcherTest" message="Failing test" name="failingTest1_ClassDefect"
      status="passed">
15      <defect expectedMessage="Failing test" link="https://css.wdf.sap.corp/sap/
        bc/bsp/spn/jcwb?crPointer=012006153200001159182015" status="In Process"
        type="cwb"/>
    </testcase>
17    <testcase actualStatus="failed" classname="test.sap.sl.sdk.authoring.
      IssueWatcherTest" message="Failing test" name="
      failingTest_Jira_MatchingMessage" status="passed">
      <defect expectedMessage="Failing test" link="https://sapjira.wdf.sap.corp/
        browse/BITBIWEBISL2-1542" status="Open" type="jira"/>
19    </testcase>
    <testcase actualStatus="failed" classname="test.sap.sl.sdk.authoring.
      IssueWatcherTest" message="Error" name="failingTest_CWB_NoMatchingMessage"
      status="failed">
21      <defect expectedMessage="Failing test" link="https://css.wdf.sap.corp/sap/

```

---

1. Au format xml

```

    bc/bsp/spn/jcwb?crPointer=012006153200001159182015" status="In Process"
    type="cwb"/>
  </testcase>
23 </testcases>

```

## 4.2 Étude préalable

Les 1<sup>er</sup> impératifs ont été de maîtriser le langage et les outils qui me permettrai de mener ce projet à terme. Je n'avais jamais entendu parler de Jenkins ou d'un quelconque logiciel d'intégration continue, et a fortiori sur la manière de l'étendre.

Il existe un archétype maven (cf annexe E page 51) permettant de générer le squelette d'un plugin Jenkins

J'ai donc commencé par installer Jenkins et ses plugins, tels qu'ils sont utilisés à SAP, de manière à me familiariser avec ceux-ci. Pour examiner le comportement de Jenkins j'ai créé de faux projets me permettant de simuler les différents états possibles. Ceux-ci m'ont permis d'observer les différences entre les statuts Jenkins et JUnit, certains statuts portent les mêmes noms mais ne signifient pas les mêmes choses provoquent quelques incompréhensions. La signification de ces statuts étant l'objet de ce plugin, il m'incombait de les comprendre. Voici le détail des configurations que j'ai testé pour déterminer l'équivalence statut JUnit/statut Jenkins :

Description du projet	Status JUnit	Statut Jenkins
projet vide et sans test	NA	Success
projet vide et avec test réussi	Passed	Success
projet vide avec test qui échoue	Failure	Unstable
projet avec erreur de compilation	NA	Error

Passée cette étape d'installations et de configurations je me suis penché sur la marche à suivre pour implémenter un plugin Jenkins.

La principale difficulté que j'ai eu à développer ce plugin a été de trouver l'information, il est reconnu sur internet que la documentation de Jenkins, lorsque l'on veut l'étendre, n'est pas claire. Mais la documentation officielle<sup>2</sup> offre un bel état des lieux de ce qu'il est possible de faire, décrit les différentes technologies constituant Jenkins et offre un tutoriel, trivial mais suffisant, pour comprendre les bases.

## 4.3 Le 1<sup>er</sup> plugin

Je me suis d'abord rendu sur la page du tutoriel officiel de Jenkins<sup>3</sup> afin d'obtenir un plugin fonctionnel avec lequel je pouvais faire mes expériences. Ensuite j'ai suivi un

2. <https://wiki.jenkins-ci.org/display/JENKINS/Extend+Jenkins>

3. <https://wiki.jenkins-ci.org/display/JENKINS/Plugin+tutorial>



autre tutoriel<sup>4</sup> non-officiel mais considérablement plus riche.

### 4.3.1 Le squelette du plugin

#### Configuration

D'après toutes les documentations disponible, le meilleur moyen de commencer le développement du plugin est d'utiliser maven<sup>5</sup> pour générer le squelette. Maven a besoin de configurations supplémentaires dans son settings.xml<sup>6</sup> pour pouvoir récupérer les sources relatives à Jenkins.

Dans « pluginGroups » il faut ajouter la ligne suivante :

```
1 <pluginGroup>org.jenkins-ci.tools</pluginGroup>
```

Et dans « profiles » il faut ajouter le profil suivant :

```
1 <profile>
  <id>jenkins</id>
3 <activation>
  <activeByDefault>false</activeByDefault>
5 </activation>
  <repositories>
7 <repository>
  <id>repo.jenkins-ci.org</id>
9 <url>http://repo.jenkins-ci.org/public/</url>
  </repository>
11 </repositories>
  <pluginRepositories>
13 <pluginRepository>
  <id>repo.jenkins-ci.org</id>
15 <url>http://repo.jenkins-ci.org/public/</url>
  </pluginRepository>
17 </pluginRepositories>
</profile>
```

L'attribut de la balise « activeByDefault » est à false, parcequ'on travaille avec d'autres profils sur d'autres projet. Mais il faut retenir que ce paramètre peut empêcher la génération du squelette du plugin via la commande :

```
mvn hpi:create
```

Pour exécuter cette commande ci-dessus il faut se placer dans le dossier qui recevra le projet.

---

4. <https://cleantestcode.wordpress.com/2013/11/03/how-to-write-a-jenkins-plugin-part-1/>

5. Maven 3 et JDK 6 ou plus récent

6. Situé dans le répertoire ~/.m2

### Génération du squelette

À l'exécution de cette commande il est demandé de renseigner le groupId et l'artifactId, le groupId peut être org.jenkins-ci.plugins et l'artifactId est le nom du plugin. Cette commande crée l'arborescence du projet ainsi que les fichiers de base.

À la génération du squelette du plugin nous obtenons un plugin qui compile dont l'architecture est présentée figure 4.2 page 28. Certaines choses importantes sont à noter, le pom.xml<sup>7</sup> est généré à la base du projet dans lequel nous trouvons des informations telles que le parent de ce projet, les dépendances, l'auteur, la licence et la description.

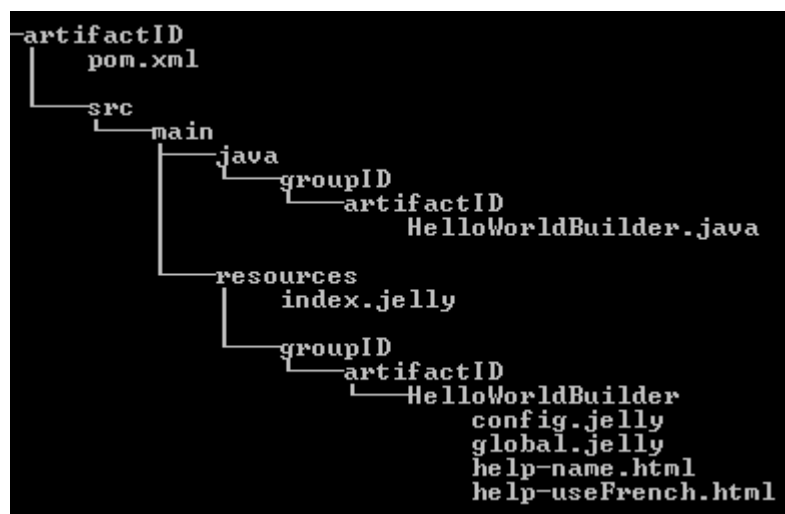


FIGURE 4.2 – Architecture d'un plugin Jenkins généré par maven

!!!!!! ok, maintenant on sais faire un plugin hello world

Plusieurs autre fichiers sont générés :

Nom du fichier	Description
index.jelly	Description utilisée à la création d'une nouvelle instance du plugin
config.jelly	Correspond à la page d'une instance du plugin
global.jelly	Correspond à la page de configuration générale du plugin
HelloWorldBuilder.java	Implémentation d'exemple du plugin
help-name.html	Fichier d'aide à la configuration lorsque clic sur le symbole d'aide
help-useFrench.html	Fichier d'aide à la configuration lorsque clic sur le symbole d'aide

!!! Tous les fichiers index.jelly config.jelly global.jelly

configure-entries.jelly main.jelly properties help

!!!Les points d'extensions

### Déploiement du squelette du plugin

Plusieurs solutions s'offre à nous pour déployer le plugin dans son environnement d'utilisation

#### Installation « à la main »

En ligne de commande et depuis la racine du projet <sup>8</sup>, il faut exécuter la commande suivante :

```
1 mvn clean install -Pjenkins
```

Celle-ci va télécharger les sources nécessaires, les compiler, exécuter les tests et, si tout se passe bien, générer le fichier .hpi dans le dossier target.

Ceci fait, il faut se rendre dans Jenkins (par exemple <http://localhost:8080/> ou autre. En fonction de la configuration), s'identifier et aller dans les paramètres avancés de gestion de plugins. Ici il faut renseigner le chemin d'accès vers le .hpi et terminer.

#### Commande du plugin Jenkins de maven

En ligne de commande et depuis la racine du projet, il faut exécuter la commande suivante :

```
1 mvn hpi:run -Djetty.port=8090 -Pjenkins
```

Cette commande va déployer une instances de Jenkins propre à maven, ce qui nous permet d'exécuter le plugin ainsi que de le debugger. On peut très bien omettre de préciser le port utilisé si celui par défaut <sup>9</sup> est disponible, mais il vaut mieux éviter. Une fois que le déploiement est terminé maven signale que « Jenkins is fully up and running » nous pouvons aller voir Jenkins à l'url <http://localhost:8080/jenkins> qui devrait ressembler à la figure 4.3 page 30.

### Essaie du squelette du plugin

Jenkins se redémarre et voilà notre plugin intégré à celui-ci. Nous pouvons aller voir l'écran de configuration générale vérifier que le fichier global.jelly a bien été ajouté, on coche la case. Maintenant configurons un projet au hasard et ajoutons une étape pré build, nous apercevons un item s'appelant « say hello world », essayons et remplissons le champ de texte. Compilons le projet et regardons la sortie console, au début nous voyons affiché notre texte.

Nous avons donc un plugin qui permet de dire quelque chose en début ou fin de build.

## 4.4 Implémentation de la base du nouveau plugin

Nous avons donc un plugin qui étend le comportement lors des builds que Jenkins exécute, ceci grâce au point d'extension Builder.

---

8. à l'emplacement du POM.xml

9. port 8080



FIGURE 4.3 – Jenkins à l'exécution de la commande maven qui l'encapsule

Jenkins possède beaucoup de points d'extensions et à cette étape il faut déterminer lequel, ainsi que quelle classe étendre ! Dans le cas où aucun point d'extension ne correspond, il est très facile d'en créer un en suivant la documentation. Autrement, nous pouvons nous inspirer du plugin Radiator et regarder quelle classe il étend.

En parcourant la liste des points d'extensions disponibles nous pouvons trouver *View*, dont l'une de ses implémentations est *ListView* ; sa description étant « Displays Jobs in a flat list view ». Notre objectif étant d'afficher les Jobs, cette solution semble correspondre. De plus, en observant la javadoc on peut trouver la méthode nous donnant accès aux projets associés à la vue (figure 4.4 page 30).

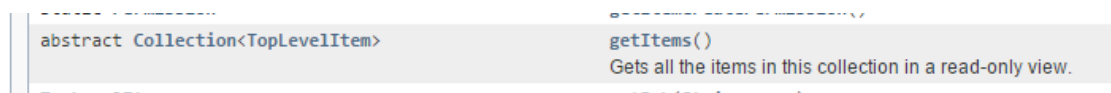


FIGURE 4.4 – Extrait de la javadoc de la classe ListView

### 4.4.1 Import du code dans l'IDE

J'ai expérimenté deux IDE pour développer le plugin : Eclipse puis NetBeans. En utilisant Eclipse j'ai rencontré beaucoup de problèmes liés aux configurations maven et d'Eclipse, de telle sorte qu'à un certain moment je me trouvais obligé de compiler le code via Jenkins, de récupérer le .hpi en local pour l'intégrer ensuite à Jenkins ! Le cycle de développement était le suivant :

1. Implémentation du code sous Eclipse
2. Compilation du code grâce à maven (ou Jenkins !) et génération du fichier .hpi
3. Désinstallation du plugin de Jenkins + redémarrage
4. Installation du nouveau plugin sur Jenkins + redémarrage
5. Test du plugin

L'utilisation de Netbeans (et de son plugin Jenkins) évite de passer par toutes ses étapes car il suffit de cliquer sur run pour exécuter le plugin et sur debug pour le debugger. Cela facilite l'implémentation du code et, surtout, permet de se focaliser sur les problèmes liés au plugin plutôt qu'à ceux liés aux outils de développement.

### 4.4.2 Implémentation de la base

Avant de pouvoir vraiment travailler à l'implémentation des fonctionnalités du logiciel il faut encore savoir où écrire le code et comment nommer ses fichier. Une fois que Jenkins à localiser l'annotation précisant qu'il s'agit d'un plugin, il utilise une convention de nommage pour accéder aux différents fichiers d'aide ou de configuration. Pour savoir comment doit se nommer mon fichier de configuration je vais chercher où, dans jenkins-core, se trouve les fichiers de configuration de Listview. Ce qui me m'offrira la possibilité d'implémenter ma propre configuration. L'arborescence de ce fichier est illustrée figure 4.4 page 30. Il convient maintenant de copier ce fichier et de le coller dans le répertoire *ressources* de notre projet.

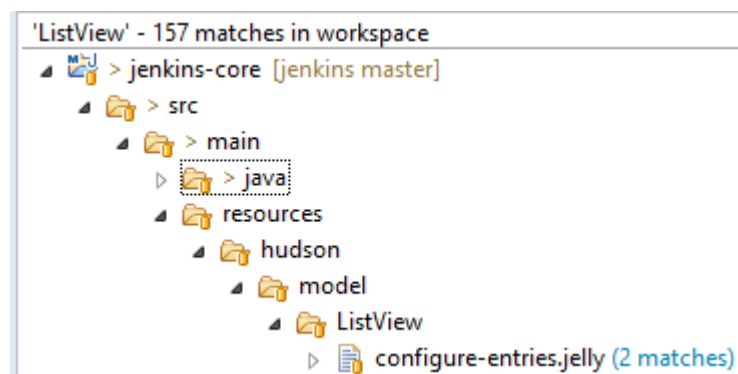


FIGURE 4.5 – Chemin d'accès au fichier de configuration par défaut

Maintenant que l'on peut personnaliser la configuration du plugin, il faut pouvoir faire de même avec l'affichage de la vue. Nous allons donc créer un nouveau fichier *main.xml* qui contiendra le code de l'interface graphique du plugin.

Après quelques tests rapides on s'aperçoit que beaucoup de code, css ou JavaScript, est exécuté lorsque l'on accède à la page. Il faut donc réinitialiser pour travailler dans de bonnes conditions. Premièrement, on supprime l'affichage du header, footer et side-panel de Jenkins (cf figure 4.6 page 32) et deuxièmement on supprime l'affichage du side-panel, après chargement de la page (cf figure 4.7 page 32).

```
<style type="text/css">
  #header{
    display:none;
  }
  #side-panel{
    display:none;
  }
  #view-message{
    display:none;
  }
</style>
```

FIGURE 4.6 – Initialisation des css avant chargement

```
$("#side-panel").removeClass("col-sm-9");
$("#side-panel").removeClass("col-md-7");
$("#side-panel").removeClass("col-lg-6");
$("#side-panel").removeClass("col-xlg-4");
```

FIGURE 4.7 – Initialisation des css après chargement

## 4.5 Travail réalisé

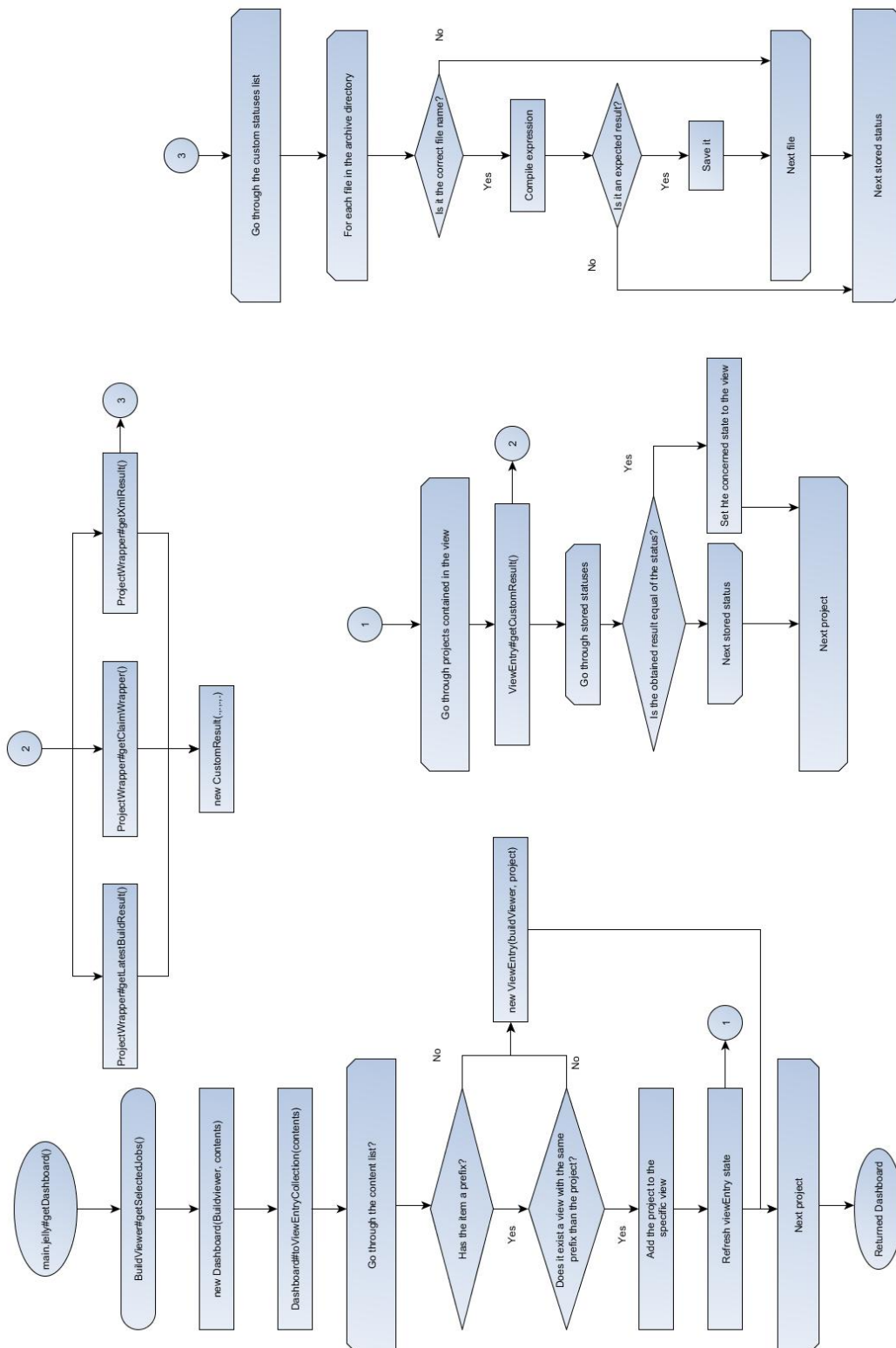
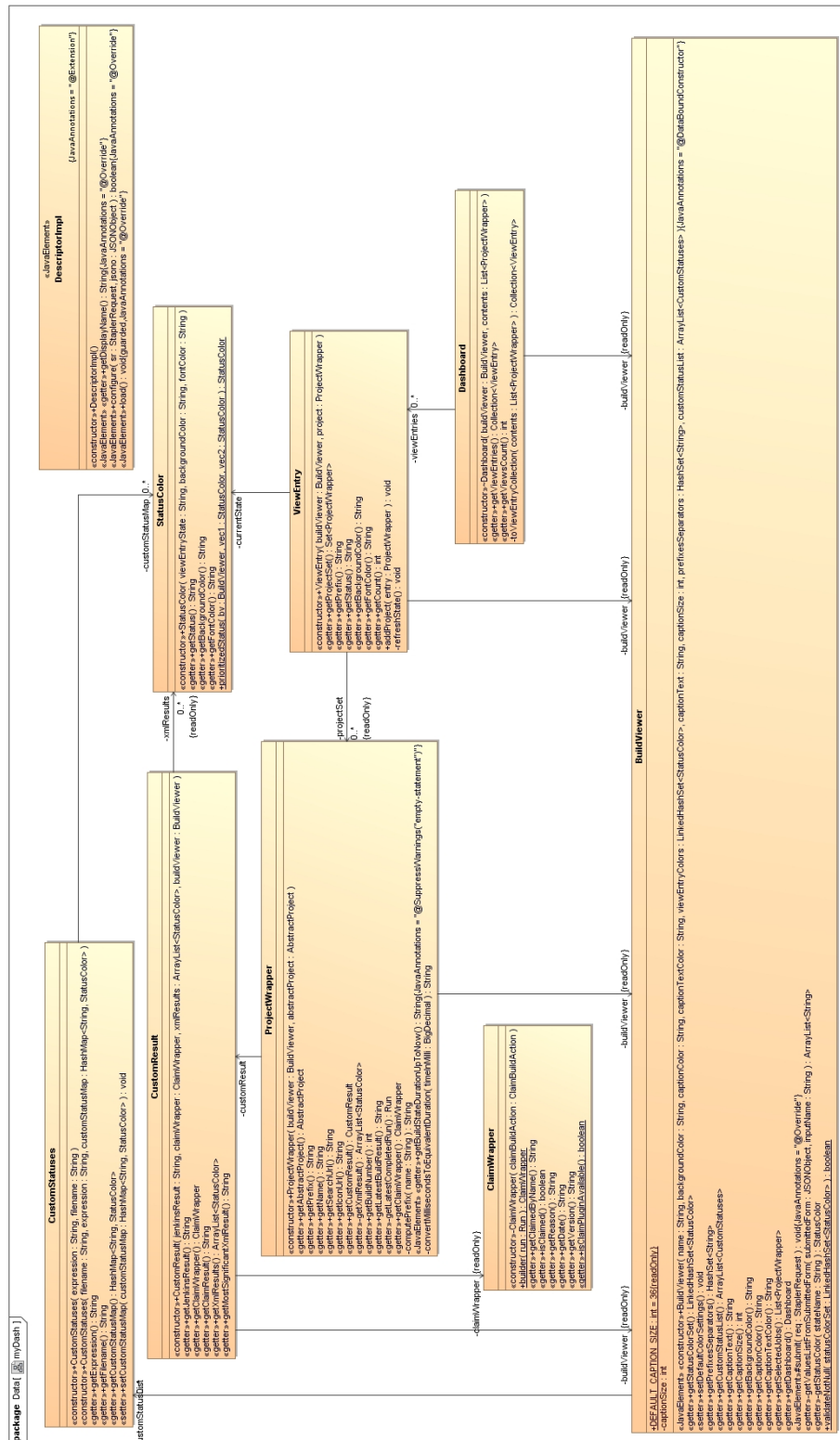


FIGURE 4.8 – Algorithme de création du Dashboard





# Conclusion

Mon contrat professionnel à SAP fût une expérience remarquable tant par la valeur de l'équipe que j'ai intégré que par les missions que je me suis vu effectuer.

M'a permis de découvrir l'environnement complexe d'une multinationale et les outils mis à disposition des employés pour permettre une bonne communication au sein de l'entreprise. J'ai pu me rendre compte de l'ampleur d'un projet comme WebI ainsi que toutes les ressources impliquées dans la conduite de celui-ci.

Les apports de ma mission pour l'entreprise ? L'avenir du projet dans l'entreprise. Un prolongement éventuel ? Une proposition refusée



Les niveaux de test

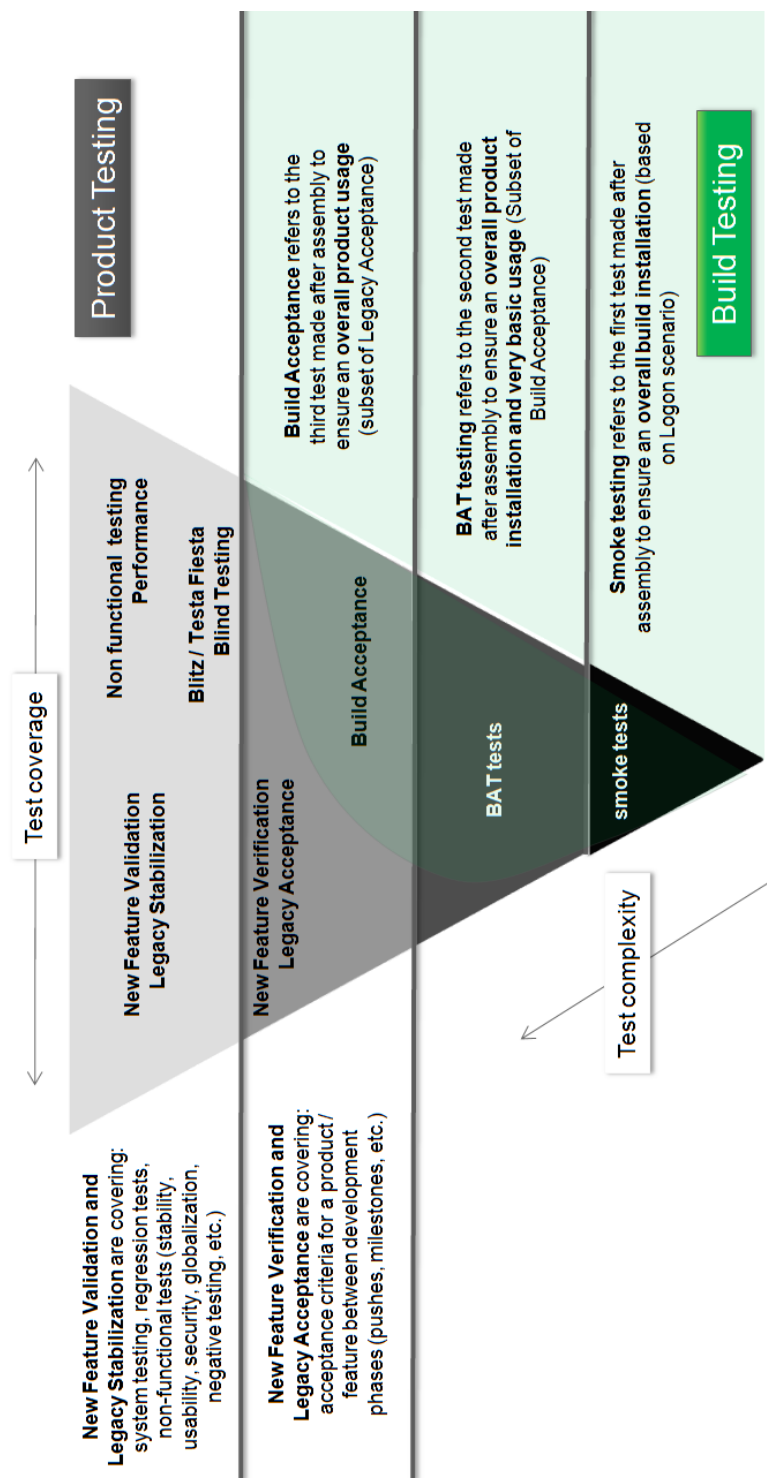


FIGURE A.1 – Les différents niveaux de test



## Résultats quotidien des tests

From: JUnit@pgdev.sap.corp  
Sent: lundi 17 août 2015 18:20  
To: DOLIMONT, Christophe; TAQUET, Pierre; COULIBALY, Nana  
Subject: JUnit 99.29% Build=aurora\_dev\_webi\_dsl\_331 Test=rebean\_wi.customers.TestSuite\_Customers Computer= 10.208.43.203

TEST = rebean\_wi.customers.TestSuite\_Customers  
SUITE = Aurora 4.2  
SERVERNAME = localhost:6400  
OS name = Windows  
OS archi = amd64  
PHASE = D2P win64\_x64  
BUILDNAME = 14.2.0.331.businessobjects64\_aurora\_dev\_webi\_dsl  
VAPP NAME = ASTEC\_AURORA42\_DEV\_WIN\_f0c68b77-27f4-475a-843e-35896aafec4

## Unit Test Results

Designed for use with [JUnit](#) and [Ant](#).

### Summary

Tests	Failures	Errors	Success rate	Time
140	1	0	99.29%	1 h 31 min 50 s

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

### Packages

Note: package statistics are not computed recursively, they only sum up all of its testsuites numbers.

Name	Tests	Errors	Failures	Time	Time Stamp	Host
<a href="#">rebean_wi.customers</a>	140	0	1	1 h 31 min 50 s	2015-08-17T14:47:39	server41

### Package rebean\_wi.customers

Name	Tests	Errors	Failures	Time	Time Stamp	Host
<a href="#">TestSuite_Customers</a>	140	0	1	1 h 31 min 50 s	2015-08-17T14:47:39	server41

### TestCase TestSuite\_Customers

Name	Status	Type	Time
rebean_wi.reporting_rendering.validation.Init_TC_itbi init_PrepForITBI	Success		17 s
rebean_wi.customers.ADP_GSI_France.ID_225480.ID_225480_TestCase_1 CM_1261231_2014_CellHeightUpdate	Success		30 s
rebean_wi.customers.Aerospatiale_Matra_Airbus.ID_187739.CM_627746_2014 CM_627746_2014_Computation	Success		3 s
rebean_wi.customers.Aerospatiale_Matra_Airbus.ID_187739.CM_731851_2014 CM_731851_2014_QuerySummary	Success		2 s
rebean_wi.customers.Amdocs_Development_LTD.ID_871420.CM_654663_2014 CM_654663_2014_Unavailable	Success		2 s
rebean_wi.customers.American_Management_Systems.ID_977175.ID_977175_TestCase_1 CSSID_0020079747_D000035772_2014	Success		3 s
rebean_wi.customers.American_Management_Systems.ID_977175.ID_977175_TestCase_2 CM_1330059_2014_PDF_Truncated	Success		3 min 50 s
rebean_wi.customers.AnnTaylor_Inc.ID_570535.ADAPT01716536_Break_FormatText ADAPT01716536_Excel_Break_FormatText	Success		3 s
rebean_wi.customers.AOK_Baden_Wuerttemberg.ID_101715.CM_254828_2015 CM_254828_2015_SplitColumsPageMode	Success		
rebean_wi.customers.Apple.ID_30934.ADAPT01717053_Multivalue_CrossTable ADAPT01717053	Success		3 s
rebean_wi.customers.Apple.ID_30934.CM_661468_2014 CM_661468_2014_Subtotals	Success		3 s
rebean_wi.customers.Apple.ID_30934.CM_48299_2015 CM_48299_2015_RelativePositionMismatch	Success		5 s
rebean_wi.customers.Aquitanis_Opac_De-La_Communaute.ID_944713.CM_1240269_2014_TP CM_1240269_2014_FormatNumber	Success		11 s
rebean_wi.customers.ARZ_Allgemeines_Rechenzentrum.ID_28259.ID_28259_TestCase_1 CM_1314984_2014_Export_BIG_Excel	Success		10 min 17 s
rebean_wi.customers.Asahi_Glass.Co.ID_145381.CM_1452299_2014_BreakError CM_1452299_2014_ScanBreakError	Success		2 s
rebean_wi.customers.Atlas_Copco_Business_Services.ID_642864.ADAPT01718277_Input_Controls_Lost ADAPT01718277_Input_Control_Lost	Success		26 s
rebean_wi.customers.Audi.AG.ID_12520.ADAPT01699626_PDF_Truncated ADAPT01699626_PDF_Truncated_Test	Success		6 s
rebean_wi.customers.Audi.AG.ID_12520.ADAPT01706987_ChartLegacyDefaultSettings ADAPT01706987	Success		4 s
rebean_wi.customers.Audi.AG.ID_12520.AUDI_1386567_2014_InputControlsUpgrade CM_1386567_2014_InputControlsUpgrade	Success		3 s
rebean_wi.customers.Australian_Health_And_Nutrition.ID_336882.CM_1562604_2014 CM_1562604_2014_SortIssue	Success		2 s
rebean_wi.customers.Axa_Group_Solutions.ID_890374.CM_3413_2015_TP CM_3413_2015_CustomSortIssue	Success		4 s
rebean_wi.customers.Bank_for_International_Settlements.ID_25735.CM_1511454_2014_TP CM_1511454_2014_CellAlignment	Success		3 s
rebean_wi.customers.Bank_Julius_Bar.ID_133577.CM_1542477_2014 CM_1542477_2014_AutoFIHTML	Success		2 s
rebean_wi.customers.Bank_of_America.ID_1166346.CM_704895_2014 CM_704895_2014_Formatting	Success		3 s
rebean_wi.customers.Cenovus_Energy_Inc.ID_1145846.CM_883660_2014_Config CM_883660_2014_ConfigVisuFile	Success		
rebean_wi.customers.Cenovus_Energy_Inc.ID_1145846.CM_883660_2014_Tomcat CM_883660_2014_TomcatRestart	Success		5 s
rebean_wi.customers.Cenovus_Energy_Inc.ID_1145846.CM_883660_2014_Config CM_883660_2014_ConfigVisuFile	Success		
rebean_wi.customers.Cenovus_Energy_Inc.ID_1145846.CM_883660_2014_Tomcat CM_883660_2014_TomcatRestart	Success		5 s
rebean_wi.customers.ConocoPhillips_Company.ID_161451.CM_110784_2015 CM_110784_2015_TP	Success		2 s
rebean_wi.customers.Biogen_IDEC.Inc.ID_302254.CM_677911_2014_TP CM_677911_2014_ERR_WJ1_20003	Success		3 s
rebean_wi.customers.Bouygues_Construction.ID_518883.CM_748730_2014_TP CM_748730_2014_CantCreateKeyFigure	Success		2 s
rebean_wi.customers.Bouygues_Construction.ID_518883.CM_124508_2015 CM_124508_2015_WrongBreakFooterValues	Success		2 s
rebean_wi.customers.British_American_Tobacco.ID_143839.CM_1432086_2014 CM_1432086_2014_MissingValues	Success		1 min 36 s
rebean_wi.customers.Cactus_SA.ID_185193.CM_24874_2015 CM_24874_2015_NavigationError	Success		4 s
rebean_wi.customers.Cedar_Sinai_Medical_Center.ID_977698.CM_237748_2015	Success		3 s



ADAPT01711625		
rebean_wi.customers.Novartis_Vaccines_And_Diagnostics.ID_184701.CM_787792_2014_TP CM_787792_2014_XLSX_Date	Success	2 s
rebean_wi.customers.Novartis_Vaccines_And_Diagnostics.ID_184701.CM_793170_2014_TP CM_793170_2014_Hyperlinks	Success	5 min 41 s
rebean_wi.customers.Ntt_Docomo.ID_997488.CM_138459_2015_TP_Init CM_138459_2015_Init_1	Success	
rebean_wi.customers.Ntt_Docomo.ID_997488.CM_138459_2015_TP CM_138459_2015_Drill_Snapshot	Success	2 s
rebean_wi.customers.Ntt_Docomo.ID_997488.CM_138459_2015_TP_Reset CM_138459_2015_3_Reset	Success	
rebean_wi.customers.Odyssey_America_Reinsurance.Co.ID_945278.ID_945278_TestCase_1 CM_1446291_2014_UnableToHide	Success	2 s
rebean_wi.customers.PepsiCo_Inc.ID_692630.ID_692630_TestCase_1 CM_1326726_2014_MissingColumns_Doc1	Success	1 min 16 s
rebean_wi.customers.PepsiCo_Inc.ID_692630.ID_692630_TestCase_1 CM_1326726_2014_MissingColumns_Doc2	Success	5 s
rebean_wi.customers.PepsiCo_Inc.ID_692630.ID_692630_TestCase_1 CM_1326726_2014_MissingColumns_Doc3	Success	1 min 21 s
rebean_wi.customers.PepsiCo_Inc.ID_692630.ID_692630_TestCase_1 CM_1326726_2014_MissingColumns_Doc4	Success	5 s
rebean_wi.customers.PepsiCo_Inc.ID_809237.ID_809237_TestCase_1 CM_1431229_2014_MissingValues	Success	5 s
rebean_wi.customers.Pfizer_Inc.ID_933432.CM_644030_2014 CM_644030_2014_RunningSum	Success	5 s
rebean_wi.customers.Pfizer_Inc.ID_933432.CM_835743_2014 CM_835743_2014_CalcIssue	Success	6 s
rebean_wi.customers.Pfizer_Inc.ID_933432.CM_1504791_2014 CM_1504791_2014_XLSX_Header	Success	7 s
rebean_wi.customers.Promega_Corporation.ID_824970.ID_824970_TestCase_1 CSSID_0020079747_0001161381_2013_DrillUpdateSet	Success	5 s
rebean_wi.customers.Qatar_Airways.ID_964262.CM_1445386_2014_TP CM_1445386_2014_noConsistentData	Success	3 s
rebean_wi.customers.Ransa_Comercial.SA.ID_881393.CM_216746_2015 CM_216746_2015	Success	10 s
rebean_wi.customers.Ransa_Comercial.SA.ID_881393.CM_216746_2015_HierarchicalNavigation CM_216746_2015_HierarchicalNavigation	Success	2 s
rebean_wi.customers.Robert_Bosch_GmbH.ID_10010.CM_710039_2014 CM_710039_2014_ExcelExportCrash	Success	6 s
rebean_wi.customers.Roberto_Cavalli.ID_969639.CM_1539874_2014 CM_1539874_2014_NoReport	Success	3 s
rebean_wi.customers.Ruokakesko_Oy.ID_512535.ID_512535_TestCase_1 CSSID_0120025231_0000476351_2014_ExportCSV	Success	2 s
rebean_wi.customers.Shell.ID_22183.ADAPT017114504 ADAPT017114504_checkQueryStrippingProperty	Success	2 s
rebean_wi.customers.Shell.ID_22136.ADAPT01676242_Duplicated_Rows_Merged_Dim ADAPT01676242	Success	3 s
rebean_wi.customers.Shell.ID_949782.CM_836429_2014_TP CM_836429_2014_AggregationIssue	Success	2 s
rebean_wi.customers.Shell.ID_22183.CM_922967_2014 CM_922967_2014_ToRefresh	Success	3 s
rebean_wi.customers.Smith_Nephew_North_America.ID_192290.CM_204685_2015 CM_204685_2015	Success	2 s
rebean_wi.customers.Socpresse.ID_458545.CM_693446_2014_TP CM_693446_2014_IncorrectSum	Success	3 s
rebean_wi.customers.Sony_Pictures_Entertainment.ID_185212.CM_858676_2014 CM_858676_2014_XLSX_SectionIssue	Success	5 s
rebean_wi.customers.Sony_Pictures_Entertainment.ID_185212.CM_1503492_2014 CM_1503492_2014_RankIssue	Success	5 s
rebean_wi.customers.Sony_Pictures_Entertainment.ID_185212.CM_79465_2015 CM_79465_2015_Reccord_Mismatch	Success	18 min 19 s
rebean_wi.customers.Southern_Wine_Spirits_Of_America.ID_492420.ID_492420_TestCase_1 CSSID_0120025231_0001134551_2013	Success	4 min 29 s
rebean_wi.customers.SUISAG_AG_fur_Dienstleistungen.ID_1009623.CM_747939_2014_TP CM_747939_2014_ExportFormat	Success	2 s
rebean_wi.customers.Tata_Consultancy_Services_Ltd.ID_31117.CM_1509927_2014 CM_1509927_2014_OutlineIssue	Success	4 s
rebean_wi.customers.Technotrans_AG.ID_834183.ID_834183_TestCase_1 CM_1368118_2014_ErrorColumns	Success	3 s
rebean_wi.customers.Technotrans_AG.ID_834183.Cm_761697_2014_TP CM_761697_2014_HeaderIssue	Success	2 s
rebean_wi.customers.Texas_Instruments_Incorporated.ID_37915.CM_707176_2014 CM_707176_2014_RsError_TC1	Success	2 s
rebean_wi.customers.Texas_Instruments_Incorporated.ID_37915.CM_707176_2014 CM_707176_2014_RsError_TC2	Success	3 s
rebean_wi.customers.Texas_Instruments_Incorporated.ID_37915.CM_799629_2014 CM_799629_2014_ExcelImageName	Success	6 s
rebean_wi.customers.The_Bank_of_Tokyo.ID_347072.CM_1156790_2014_InitTC CM_1156790_2014_Init	Success	14 s
rebean_wi.customers.The_Bank_of_Tokyo.ID_347072.ID_347072_TestCase_1 CM_1156790_2014_MaxCharacterStreamSize	Success	26 s
rebean_wi.reporting_rendering_validation.Init_TC_Ibtl Init_PrepaForITBI	Success	13 s
rebean_wi.customers.The_Hartford_Financial_Services.ID_353355.CM_45776_2015 CM_45776_2015_ReportFilterIssue	Success	18 s
rebean_wi.customers.The_Lubrizol_Corporation.ID_197246.ID_197246_TestCase_1 CM_1383807_2014_FormattedValuesErrors	Success	3 s
rebean_wi.customers.The_Procter_And_Gamble_Company.ID_27345.CM_187536_2015 CM_187536_2015_MultiValue	Success	52 s
rebean_wi.customers.The_Procter_And_Gamble_Company.ID_27345.CM_127265_2015 CM_127265_2015_XCELMissingParts	Success	5 s
rebean_wi.customers.The_Standard_Bank_of_South_Africa.Ltd.ID_126497.CM_1573726_2014_TP CM_1573726_2014_CollapseHierarchy	Success	2 s
rebean_wi.customers.TIS_Inc.ID_300998.CM_67357_2015_TP CM_67357_2015_Missing_Record_CDP	Success	8 s
rebean_wi.customers.Union_Gas_Limited.ID_195098.CM_1288179_2014 CM_1288179_2014_CustomSqlNotEnable	Success	2 s
rebean_wi.customers.University_of_Queensland.ID_966000.CM_76245_2015_TP CM_76245_2015_ExcelExport	Success	6 s
rebean_wi.customers.US_Dept_of_Health_And_Human_Svcs.ID_11111.CM_672308_2014 CM_672308_2014_MultivaluedIssue	Success	6 s
rebean_wi.customers.Warner_Bros.ID_132729.CM_97749_2015 CM_97749_2015_MissingData	Success	13 s
rebean_wi.customers.Zurich.ID_960077.CM_495925_2014_TP CM_495925_2014_ExportTXT	Success	51 s
rebean_wi.customers.Zurich.ID_960077.CM_721110_2014_TP CM_721110_2014_PreviousIssue	Success	4 s
rebean_wi.customers.Zurich.ID_960077.CM_932326_2014_TP CM_932326_2014	Success	2 s

[Properties »](#)



Tous les tests implémentés

DATE	ID	SUMMARY	Testcase name	LINK
04/08/2014	6665695	add a testplan		<a href="#">08\04082014071708.xps</a>
04/08/2014	6665796	Add the testplan in the testsuite. Opening and outline tests.	- BIT BIWEBISL2_1464_OpenADoc - BIT BIW EBIS L3_1 605_ outli ne	<a href="#">08\04082014084249.xps</a>
05/08/2014	6669508	Add wid files for webi_perf_wrkf lw_amandine		<a href="#">08\05082014031026.xps</a>
11/08/2014	6696723	Add testcase for break header	CM_1276108_2014_MergeBreakHeaderTC	<a href="#">08\11082014084911.xps</a>
13/08/2014	6706082	Add testcase for input controls upgrade	CM_1386568_2014_InputControlsUpgrade	<a href="#">08\13082014093545.xps</a>
13/08/2014	6706165	Modify the test file		<a href="#">08\13082014104552.xps</a>
14/08/2014	6709692	Modify test case about upgrade		<a href="#">08\14082014020039.xps</a>
14/08/2014	6710130	Add test case architecture for ADAPT01709014_spacing	ADAPT01709014	<a href="#">08\14082014040011.xps</a>

20/08/2014	673542 9	Add testcase for check spacing. Add an html output for IE9	ADAPT01709014	<a href="#">08\20082014 082612.xps</a>
20/08/2014	673555 4	Add reference for ADAPT0170901 4		<a href="#">08\20082014 093427.xps</a>
21/08/2014	673979 2	Add testcase for CM_1445386_2 014	CM_1445386_2014_n oConsistentData	<a href="#">08\21082014 032600.xps</a>
21/08/2014	674000 1	Add static testcase for CM_1445386_2 014	CM_1445386_2014_n oConsistentData	<a href="#">08\21082014 064431.xps</a>
22/08/2014	674453 4	Add a txt_ reference for CM_1445386_2 014		<a href="#">08\22082014 030624.xps</a>
25/08/2014	675662 2	Add a txt_ ref for CM_1445386_2 014		<a href="#">08\25082014 031203.xps</a>
05/09/2014	680699 8	Add a new helper file specific to aurora		<a href="#">09\05092014 062655.xps</a>
09/09/2014	682068 8	Improve AuroraHalper and complete		<a href="#">09\09092014 021645.xps</a>

---

the search universe method				
17/09/2014	685653 5	Add testcase CM_1451542_2 014. Reopen doc after restart server	CM_1451542_2014_ actionCannotPerformed	<a href="#">09\17092014 064619.xps</a>
19/09/2014	686615 2	Add query spec for CM 1451542 2014		<a href="#">09\19092014 024806.xps</a>
19/09/2014	686658 2	Add testcase for CM 659376_2014	CM_659376_2014_Mis spelled	<a href="#">09\19092014 081811.xps</a>
19/09/2014	686661 9	Add document save for CM 659376 2014	CM_659376_2014_Mis spelled	<a href="#">09\19092014 085603.xps</a>
22/09/2014	687858 7	Improve testcase CM_659376_20 14	CM_659376_2014_Mis spelled	<a href="#">09\22092014 031008.xps</a>
22/09/2014	687911 7	Modify the save of documents in personal documents		<a href="#">09\22092014 093842.xps</a>
23/09/2014	688350 7	complete auroraHelper	CM_659376_2014_Mis spelled	<a href="#">09\23092014 040249.xps</a>
24/09/2014	688881 9	save in folder when savingDoc is true	CM_1451542_2014_ actionCannotPerfor med	<a href="#">09\24092014 032401.xps</a>

---

25/09/2014	689466 9	Add setFormula method in auroraHelper		<a href="#">09\25092014 102220.xps</a>
25/09/2014	689468 8	Modify test which use auroraHelper. The docInst setter of docCalc	CM_659376_2014_Mis spelled	<a href="#">09\25092014 102742.xps</a>
26/09/2014	689989 3	Add test case for CM_704895_20 14	CM_704895_2014_For matting	<a href="#">09\26092014 094053.xps</a>
<b>31/10/2014</b>	<b>705673 5</b>	Add a test case for change source feature		<a href="#">10\changesourc e.xps</a>
<b>13/11/2014</b>	<b>711350 4</b>	Add CM_840064_2014 _formulasChange	CM_840064_2014_for mulasChange	<a href="#">11\840064.xps</a>



# Annexe D

## Les bases pour comprendre Jenkins

cf annexe différence entre job et projet [http ://jenkins-ci.361315.n4.nabble.com/Is-it-called-quot-Project-quot-or-is-it-called-quot-Job-quot-td4628610.html](http://jenkins-ci.361315.n4.nabble.com/Is-it-called-quot-Project-quot-or-is-it-called-quot-Job-quot-td4628610.html)

cf annexe status JUnit vs status Jenkins

- [https ://wiki.jenkins-ci.org/display/JENKINS/Terminology](https://wiki.jenkins-ci.org/display/JENKINS/Terminology)

!!!! La base de l'implémentation d'un plugin

[https ://wiki.jenkins-ci.org/display/JENKINS/Extend+Jenkins](https://wiki.jenkins-ci.org/display/JENKINS/Extend+Jenkins)

[https ://wiki.jenkins-ci.org/display/JENKINS/Plugin+tutorial](https://wiki.jenkins-ci.org/display/JENKINS/Plugin+tutorial)





# Annexe E

## Maven

C'est quoi ?

Concrètement ça sert à quoi ?

Comment ça s'installe ?

Exemple d'utilisation : génération de skeleton du plugin Jenkins



# Index

Business Intelligence, 6

CMS, 11, 13, 15

Correction Measure, 13, 15

Defect, 19, 20

Framework, 7

Graphical User Interface, 10

JAR, 7

Java Correction WorkBench, 5, 13, 19, 20

Jenkins, 19

Jira, 5, 19, 20

lcmbiar, 7

Perforce, 5, 19

querspec, 7, 10, 14, 15

Rich Client, 6

SDK, 10, 16

Software Tester, 5, 8, 15

Source Code Management, 7, 13

Test coverage, 5

Testcase, 8

wid, 7, 8, 10, 11, 14, 15



# Table des figures

2.1	Le process de test . . . . .	6
2.2	Capture d'écran de WebI de l'arborescence du document de référence . . .	11
2.3	Capture d'écran de WebI de l'arborescence du document généré . . . . .	12
2.4	Capture d'écran de WebI de l'arborescence des références locales . . . . .	12
2.5	Écran de JCWB propre à une CM . . . . .	13
2.6	Écran de comparaison des 2 versions d'un même fichier (avant et après correctif) . . . . .	14
2.7	Diagramme UML des suites de tests . . . . .	14
2.8	Diagramme représentant les différents éléments qui compose la coquille vide d'un test dynamique . . . . .	17
2.9	Capture de l'écran de propriété d'un document WebI . . . . .	18
3.1	Le process utilisé lors de gestion d'un bug . . . . .	21
4.1	L'environnement d'utilisation du plugin . . . . .	24
4.2	Architecture d'un plugin Jenkins généré par maven . . . . .	28
4.3	Jenkins à l'exécution de la commande maven qui l'encapsule . . . . .	30
4.4	Extrait de la javadoc de la classe ListView . . . . .	30
4.5	Chemin d'accès au fichier de configuration par défaut . . . . .	31
4.6	Initialisation des css avant chargement . . . . .	32
4.7	Initialisation des css après chargement . . . . .	32
4.8	Algorithme de création du Dashboard . . . . .	33
4.9	Diagramme de classe du plugin réalisé . . . . .	34
A.1	Les différents niveaux de test . . . . .	37



## Liste des tableaux





# Table des matières

<b>Remerciements</b>	<b>ii</b>
<b>Sommaire</b>	<b>v</b>
<b>Introduction</b>	<b>1</b>
<b>1 Présentation de l'entreprise</b>	<b>3</b>
1.1 SAP dans le monde . . . . .	3
1.2 Contexte actuel . . . . .	3
1.3 Présentation de l'équipe . . . . .	3
<b>2 Software tester</b>	<b>5</b>
2.1 Généralités sur le test . . . . .	5
2.2 Le test à SAP . . . . .	5
2.3 Présentation du produit testé : WebI . . . . .	6
2.4 Préparation aux tests . . . . .	7
2.5 Déroulement de la 1 <sup>ère</sup> mission . . . . .	7
2.5.1 Tests statiques ou dynamiques . . . . .	7
2.5.2 De l'étude à l'intégration . . . . .	13
2.6 Bilan de mes 1 <sup>ères</sup> missions . . . . .	16
2.6.1 Les 1 <sup>ères</sup> erreurs . . . . .	16
2.6.2 Les acquis . . . . .	16
<b>3 Migration Jira/Java Correction WorkBench</b>	<b>19</b>
3.1 Qu'est-ce que Jira et Java Correction WorkBench . . . . .	19
3.2 Présentation du contexte . . . . .	19
3.2.1 Plugin de reporting . . . . .	20
3.3 Process actuel . . . . .	21
3.4 Travail effectué . . . . .	22
3.4.1 Fonctionnement et utilisation de prod-pass-access . . . . .	22
3.5 Résultats obtenus . . . . .	22

<b>4</b>	<b>Plugin Jenkins</b>	<b>23</b>
4.1	Contexte initial . . . . .	23
4.2	Étude préalable . . . . .	26
4.3	Le 1 <sup>er</sup> plugin . . . . .	26
4.3.1	Le squelette du plugin . . . . .	27
4.4	Implémentation de la base du nouveau plugin . . . . .	29
4.4.1	Import du code dans l'IDE . . . . .	31
4.4.2	Implémentation de la base . . . . .	31
4.5	Travail réalisé . . . . .	32
	<b>Conclusion</b>	<b>35</b>
<b>A</b>	<b>Les niveaux de test</b>	<b>37</b>
<b>B</b>	<b>Résultats quotidien des tests</b>	<b>39</b>
<b>C</b>	<b>Tous les tests implémentés</b>	<b>43</b>
<b>D</b>	<b>Les bases pour comprendre Jenkins</b>	<b>49</b>
<b>E</b>	<b>Maven</b>	<b>51</b>
	<b>Table des figures</b>	<b>53</b>
	<b>Liste des tableaux</b>	<b>55</b>
	<b>Table des matières</b>	<b>57</b>



**Titre du résumé**  
**sous-titre de mon résumé**

---

**Résumé**

Ici, mon résumé en français long de 200 mots maximum

---

**Mots-clefs**

---

1. Software tester
2. JavaEE
3. Agile
4. Jenkins

**English title**  
**Sub-title**

---

**Abstract**

Here, an 200 words maximum long English abstract

---

**Keywords**

---

Ici, une liste de mots-clés